Modelling Incremental Self-Repair Processing in Dialogue

Julian Hough

Submitted for the degree of Doctor of Philosophy

Queen Mary University of London

2015

Declaration

I hereby declare that the work presented in this thesis is my own work carried out under normal terms of supervision and that the research reported here has been conducted by myself unless otherwise indicated.

Julian Hough
London, 23rd November 2014

Funding acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) Doctoral Training Account (DTA) scholarship from the School of Electronic Engineering and Computer Science at Queen Mary University of London.

Modelling Incremental Self-Repair Processing in Dialogue

Julian Hough

Abstract

Self-repairs, where speakers repeat themselves, reformulate or restart what they are saying, are pervasive in human dialogue. These phenomena provide a window into real-time human language processing. For explanatory adequacy, a model of dialogue must include mechanisms that account for them. Artificial dialogue agents also need this capability for more natural interaction with human users. This thesis investigates the structure of self-repair and its function in the incremental construction of meaning in interaction.

A corpus study shows how the range of self-repairs seen in dialogue cannot be accounted for by looking at surface form alone. More particularly it analyses a string-alignment approach and shows how it is insufficient, provides requirements for a suitable model of incremental context and an ontology of self-repair function.

An information-theoretic model is developed which addresses these issues along with a system that automatically detects self-repairs and edit terms on transcripts incrementally with minimal latency, achieving state-of-the-art results. Additionally it is shown to have practical use in the psychiatric domain.

The thesis goes on to present a dialogue model to interpret and generate repaired utterances incrementally. When processing repaired rather than fluent utterances, it achieves the same degree of incremental interpretation and incremental representation. Practical implementation methods are presented for an existing dialogue system.

Finally, a more pragmatically oriented approach is presented to model self-repairs in a psycholinguistically plausible way. This is achieved through extending the dialogue model to include a probabilistic semantic framework to perform incremental inference in a reference resolution domain.

The thesis concludes that at least as fine-grained a model of context as word-by-word is required for realistic models of self-repair, and context must include linguistic action sequences and information update effects. The way dialogue participants process self-repairs to make inferences in real time, rather than filter out their disfluency effects, has been modelled formally and in practical systems.

Submitted for the degree of Doctor of Philosophy

Queen Mary University of London

2015

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgements

This thesis has been made possible by several amazing people I am lucky to work with or generally have in my life. I will no doubt omit many and cannot do justice to those mentioned here.

Firstly, my supervisor Matt Purver has provided exactly the guidance I needed throughout. I never once left a supervision meeting not feeling better than when I knocked on his oft knocked-on door, despite sometimes stale progress to report. He has always had time for correspondence, writing, feedback and meetings to lend his vast technical and linguistic expertise in a generous, fair and measured way, and hopefully we've shared a laugh occasionally too. I sincerely hope his clear-mindedness has rubbed off on me. I cannot thank him enough.

Next, I thank Pat Healey for admitting me to the fantastic Cognitive Science group and supporting me throughout, being a great source of inspiration and exchange of ideas on real interactivity, often at the pub. Ruth Kempson, the third member of my progress panel has been one of the great inspirations of the ideas in this thesis, and her burning passion for all things linguistic, incremental and interactive has been a joy to be around.

Thanks to my examiners Nick Asher and Wilfried Meyer-Viol for their insightful comments at my viva and pointing the way to future directions, offering their extensive experience very generously. They reminded me how much work is still to be done in our field, which is exciting for the future of dialogue models and systems in general.

I've had the chance to work with and become friends with some superlative people at Queen Mary and the University of London during this PhD. Arash Eshghi has been a motivating and passionate force for getting to the heart of the matter in everything we do, and I hope this continues well into the future. Chris Howes has been a wonderful office neighbour, collaborator and motivator, and her encouragement to apply my work to something more tangible and practical has been fantastic. Suzanne Labelle originally encouraged me to do linguistics and supported me throughout. Kavin Narasimhan and I have struggled in parallel with our projects and have kept each other going, which has been great fun. This was similarly the case with Huayi Huang, who was also kind enough to help out with annotation tasks. It has been fantastic variously working with, hanging out with and debating with CogSciers and related friends Eleni Gregoromichelaki, Nicola Plant, Jeni Maleskhova, Robin Fencott, Claude Heath, Chrystie Myketiak, Ioana Dalca, Pollie Barden, Saul Albert, Fiore Martin, Matthew Gotham, Dmitrijs Milajevs, Stephen McGregor, Graham White, Chrisantha Fernando, Hamed Haddadi and Geraint Wiggins. Finally in this regard, the band Haskell, formed and fronted by the enigmatic and excellent Shalom Lappin has been fantastic to be part of. I give credit to the enduring spirit of Peter Ridley, Stelios Chatzikyriakidis, Sam Duffy and 'manager' Peter Sutton, the other members of University of London's inaugural country and blues band that is also a functional programming language.

It is often said the buildings don't make a place but the people do, and this is certainly the case at my new home at Bielefeld University's Dialogue Systems (DSG) Group. I was fortunate enough to get a job with the best incremental dialogue systems group in town with David Schlangen, whose work has heavily inspired this thesis, and who was kind enough to give me some time to complete it, despite being needed elsewhere. DSG'ers and friends Spyros Kousidis, Casey Kennington, Ting Han, Zina Sarriess, Soledad López Gambino, Birte Carlmeyer, Simon Betz, Laura de Ruiter, Sebastian Loth and Iwan de Kok have made this a welcome first year.

Now to briefly thank the people outside my academic arena who have given their energy and encouragement in helping me complete this thing. In London over the last ten years I've

been fortunate to be around these people, in roughly the order I met them: Simon Garrard, Rupert Grinling, Alex Greer, Dickie Ellis, Charl Green, Hollie and Edd Harrison, Laura and Jack Gilbert, Cat and Youseff Elgonaid, Toby and Nicki Welch, Susie Garrard, Genia and Alex Marek, Tim Reynolds and Edd Lee. Older friends from Stroud have been an emotional engine, again in roughly the order of meeting: Sam Wyatt, James Stevenson, Hannah Swain, Kate Gillingham, Iain Whitfield, Fletcher Brown, Toby Ireland, Ander and Tom Russell, Carl Jenkins, Andy Smith, Will Searight, Pat Van der Waals, Claudia Copestake and Dave Wiseman. There are others who deserve mention and I am grateful to them all.

My sister Sophie has also always been there for me and my Auntie Rebecca and Uncle Bill have been most generous and encouraging throughout.

Finally, this thesis is dedicated to my Mum and Dad, whose generosity has never known any bounds, despite me causing them so much stress over the years. Thank you.

And a post-finally thank you to Lauren who has made the last couple of years the best of my life.

Previously Published Material

Content from the following publications appears in this thesis. All of the following have been written with the guidance of, and in collaboration with, my supervisor Matt Purver, and in Howes et al. (2014) also in collaboration with Chris Howes and Rose McCabe.

Julian Hough. 2011. Incremental semantics driven natural language generation with self-repairing capability. In *Proceedings of the Student Research Workshop associated with RANLP 2011*, pages 79–84, Hissar, Bulgaria (Chapter 6)

Julian Hough and Matthew Purver. 2012. Processing self-repairs in an incremental type-theoretic dialogue system. In *Proceedings of the 16th SemDial Workshop on the Semantics and Pragmatics of Dialogue (SeineDial)*, pages 136–144, Paris, France (Chapter 6)

Julian Hough and Matthew Purver. 2013. Modelling expectation in the self-repair processing of annotat-, um, listeners. In *Proceedings of the 17th SemDial Workshop on the Semantics and Pragmatics of Dialogue (DialDam)*, pages 92–101, Amsterdam (Chapter 4 and Chapter 5)

Julian Hough and Matthew Purver. 2014a. Probabilistic type theory for incremental dialogue processing. In *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pages 80–88, Gothenburg, Sweden. Association for Computational Linguistics (Chapter 7)

Julian Hough and Matthew Purver. 2014b. Lattice theoretic relevance for incremental reference processing. Edinburgh. Poster presentation at REFNET Workshop on Computational and Psychological Models of Reference Comprehension and Production (Chapter 7)

Christine Howes, Julian Hough, Matthew Purver, and Rose McCabe. 2014. Helping, I mean assessing psychiatric communication: An application of incremental self-repair detection. In *Proceedings of the 18th SemDial Workshop on the Semantics and Pragmatics of Dialogue (DialWatt)*, pages 80–89, Edinburgh (Chapter 5)

Julian Hough and Matthew Purver. 2014c. Strongly incremental repair detection. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 78–89, Doha, Qatar. Association for Computational Linguistics (Chapter 5)

Chapter 1

Introduction

I begin with examples of the phenomena this thesis is concerned with in the context of a dialogue:

	GARETH	Do you know who done the picture?
→	KEITH	*Yeah... no, I mean no.*
	GARETH	Right. Your first answer was 'yeah', wasn't it?
	KEITH	I meant no.
	GARETH	Well, why did you get...?
→	KEITH	*Uhh*... I don't know.
	GARETH	Am I making you nervous?
→	KEITH	*No. I mean, yeah.*
	GARETH	Hmmm. That's interesting.

('The Office', BBC comedy, Series 1, 2001, episode 2.)

Self-repairs and edit terms, the concern of this thesis, are italicised, underlined and marked with an arrow. This example shows the interactive effect of repaired utterances and how they construct meaning in dialogue. People are aware of their own and their dialogue partner's disfluencies, and process their meaning as easily and quickly as they do fluent contributions.

I believe the remit of a model of human language comprehension and production needs to include mechanisms for self-repair for it to have explanatory adequacy. From the beginning of the field of conversation analysis (CA) it was observed that they are abundant in natural conversation, take varied forms and adopt different communicative functions (Schegloff et al., 1977; Schegloff, 1992). Self-repairs are the most common type of repair phenomena in dialogue as the initial CA work and more recent statistical corpus work has shown, appearing with different frequency between dialogue domains and speaker groups (Shriberg, 1996; Bortfeld et al., 2001; Colman and Healey, 2011). Different types of self-repair can be elicited from speakers in experiments,

both in description tasks (van Wijk and Kempen, 1987) and in dialogue settings (Healey et al., 2011) with some degree of predictability. Experimental results have shown how hearers process speech repairs as easily as fluent utterances, and in fact in specific cases the speed of incremental semantic processing can be increased when hearing repaired, rather than fluent, speech (Brennan and Schober, 2001). Finally, the distribution of self-repairs across speakers has been a useful tool in the psychiatric domain as it can predict outcomes for patient adherence to treatment (McCabe et al., 2013).

Despite this overwhelming empirical evidence that self-repair is a core mechanism of linguistic and cognitive processing, it is positioned at the periphery of theoretical and computational linguistics. In fact, self-repairs are rendered anomalous and unworthy of formal treatment by many mainstream accounts of the human language faculty, a stance originally endorsed by the formal linguistic *competence-performance* distinction championed by Chomsky (1965). Consequently, due to its status as a "performance" phenomenon, describing and explaining self-repair was not a motivating factor for popular theories of syntax such as the principles and parameters approach (Chomsky, 1981) or the minimalist program (Chomsky, 1995), whose objective was to maximise descriptive coverage of structures over written strings of different languages, a stance which much of theoretical linguistics has adhered to since. It has been left to formal accounts of dialogue to characterize self-repair in a systematic way, and only in recent preliminary accounts (e.g. Ginzburg et al., 2007, 2014).

In terms of the distribution of research effort, the story is analogous in computational linguistics, wherein the most often used task for training and evaluating parsers is the transduction of trees from sentences in the Wall Street Journal section of the Penn Treebank (Paul and Baker, 1992), an American English text corpus. This is a task that clearly does not pose the full challenge of interpreting on-line natural dialogue data where self-repairs and other non-sentential phenomena are the norm rather than the exception. Despite this field bias, the work begun by Shriberg (1994) began to comprehensively deal with speech repairs and other disfluencies by providing annotation schemes for dialogue transcripts and developing computational models for integrating disfluency detection within speech recognition, spawning a recognized detection task that has attracted various statistical approaches (Shriberg and Stolcke, 1998; Liu et al., 2003; Maskey et al., 2006). More linguistically motivated detection and correction approaches have been taken with Heeman and Allen (1999)'s multiple knowledge source based model for im-

proving speech recognition through finding structural patterns of repairs and discourse markers, Johnson and Charniak (2004)'s string-alignment approach to detecting disfluencies and Snover et al. (2004)'s lexically-driven model for detection, all addressing the task of automatic structural processing of self-repairs. While some of these approaches are linguistically inspired, they treat repair detection as a tagging problem orthogonal to other linguistic processing and as a method to filter out disfluencies before processing cleaned utterances; this thesis sets out to move away from such an approach.

Modelling self-repair and disfluency has been more of an imperative for models of human speech production (van Wijk and Kempen, 1987; Levelt, 1983, 1989) and psycholinguistically motivated NLG (natural language generation) (De Smedt, 1990, 1991; Neumann, 1994; Guhe, 2007) rather than comprehension or parsing. The earlier psycholinguistic models used speech repairs to give insights into the mechanisms underlying speech production, for example analysing self-corrections in terms of recovering from incorrect lemma selection from the lexicon (van Wijk and Kempen, 1987) or the evidence for self-monitoring being a crucial part of speech production (Levelt, 1983, 1989). As will be described, these approaches have inspired implementations into working interactive dialogue systems (Skantze and Hjalmarsson, 2010; Buß and Schlangen, 2011), albeit with simplifications for the purposes of practical system building.

In modelling self-repair computationally, I follow an approach akin to Mark Johnson's proposed modification of the parsing paradigm:

> "One way to achieve an open-world approach to parsing while maintaining the standard closed-world conception that grammars generate only grammatical analyses is to abandon the claim that a parse is a proof of the grammaticality of the input sentence. One way to do this is to *incorporate explicit models of disfluencies into the parsing process*."
> Mark Johnson. 2011. How relevant is linguistics to computational linguistics? *in Linguistic Issues in Language Technology. Vol. 6 (7).CSLI*

While combination of parsing and disfluency detection has recently been attempted in line with this proposal (Rasooli and Tetreault, 2013, 2014; Honnibal and Johnson, 2014), this thesis intends to go beyond this by not only extending repair processing ability to full NLU (natural language understanding), NLG and dialogue management models, but also to the grammar (the general linguistic processing model), setting out to propose unifying repair mechanisms shared by all these processes.

1.1 The importance of self-repair in dialogue

The ostracism of self-repair data by formal accounts of language processing, which may be due to methodological preferences and philosophical sensibilities, stands in contrast to an empirical stance of building models that account for real dialogue data. In these models self-repair phenomena are given equal importance to fluent utterances. The seminal CA paper that highlighted the regularity and systematicity of self-repairs noted that they vary in cause and form, and are "neither contingent upon error, nor limited to replacement" (Schegloff et al., 1977, p. 363)– in line with this, this thesis intends to explore the fullest taxonomy of self-repair that can be observed in dialogue. Directly parallel to this, it addresses the more practical question of how a dialogue system should be designed to process self-repairs and fluent utterances in an integrated and efficient way. These strands of enquiry are very much part of the same effort, under the assumption that a computationally implemented dialogue system can be seen as a cognitive model (McDonald, 1987; Schlangen, 2009).

It is surprising that dialogue theorists and dialogue system designers have dedicated disproportionately little research effort to self-repair, instead favouring treatment of 'higher-level' repair phenomena that are considered more contingent on interaction – for example clarification requests (Purver, 2004; Rodríguez and Schlangen, 2004; Ginzburg, 2012). However, self-repair need not be seen as a 'low-level' process which is confined to a correction mechanism at the phonetic or lexical levels, although a subset could be described as such. Not only do people process self-repairs and other disfluencies as easily as fluent utterances, but furthermore, people can be aware of them phenomenologically. For example, internal states such as forgetfulness or nervousness can be demonstrated and inferred by dialogue participants as the opening example and the one below demonstrate (again self-repairs are italicised and marked with an arrow but here the reference to the self-repair from the other dialogue participant is underlined, with notes in square brackets):

	PAMELA	the illusions of this life, in you know, I
→	PAMELA	*in (.) I, uh, I, I.*
	DARRYL	<u>eh be eh be</u> [imitating a stutter]
	PAMELA	my favourite word when I was twelve was paradox.

(Santa Barbara corpus, file SBC0005.
Husband and wife free conversation.)

Darryl and Gareth's demonstration of the awareness of their interlocutor's self-repair show that accounts which excise the repaired material at the phonetic level, before semantic processing,

are clearly unsatisfactory, not just in terms of theoretical plausibility, but also in terms of forming the basis for dialogue systems with more natural interaction capabilities. An interactive approach to the phenomena is clearly preferable, as disfluency and repair markers can be seen as signals optimal for not just the speaker's formulation purposes, but also for the hearer's understanding that the interlocutor has a production problem or intends to change their prior contribution, under the assumption of least joint communicative effort (Clark, 1996).

From an implementational perspective, recent interactive systems with self-repairing capability have been shown to have positive effects on user experience, resulting in interactional advantage and greater efficiency in task-based micro domains (Skantze and Hjalmarsson, 2010; Buß and Schlangen, 2011; Kousidis et al., 2014), even though the self-repair capability is confined to simplified generation and synthesis and dialogue management components, rather than interpreting their meaning in NLU. While intuitively, it seems odd and impractical to design a computational system that 'makes mistakes' in its vocal interaction with human users, in fact not only do systems which generate self-repairs seem more pleasant to use, a system with natural self-repair behaviour can increase communicative clarity. For example, information giving dialogue systems that can revise their communicative goals and make this obvious to a listener are far clearer, as can be seen in the preference for (1.2) over (1.1) below:

(1.1) System: I have two seats available. I have one seat available.

(1.2) System: I have two seats...uh no... one seat available.
(Guhe and Schilder, 2002)

Recently, modelling the dialogue semantics of self-repair has also received attention through Ginzburg and colleagues (Ginzburg et al., 2007; Ginzburg, 2012; Ginzburg et al., 2014), though this currently lacks an incremental grammar to interface with their higher level discourse model. Given the lack of a unified account of self-repair within a single framework, one question which this thesis addresses is: *Where should self-repair processing be situated within a dialogue system, and how are parsing, generation and dialogue models of self-repair connected?*

1.2 The need for an incremental processing approach

This thesis investigates self-repair from an incremental perspective. While the technical meaning of incrementality and its different aspects in dialogue will be fleshed out later, here I introduce

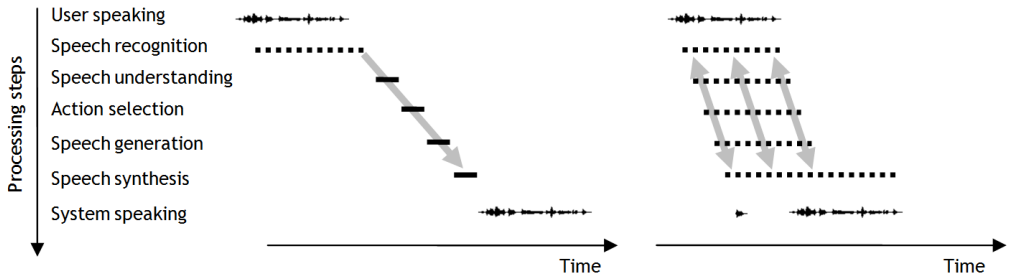


Figure 1.1: Non-incremental (left) and incremental (right) dialogue systems from Skantze (2014)

this as *the construction of meaning in real time as linguistic information is processed*. A stronger motivation for an on-line processing model of human language beyond the requirements of modelling self-repair comes from an evolutionary perspective. It is obvious humans need the ability to interpret information in incoming linguistic stimuli as quickly as possible: as Briscoe (1987) and Milward (1991) point out, the inability to process incomplete utterances such as "Look out, there's a fall-…" would surely result in a survival disadvantage. On-line build-up of meaning is a more realistic view of interpretation rather than the delayed computation over complete sentential utterances, and so partial structures should be as well studied as complete ones. In language production, given a hearer can *decode* information on-line and incrementally, it must surely be possible for a speaker to *encode* that information in their speech using a similarly incremental build-up of meaning, therefore both the comprehension and production processes must share similar mechanisms. In this regard, *compound contributions*, where a speaker begins an utterance and their interlocutor then completes or extends it, have shown to be pervasive in dialogue (Howes et al., 2011), and are very much a shining beacon for the incremental perspective in addition to the repair phenomena described here.

As well as being empirically motivated, the computational benefits of incrementality are clear. Several recent approaches show promising benefits to interactive systems: DeVault et al. (2011) show how partial automatic speech recognition (ASR) results can be incrementally used to narrow the space of possible semantic representations for a semantic parser and Skantze and Schlangen (2009) show how partial ASR results can be used for increasing speed of interpreting user input in micro-domains in dialogue. These systems fulfil one central desideratum of incremental processing, termed *Wundt's Principle* by Willem Levelt, which is that each processing component should be triggered into activity by a minimal amount of its characteristic input

(Levelt, 1989). Schlangen and Skantze (2009) introduce a dialogue system framework built on this principle, which can be summarized as in Figure 1.1 where the processing steps of a non-incremental dialogue system can be seen on the left-hand side while an incremental one can be seen on the right. In the non-incremental system, the processing steps, beginning with that closest to the speech input, the ASR, must complete in their entirety before passing their results to downstream modules, whereas the processors in the incremental system may begin processing with partial input, and output partial results, resulting in shorter delays to the final response, and also engaging the higher-level modules such as speech understanding early on to guide the lower-level decisions.

Several other principles of incrementality will be described in detail in Chapter 3, and given a suitable definition and methods for evaluating them, the second major underlying question to this thesis can be posed: *To what extent can the principles of incremental dialogue processing be adhered to in self-repair processing, and what changes to dialogue models are required to make this possible?*

1.3 The aims and scope of this thesis

In order to address the above questions, the analytic part of this thesis falls into two main sections: (i) dialogue corpus analysis and building of empirical models (ii) building of formal and computational models.

1.3.1 Empirical corpus work and automatic repair detection

As will be explained, corpus studies on disfluencies in natural dialogue (Shriberg, 1994, 1996; Shriberg and Stolcke, 1998; Heeman and Allen, 1999; Bortfeld et al., 2001; Colman and Healey, 2011) variously fall short in analysing self-repair phenomena in sufficient detail to build a broad-coverage empirical model. The precise short-comings will be discussed in Section 2.5. The corpus work in Chapter 4 aims to be sufficiently detailed as to inform the building of repair processors in detection, parsing and generation which can process self-repair incrementally. The analysis will be multi-faceted, using multiple knowledge bases such as syntactic, lexical, semantic and pragmatic information, in the spirit of Heeman and Allen (1999) and Liu et al. (2003), but with an aim to produce *interpretable* results rather than a pure application evaluation that previous practical approaches to the problem have carried out. Chapter 5 then describes the building of

an evaluated incremental repair detector that improves on the incremental performance of state-of-the-art repair detection systems with a model that is time-linear and strongly incremental in its operation.

1.3.2 Formal modelling and dialogue system module design

The principal contribution for formal and implemented dialogue systems is the analysis, design and evaluation of parsing and generation models for self-repairing capability, as well as for their general incremental processing requirements. Chapter 6 describes a semantics-driven model of generation and parsing which incorporates self-repair in a psycholinguistically plausible and computationally efficient way, with methods for practical implementation also given. Chapter 7 extends the model to account for psycholinguistic results, and to achieve this it requires access to a situation model, rather than just a generation and parsing context. The models and prototype implementations are not extended to voice synthesizers and automatic speech recognizers (ASR). This is not due to any theoretical commitment, but more due to keeping the problem defined in terms of syntactic-semantic processing for clarity's sake and of course due to limitations of time for completion of this project. However, considerations as to how the parsing and generation modules developed interface with state-of-the-art ASR and synthesis will be taken.

1.4 Structure of this thesis

The structure of the remainder of this thesis is as follows:

Chapter 2: A review of empirical approaches to characterizing and investigating self-repair with proposed directions for research.

Chapter 3: A review of computational and formal approaches to modelling, detecting, parsing and generating self-repair phenomena with proposed directions for research, including the incremental requirements for dialogue systems and self-repair models.

Chapter 4: A corpus study investigating the relationship between structural, syntactic, lexical, semantic and dialogue-level context and the presence and form of repairs.

Chapter 5: Definition and evaluation of an incremental self-repair detection system STIR which achieves state-of-the-art incremental performance.

Chapter 6: Definition of a self-repair processing model into a formal dialogue framework and prototype incremental NLU and NLG modules.

Chapter 7: Extension of the framework in Chapter 6 so the modules interface with a probabilistic semantics to model the semantic and pragmatic effects of self-repairs and incremental processing in general in dialogue.

Chapter 8: Conclusions from the thesis and discussion on potential future work arising from it.

Chapter 2

Empirical Approaches

Self-repair has been studied within a variety of empirical paradigms, including conversation analysis (CA), experimental psychology and corpus linguistics. These approaches consider the various forms, causes, effects, statistical distribution, classification, utterance and dialogue position, syntactic, phonological and phonetic aspects of self-repair, with the principal intention to provide descriptive and explanatory cognitive models of the phenomenon. The key approaches will be outlined here, with assessment of their coverage and the assumptions they adopt. Section 2.5 summarizes where this previous work is inadequate, in terms of its failure to describe and explain all the self-repair phenomena in sufficient detail for broad-coverage and potential for computational implementation, and directions for future empirical work are outlined.

2.1 The form of self-repair events

Self-repair phenomena in spoken language are events where a speaker will begin an utterance and, at some point after commencing, noticeably reformulate or restart their current utterance to continue their contribution, or else extend or repair an entire contribution after an apparent turn completion, either immediately after their own contribution or after an interposed contribution from another conversation participant. To illustrate the phenomena under discussion, the following are typical self-repairs found in natural spoken dialogue of the first position (same-turn), self-initiated type:

(2.1) "... but my kids are only elementary [grades, + levels] right now"

(Switchboard conversation number sw4325)

(2.2) "And because [this is such + this is for television], [[it's a + we have a market range of] + {like,} it's an international market range]"

(AMI corpus)

(2.3) "Peter went [swimming with Susan, + {or rather,} surfing] yesterday"

(Constructed example from anonymous Semdial 2012 reviewer)

For terminological and annotation purposes, following Shriberg (1994); McKelvie (1998) and the Switchboard corpus disfluency annotation schema (Meteer et al., 1995), first position self-repairs will be discussed in terms of a division into the part of the utterance before the repair, which can be called the *original utterance*, the *reparandum* (the part of the contribution that is repaired, notationally the strings from the opening square bracket up to the repair point +), a possibly null *interregnum* (the words between the {} brackets) and the following *repair* (the words after the *interruption point* + up to the closing square bracket), and finally the *continuation* of the utterance after the end of the repair, in the below structure:

$$\underbrace{\text{John}}_{\text{original utterance}} \quad \underbrace{[\ \text{likes}}_{\text{reparandum}} \ + \ \underbrace{\{\text{uh}\}}_{\text{interregnum}} \quad \underbrace{\text{loves}\ ]}_{\text{repair}} \quad \underbrace{\text{Mary}}_{\text{continuation}} \tag{2.4}$$

The structural divisions made in (2.4) not only provide consistent terminology for technical discussion but are also important for practical reasons: automatic processing of self-repairs, as will be shown in Chapter 3 relies on the persistence of these divisions, despite the fact that not all of the above components of a repair are always present.

For hearers and computational interpretation systems, detecting these structures is not a trivial task, however it is a necessary one to compute the meaning of a repair in a dialogue. Much of this thesis investigates the capacity of a system to detect these structures in order to compute the semantics of the repaired utterance appropriately. On the production side of the repair process, speakers and generators who produce such utterances must decide on-line where to begin repairing and which part of their utterance to repair. While these decisions are influenced by far more than just the surface form or acoustic context of their utterances, speakers and generators use this information in their utterance incrementally built up so far to decide what they are repairing and how to do it. In terms of the application of this annotation structure, an interpreter needs to decide where to place the brackets and cross and compute the inferences made by the detection,

and a generator needs to know how to generate utterances such that they adopt these structures to allow appropriate interpretation.

Using the surface analysis afforded by the bracketing system, several self-repairs can occur within the same utterance, some in succession, and some resulting in recursive embedding of the annotation structure, as shown from the second (nested) self-repair onwards in example (2.2), above. Also, as will be described below, I broaden the remit of first position self-repair beyond reformulation and restarting, as is demonstrated by what I will call, for now, *edit terms* such as (2.5), where, unlike the other types there is no verbal reparandum and consequently no downgrading of discourse status of any part of the utterance built up so far (at least on a lexical-semantic level) with the onset of the repair– these have also been called *abridged* repairs (Heeman and Allen, 1999), *foward-looking disfluencies* (Ginzburg et al., 2014) and in CA paradigms often fall within *transition space* repairs (Schegloff et al., 1977; Healey and Thirlwell, 2002). Stand-alone edit terms have a discourse function, and for now I commit to this being a repair-like function, but with different update effects on the dialogue context. Edit terms will be notated, as with interregna as falling within {} brackets and will be termed edit terms whether they fall within an interregnum or not.

(2.5) "John goes to Paris {uh} from London"

(Constructed example)

At this point, I briefly make the distinction between first position self-repairs and disfluencies: the former is not necessarily a proper subset of the latter, as utterances which are abandoned for external reasons may be disfluent, but with no attempt at repair. Repair involves work by the speaker to alter their utterances and this repair action has a communicative function in dialogue. While I focus on the most common type of self-repair, the first position variety, these fall within a bigger set which include *third position* and *third turn* self-repairs, which will be discussed in more detail below in Section 2.2.1.

2.2 CA classification and study of self-repair

It is useful to consider the distribution of self-repair phenomena in dialogue, both absolutely, and in terms of their distribution within all types of repair. Researchers in conversation analysis (CA) have studied self-repair since near the establishment of the field. Since CA is inherently

an interaction-oriented discipline, repair was analysed for its role in sequential turns in a dialogue (assuming shared time between the speaker and the hearer), rather than just for syntactic surface-form regularity. Schegloff et al. (1977) established the idea of an *organization of repair* in communication, making distinctions between different repair classes based on the *speaker* of the repair (*self*-repair or *other*-repair), the *initiator* of that repair — whether it was caused by the speaker (self-initiated) or the hearer (other-initiated) of the turn being repaired — and the *position* of the repair relative to the turn being repaired, which is its sequential position in dialogue turn structure, either *within* the same turn (*first position*), in the *transition space* after an apparently complete turn, in the immediately following turn (*second position*), or in the third turn after the initiation (*third position* or *third turn*)- see examples (2.6)-(2.11) below for the types of self-repair originally defined.

Schegloff et al. (1977) provided several important insights into self-repair phenomena, such as the fact they are not equivalent to speech errors: often there are no noticeable phonological or syntactic mistakes before the repair point (see the examples below) and conversely speech errors are often not repaired. Their categorizations are useful in that they showed systematicity and regularity of repair events within perspicuous dimensions. Schegloff et al. make the observation about repair in dialogue that "self-correction and other-correction are related organizationally, with self-correction preferred to other-correction" (ibid., p.362). Although lacking quantitative results, they showed through CA analysis some convincing evidence for the preference for self-repair through annotated examples.[1] In terms of the causes of these phenomena, they observed self-repair could issue from other-initiation, not just from self-initiation, and that this was a prevalent phenomenon: "other-initiations overwhelmingly yield self-corrections" (Schegloff et al., 1977, p.376)– this can be seen as a description of a repair initiated by a turn by the interlocutor called a *clarification request* (Purver et al., 2003) or *NTRI (Next turn repair initiator)* (Levinson, 1983, p.339). In analysing self-repair more systematically than had been done previously and implicitly arguing for the psychological systematicity of the mechanism, they also observed how repair *failure* could result from self-initiation, defining a failed repair attempt as one that is "marked by an overt withdrawal of the repair effort" (Schegloff et al., 1977, p.362,fn)– see the abandoned turn in example (2.7).

(2.6) "Sure enough ten minutes later [the bell r- + *the doorbell rang*]"'

[1] The claim is backed up with empirical distributional data in Colman and Healey (2011) that will be discussed below.

(self-initiated first position self-repair)

(2.7) '[Awl I her + [All I + [Awl I ree- + [all you- + all I ree-]]]]"
(failed first position repair issuing from self-initiation)

(2.8) "He had dis uh Mistuh [W- + {whatever k- I can't think of his first name} Watts] on, the one thet wrote that piece"
(self-initiated first position self-repair with long interregnum 'aside')

(2.9) M: He's stage manager.
(2.0)
M: He's actually first assistant but- he's calling the show
(self-initiated transition space self-repair)

(2.10) Hannah: And he's going to make his own paintings.
Bea: Mm hm
Hannah: And- or I mean his own frames.
(self-initiated third position self-repair (third turn))

(2.11) A: Have you ever tried a clinic?
B: What?
A: Have you ever tried a clinic?
(other-initiated third position self-repair)

The finding of heavy preference for self-initiated self-repair in dialogue was attributed to the fact that opportunities for self-initiation arise before opportunities for other-initiation, so when trouble is detected, it is highly likely to be resolved by the speaker who detects it as quickly as possible, either through a same-turn (first position) self-interruption and repair (example (2.6)), or through extending at the transition point (TP) after a seemingly complete turn (example (2.9)). However, a repair initiated by the repair maker's interlocutor (the hearer) after the trouble source can involve a third position self-repair (example (2.11)) (subtly different to *third turn* self-repair example (2.17), as will be explained below), the initiation normally taking the form of a clarification request.

2.2.1 Third position and third turn repair

Schegloff (1992) develops the previous CA analysis of third position (P3) repairs such as example (2.11) in detail, and claims the phenomenon is a prototypical case study of breakdown in inter-subjectivity and shared understanding in conversation. Schegloff carries out a detailed analysis of P3 repairs, identifying four structural components, which, while they are not always all necessarily present, can normally be found occurring in a canonical order. There is always a trouble source turn (**TS**) produced by the speaker, which could be considered as constituting the

reparandum. The components of the consequent repair turn defined from Schegloff's analysis of natural dialogues in American English are, in order, as follows:

- **A** -*Repair initiation*. Normally takes the form of a discourse marker such as "well", or "no", singly or in multiples, and occasionally replaced by "oh" or in combination with it (e.g. "oh no"). Schegloff makes the distinction between the onset of P3 repair and the onset of a disagreement turn clear.

- **B** - *Agreement/acceptance component*. The most likely component to be omitted. It almost exclusively appears in response to a second position complaint by the hearer.

- **C** - *Rejection component*. The speaker rejects the interpretation that he/she infers the turn containing the trouble source had been given by the hearer. The three alternating formats of this component are described as: (i) specification of a misunderstanding of problematic reference, often by the form "I don't mean *X*" (ii) specification of misunderstanding of illocutionary force, where the form is "I'm not *X*ing, where *X* is the name of some illocutionary action (e.g. "I'm not asking you..."), or (iii) the misunderstanding being referred to by a pronoun ("I don't mean *that*")- example (2.12) shows a P3 repair with components A, B and C.

- **D** - *The repair proper*. The component least likely to be omitted from the P3 repair structure. This may take the form of a "clearer repeat"- see the first attempt at repair, the **Drep** type in excerpt (2.15) below- or more likely use of the phrase "I mean" followed by an idiomatic contrasting paraphrase of the trouble source (e.g. the paraphrase by Ken in the D component in excerpt (2.13)) and/or a reformulation of the trouble source (see excerpt (2.14)) and/or specification (see Lehroff's "Is it- rain(ing)? uh windy? or what?" (**Dspec**) following the clearer repeat (Drep) in (2.15)) and/or an explanation type repair (see the caller's explanation in (2.16)). The other operation that may happen is characterizing the trouble source as non-serious if it was taken as serious, usually by means of a phrase to the effect of "I was joking".

(2.12) Excerpt showing "rejection" component C of a P3 repair from (Schegloff, 1992, p.1306)[2]

	Agnes:	I love it.
		(0.2)
(TS)	Portia:	Well, honey? I'll pob'ly see yuh one a' these day:s
	Agnes:	Oh:: God yeah
	Portia:	Uhh huh!
	Agnes:	[We-
	Agnes:	B't I c- I jis' couldn' git down there.
(A,B)	Portia:	[Oh- [Oh I know.
(C)		I'm not askin yuh tuh come dow-
	Agnes:	[Jesus. [I mean I jis'- I didn' have
		five minutes yesterday.

(2.13) Excerpt showing "paraphrase" form of D component of a P3 repair from (Schegloff, 1992, p.1310)

	Roger:	Yeah. But t(h)ell me is everybody like that or am I just out if it.
(TS)	Ken:	[I-Not to change the subject but-
	Roger:	Well don't change the subject. Answer me.
(A,D-,B)	Ken:	[No I mea- I'm on the subject
(B,D)		I'm on the subject. But- I-I mean "not to interrupt you but-" uh a lotta times I'm sitting in class, I'll start- uh I could be listening to the teacher and my mind'll be four million miles away.

[2]Traditional CA conventions apply to the excerpts here, and are presented as they were in the original text, with overlapping speech ([), lengthened syllables (e.g. "e:dge"), emphasis with underlining and pause length indicated with bracketed numbers denoting their length in seconds such as (3.0). The trouble source (TS) and the 3P A-D components which are present are annotated here on the left side. Many thanks to Saul Albert, who generously made available and explained the LaTeX code for CA-style transcripts on his research blog http://saulalbert.net/blog/2012/06/how-to-do-ca-transcriptions-in-latex/

(2.14) Excerpt showing "reformulation" type of D component of a 3P repair from (Schegloff, 1992, p.1311)

(TS)	Host:	Whaddiyuh afraid of.
	Caller:	I dun'kno:w, see uh
(A,D)	Host	Well I mean waitam'n. What kind of fear izzit. 'R you afraid yer gunnuh drive off the e:dge? 'R you afraid thet uh yer gonnuh get hit while yer on it?=

(2.15) Excerpt showing "clearer repeat" (Drep) and "specification" (Dspec) types of D component of a 3P repair from (Schegloff, 1992, pp.1311-12)

	Lehroff:	What is the weathuh. Out in that area now.
TS	Zebrach:	No winds, er it's squalling, rain, the winds are probably out of the north,- west, at uh estimated gusts of uh sixty five miles an hour.
	( ):	(Whew!)
	Zebrach:	Sustained winds of about thirty five to forty five miles per hour. And uh anticipated duration,
Drep,Dspec	Lehroff:	How is the wah- weather period outside. Is it-rain(ing)? uh windy? or what?

(2.16) Excerpt showing "explanation" form of D component of a P3 repair from (Schegloff, 1992, p.1312)

	Host:	Good evening, WNBC,
TS	Caller:	Good evening, this is uh, oh boy.
	Host:	ehh heh heh hyah hyah!
(A,D)	Caller:	[No I was listening to the commercial, and I'm just kinda- confused fer a min-ute

Schegloff (1997) treats *third turn* repairs as qualitatively differently to third position repairs - see example (2.17), where there is no claim of trouble or clarification in the interpolated turn.

(2.17)

TS	B:	And he's going to make his own paintings
	A:	Mm hmm.
A, Drep	B	*And- or I mean his own frames*
A:	Yeah	

(third turn repair, from (Schegloff, 1997, p. 32)

Some third turn repairs could be similar in form to transition space type self-repairs such as example (2.9). The distinction between these two assumes that the interposed turn from the conversation partner allows different possible continuations– Ginzburg et al. (2014) show the parallels and differences between these.

As can be seen, the detailed breakdown of third position self-repairs is an incredibly complex task, and is very difficult to achieve from transcripts; CA-level annotation is needed to get a procedural handle on describing the different forms and possible causes, and the interpretations are incredibly subtle. As will be seen below, identifying a third position self-repair is particularly problematic for human dialogue annotation: it requires ascertaining the initiator of the repair, which is often contingent on identifying the presence of a clarification request preceding it. Corpus studies on clarification requests have used multiple dimensions of dialogue information to characterize these phenomena (Purver, 2004; Rodríguez and Schlangen, 2004).

2.3 Statistical corpus studies

2.3.1 Annotation protocols and reliability

Before discussing the results and analyses of statistically-driven corpus studies, it is worth briefly discussing the methods by which corpora can be annotated for self-repair, either by human annotators or by automatic means, and the issue of reliability and agreement.

Healey et al. (2005) present a systematic effort to test the reliability of a human annotation scheme for repair, developing Healey and Thirlwell (2002)'s annotation protocol for identifying the different CA types of repair in raw dialogue transcripts (see Figure 2.1 with the self-repair types circled). The authors tested the validity and reliability of the protocol through an analysis of two of the authors coding a corpus of repair sequences drawn from the CA repair literature with their original coding removed. The validity of the protocol was shown to be encouraging overall, with 75% of the repairs being assigned the same category as that of the original papers.

After analysing the breakdown of their target phenomena, it appears that the rate of successful identification was in fact identical for the self-repair phenomena (P1-SI-SR(first position self-repair), P1-SI-SR(transition space), P1 fail, P3-SI-SR (third turn), P3-OI-SR(third position)) as it was for the other-repair phenomena (P2-OI-OR, P2-NTRI, P2-NTRI fail) at 75% for both sets. The within-turn self-repairs (P1-S1-SR) had the highest identification out of the sub-categories at 88% and the lowest were P1 failures (50%).

Despite the identical identification rate overall of self and other repair, there was a difference in the reliability measures (inter-annotator agreement scores) between self-repair and other-repair as this was 63% and 73% respectively- the authors report that most frequent mismatch was between events annotated as P1 formulation problems and P1 articulation problems, and that the levels of agreement were primarily reduced in situations where one judge identified a repair event where the other did not, rather than differing in their classification judgements- this suggests that the protocol itself is fairly precise should the repair be identified. Within self-repairs, P1 (63%) and P3 (62%) had almost identical agreement levels. Interestingly the identification of repairs had greater agreement (63%) than repair initiators (50%).

The results suggest the potential of a rigorously defined repair protocol for use on raw transcripts but with the caveat that while the classification once a repair event is identified can be quite reliable (the high classification rate for P1-SI-SR, the first position repair being particularly encouraging), the difficulty in getting a reliably agreed upon protocol for initial identification of a repair phenomena persists.

Annotating causes of self-repair may be problematic, as suggested by the error in distinguishing between articulation and formulation P1 repairs described above, and by the fact that annotators cannot ascertain from form alone what the exact nature of the underlying problem is, and even with CA-level of contextual analysis, this can be difficult. However annotation for first position self-repairs in transcripts can be systematized more straightforwardly, by marking disfluent parts of utterances in the transcript (Shriberg, 1994; Meteer et al., 1995) according to the annotation scheme in (2.4), and use of this coding scheme will be described in the work below.

2.3.2 Statistical distribution of self-repair in the repair space

The early CA analyses of repair identified the various types of repair according to their position in dialogue and their initiator, but did not include an accurate analysis of their frequency of occurrence in human conversation. Colman and Healey (2011) provide a statistical distribution

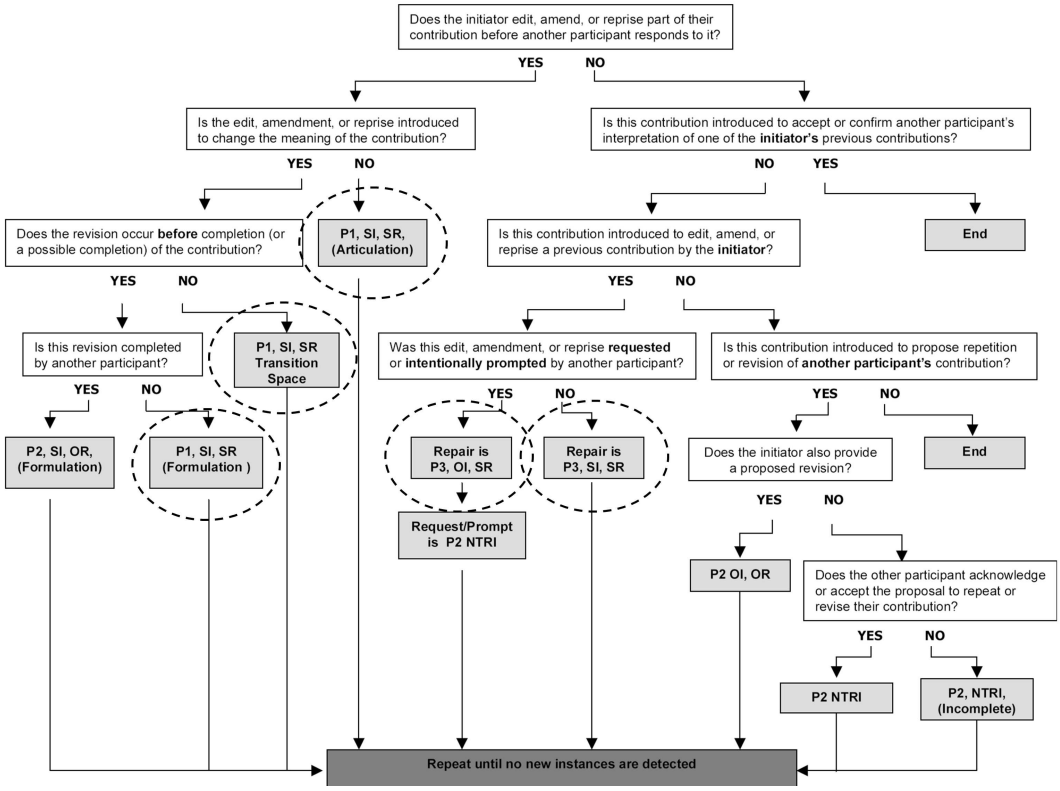


Figure 2.1: Repair Identification Protocol from Healey et al. (2005).

of different types of self-repair in a comparative corpus study of normal (non-task orientated) dialogues from the British National Corpus (BNC, Burnard, 2000) and task-oriented dialogues from the HCRC Map Task corpus (Anderson et al., 1991). They devise an annotation based on Healey et al. (2005)'s repair protocol explained above, the gloss for which can be seen in Table 2.1, with examples of the repair types in Table 2.2.

GLOSS	**REPAIR PROTOCOL CATEGORY**
Repeat	Position 1 Self-Initiated Self-Repair 'Articulation'
Restart	Position 1 Self-Initiated Self-Repair 'Formulation'
Transition	Position 1 Self-Initiated Self-Repair in Transition Space
Clarification Request (CR)	Position 2 Next Repair Initiator (NTRI)
Correction	Position 2 Other-Initiated, Other-Repair
Follow-up	Position 3 Other-Initiated, Self-Repair
Reformulate	Position 3 Self-Initiated Self-Repair

Table 2.1: Gloss for repair annotation in Colman and Healey (2011).

GLOSS	**EXAMPLE**
Repeat	Follower: which is due we– *due west*?
Restart	Giver: so you're underneath them. . . *between them?*
Transition	Giver: and, start going down Southeast. . . . . *you go past a pine forest on your right*
Clarification Request (CR)	Giver: past a forge on your right? Follower: *past a what?*
Correction	Giver: right to the end of. . . paper Follower: *the very end of the map?*
Follow-up	Follower: so you want me to go. . . east. . . then south? Giver: *no, south then east we may have a different map*
Reformulate	Giver:right. . . now,have you got the hot wells? Follower: they're over a bit Giver: *or hot springs?*

Table 2.2: Examples for annotation scheme from Colman and Healey (2011).
Example repairs in *italics*

The distribution they found in natural conversation in the BNC dialogues confirmed the CA observation for the preference of speakers for self-repair over other-repair (see Figure 2.2). In terms of the overall frequency of repair events however they found a difference between corpora: repair occurred once every 36 words, or every 5 turns, in natural conversation (BNC), compared to once every 20 words, or every 2.5 turns, in task-oriented dialogue (Map Task). This suggests that task difficulty has a systematic effect on how likely speakers are to repair. Significant statistical differences were also found in the distributions of repair– *restarts* are much more prevalent in task-oriented dialogues (Map Task) than in free conversation (BNC) (see the difference in distributions in the bar charts in Figures 2.2 and 2.3), which suggests a preference for making utterances optimally clear through revising previous contributions in task-oriented domains. The transactional nature of map-following and instruction giving puts a higher demand on clarity from speakers in the 'route-giving' instructor role, and this skewing between roles was shown to be significant, with route-givers exhibiting significantly more restarts, reformulations, transition space repairs and follow-ups (i.e. self-repairs) in their speech and clarification requests and corrections (i.e. other repairs) being significantly more frequent in the follower's speech. What is striking here is support shared between CA analysis and quantitative analysis for the fact that repair is as *local* to the trouble source as possible in the course of a dialogue, and that its frequency is influenced by the domain. The role of self-repair in interaction will be discussed in more detail

below in Section 2.4.2.

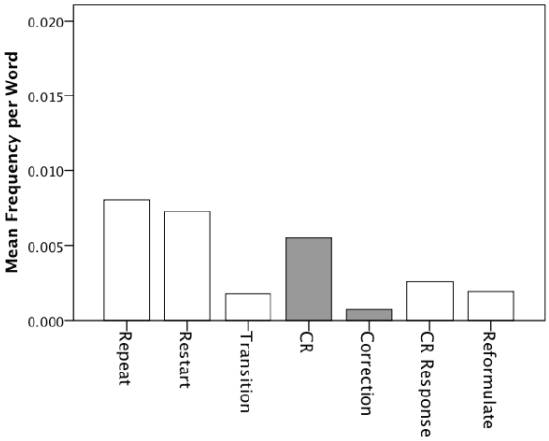


Figure 2.2: Repair Distribution in Ordinary Conversations (BNC) (Colman and Healey, 2011). Light Bars = 'Self', Grey Bars = 'Other'

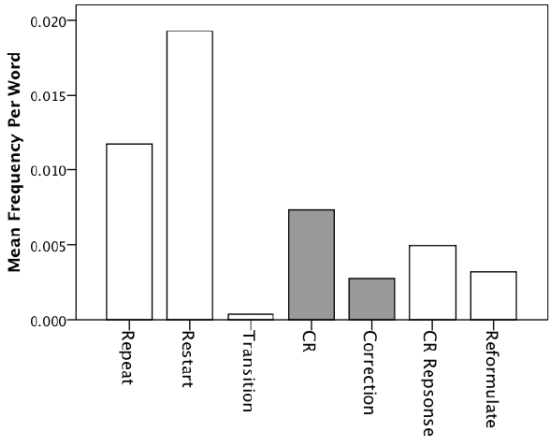


Figure 2.3: Repair Distribution in Task-Oriented Dialogues (Map Task) (Colman and Healey, 2011). Light Bars = 'Self', Grey Bars = 'Other'

2.3.3 Types of self-repair and disfluency surface forms in corpora

Moving to a within-utterance level of analysis of self-repairs in particular, Shriberg undertook the first major computational corpus work on disfluencies in her thesis (Shriberg, 1994) and in subsequent papers (Shriberg, 1996; Shriberg et al., 1998, *inter alia*), with the intention of improving speech processing in automatic systems.

Shriberg (1994)'s thesis provided a systematic way of characterizing the different types of repair and disfluency forms. She also provided a robust annotation protocol for disfluency types which could be semi-automated. She identified types by the relationship between the *reparandum* and *repair* components of the speech repair (see Section 2.1), as can be seen in Tables 2.3 and 2.4.

Shriberg provided a reliable decision protocol for coding disfluency, in the bracketing scheme described above in (2.4), differentiating between the different surface forms of speech repairs and other disfluencies, and defining the different classes through intersecting the presence of different members of the annotation symbols $\{\sim, f, s, d, r, c, i\}$ assigned to words and partial words with their meanings as in Table 2.3 giving the classification as shown in Table 2.4. The classes were: filled pause (FP), articulation disfluency (ART), hybrid disfluency (HYB), substitution (SUB), insertion (INS), deletion (DEL), repetition (REP) and conjunction (CON), the last class occurring between speaker sentences. The reliability of this protocol meant it could be scalable and semi-automated.

2.3.4 Distribution of repairs and their relation to other disfluencies

The overall distribution of these repairs across three different dialogue domains can be seen in Figure 2.4, where the most common type in all three domains is the filled pause (FP) and the X-axis reads off the rank of frequency across all three domains from left-right. It was observed that in terms of interaction with position of the interruption point, per-word rates by position showed that the three most common disfluency types (FP, REP, and DEL) were much more likely to occur in initial position than in medial position. The remaining types appear to be roughly equally likely in initial and medial positions. This suggests an interaction between the self-repair strategy employed and the length of their utterance so far, an issue that will be addressed in Chapter 4.

2.3.5 Characterizing retrace lengths with statistical modelling

While work had been done on characterizing the forms of self-correction in terms of the permissible syntactic points of a sentence from which a speaker could retrace (Levelt, 1983, 1989), these accounts did not address the *probability* of the retrace occurring from each of these possible points or the probable length of how much of the utterance was to be retraced given these positions. To address this, Shriberg and Stolcke (1998)'s corpus study developed a quantitative model of repetition-type self-repairs that was purely word position based to attack the central question: *when speakers retrace, what predicts how far back they go?* (Shriberg and Stolcke,

Symbol	Explanation	Example	Section
Region-delimiting			
[ ]	onset RM, offset RR	(see all examples below)	4.3.4.1.1
.	IP	(see all examples below)	4.3.4.1.2
Syntactic-word			
r	repeated word	she she liked it [r . r]	4.3.4.2.1
s	word in substituted string	she my wife liked it [s . s s]	4.3.4.2.2
i	inserted word	she liked really liked it [r . i r]	4.3.4.2.3
d	deleted word	it was very she liked it [d d d .]	4.3.4.2.4
Extra-syntactic-word			
f	filled pause	she uh liked it / she uh he liked it [.f] [s . f s]	4.3.4.3.1
e	explicit editing term	she sorry he liked it [s . e s]	4.3.4.3.2
p	discourse marker	she liked well she liked it [r r . p r r]	4.3.4.3.3
Inter-sentence-word			
c	coordinating conjunction	she saw it and and she liked it [c . c]	4.3.4.4.1
Diacritics			
-	word fragment	she li- he liked it [s r- . s r]	4.3.4.5.1
~	misarticulated word	shle she liked it [r~ . r]	4.3.4.5.2
^	contracted word	she'd she'll like it [r^s . r^s]	4.3.4.5.3
=	substituted-string fragment	she thought highly she liked it [r s s= . r s]	4.3.4.5.4

Table 2.3: Tag set for type of disfluent words (Shriberg, 1994, p. 57)

1998, p.2183).

From 1115 conversations from the Switchboard disfluency-tagged corpus that had been marked

TYPE	Must Include	Must Not Include	May Include	Examples
ART	~		s i d r c f	shle she liked it [r~.r] i'd like to fry uh to fly from boston [rr~.fr show grand trou- ground transport [r~r~-.r
HYB	2^+ from: s i d	~	r c f	(example of s + i): she liked he really liked it [sr.sir]
SUB	s	~ i d	r c f	she he liked it [s.s] she uh he liked it [s.fs] and she liked and he liked it [csr.csr]
INS	i	~ s d	r c f	she liked really liked it [r.ir] she liked uh really liked it [r.fir] and she liked and she really liked it [crr.cri
DEL	d	~ s i	r c f	it was she liked it [dd.] it was um she liked it [dd.f] and it was and she liked it [cdd.c]
REP	r	~ s i d	c f	she she liked it [r.r] she liked uh she liked it [rr.frr] and she and she liked it [cr.cr]
CON	c	~ s i d r	f	she saw it and and she liked it [c.c] she saw it and uh and she liked it [c.fc]

Table 2.4: Disfluency type annotation scheme for dialogue (Shriberg, 1994, p. 78)

for sentence boundaries and disfluencies, they harvested 30,524 disfluent utterances containing one or more retraced words. In gathering their data, they first observed that retracing did not occur across sentence boundaries nor go back to mid-word points of a previously uttered word. They characterized retracing in terms of the number of retraced words in the sentence *k* (i.e. the length of the reparandum, from [up to the +), and the number of words in the utterance before

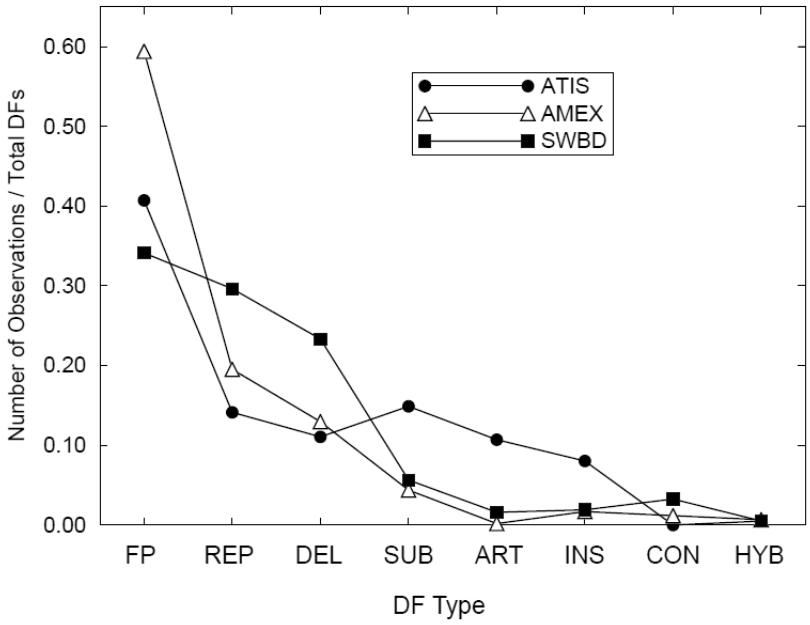


Figure 2.4: Distribution of disfluency types (Shriberg, 1994)

the repair was initiated m (that is the number of words from the beginning of the utterance before the repair point $+$). Their word-based measures can be seen in Figure 2.5.

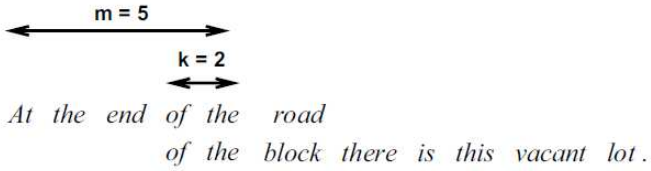


Figure 2.5: Word-based measures used in Shriberg and Stolcke (1998)'s study.

Their central finding is the apparent exponential decay in frequency in k with increasing values of m: in other words, speakers are much more likely to trace back one word than they are two words, and so on. They also identified the exception to this trend for all values of m in the boosted frequency of retraces spanning the entire length of the utterance so far, i.e. when $k = m$. See the final "hooks" of the lines in Figure 2.6 for values of $m \leq 6$.

There is also an exponential decay in this boosted probability value of $k = m$ retraces, in that the skip back is exponentially less likely to happen when the utterance is one word further on; for example when a speaker is only two words into a sentence the extra probability that they will skip back to the start is around 70%, whereas four words into a sentence this is only around 5%

(Shriberg and Stolcke, 1998, p.2185).

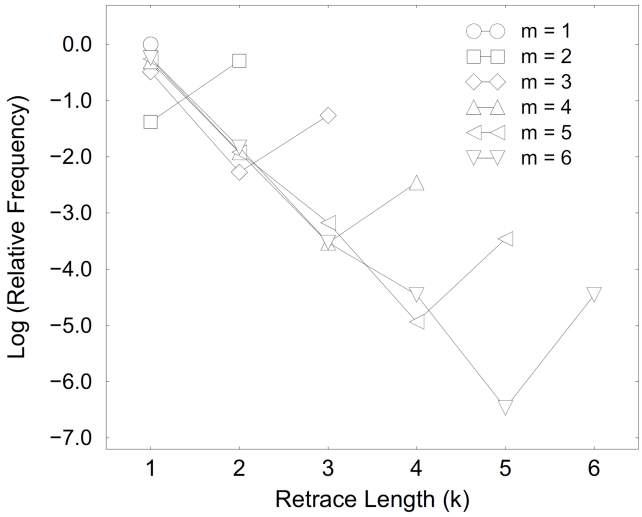


Figure 2.6: Distribution of Retrace Lengths by Position. From (Shriberg and Stolcke, 1998)

Additionally, their results clearly show the uniform relationship between the probability of retracing from N words back and that of retracing from N+1 words back- this shown by the straightness of the lines in Figure (2.6) which indicate a proportionally decreasing logarithmic frequency of occurrence for increasing values of *k*. It is suggested by the authors that the reason for this uniformity is due not only to constituency boundary reasons where it is more effortful to restart larger constituents[3] but also for processing and *temporal* factors: they claim that if speakers are optimizing speaker time rather than silence time, they retrace more when they need more time to reformulate.

The other interesting aspect of their model is that, except for the boosted probability of retracing the entire length of the sentence, the likelihood of a retrace of a given length *k* is equally likely regardless of the number of words uttered in the sentence so far *m* (the distance from the beginning of the utterance to the repair point) - this can be seen graphically by the close proximity of the frequency points for each value of *k* in all the *m*-value lines, except when $k = m$. The authors do not offer a complete explanation for this, but this constitutes clear evidence that speakers show a tendency to retrace as *locally* to the trouble source as possible. Eklund (2004)'s thesis drew similar conclusions about verbatim retracing and its regularity in that the maximum retrace length observed among several corpora was 4.

[3]It is worth bearing in mind the observation by Levelt (1989) that in English, a predominately syntactically right-branching language, under hierarchical phrase-structure type syntactic analyses every word typically marks the onset of some constituent.

The authors concede that the model requires some syntactic constituency considerations to get greater coverage. They show this through conducting Monte Carlo experiments on part-of-speech (POS) tagged versions of all the retraces in the corpus used for their model. Taking the value of *m* for each sentence they generate a *simulated retrace* by selecting a random value for *k* associated with the *m* value in question, taken from the empirical distribution from the data, and taking the reparandum as beginning word *k*-words back. In their comparison, they found that for the majority of POS types the simulation produced close to the frequency obtained from the real data. However, they mention that the word position based distribution under-determined (failed to predict) the frequency of *prepositions* being retraced, with speakers showing a preference for retracing from preposition onsets above the quantitative prediction (which they report as still being high, probably due to the fact that prepositions frequently begin phrases of only 2 words, but this was still not high enough). The authors also interestingly report that the model over-predicted for verbs, claiming a weighted preference of speakers not to retrace from verb phrase onset boundaries.

In summary, the purely quantitative model they proposed predicts the retrace points with measurable success without any notion of syntactic constituency. The model, while simple, places certain constraints on any generative model of retracing as a word position only based analysis of data gives some interesting regularities that could not be gleaned by analysing syntactic category alone.

2.3.6 Distribution of discourse markers in interregna

Heeman and Allen (1999)'s model for automatic disfluency detection and correction will be discussed below in Section 3.1 for its efficacy in application, but it is worth describing the useful empirical data on self-repairs found in their task-oriented TRAINS corpus (Heeman and Allen, 1995). In particular, they present corpus statistics on the presence of discourse markers such as "well" and "actually" and the distribution of filled pauses in the editing terms of repairs. They provided useful evidence about repair strategies (and consequently were used for improving a model of repair detection and correction, as will be described).

The role of discourse markers for predicting the presence and type repair was explored, with the statistics in Table 2.5 demonstrating that discourse markers were part of the interregnum or the onset of the repair for 40% of fresh starts, 13.8% of modification repairs and 10% of abridged repairs, with a significantly higher occurrence in the onset of modification repairs. This suggests

	Abridged Repair	Modification	Fresh starts
Number of repairs	423	1301	671
DM in editing term	36	60	155
DM in alteration onset	8	126	147
Either	41	179	269

Table 2.5: Frequency of discourse markers in the editing term of speech repairs and as the alteration onset in Heeman and Allen (1999)'s Trains corpus

that while on its own the presence of a discourse marker cannot be strongly predictive of a given repair type, there is a difference in the distribution of the types of repair form which contain a discourse marker.

2.4 Psycholinguistic approaches

This section summarizes a number of approaches to experimentally finding the causes and communicative effects of producing and comprehending self-repairs.

2.4.1 Causes and syntactic form of first position self-repair

Levelt (1983)'s study aimed to systematically identify the form and causes of self-repair in a rule-based manner. In a corpus study Levelt demonstrated the frequent parallelism between the reparandum and the repair, and the syntactic and semantic felicity of self-repairs. He showed the listener faces the *continuation problem* of how to integrate the material from the repair phase into the previous material.

To address this he developed the "well-formedness" rule (WFR), which stated that for an original utterance (O) plus repair (R), {OR} is well formed if and only if there is a string C such that the string {OC or R} is well-formed, where C is a completion of the constituent directly dominating the last element of O. For example for "Is [the nurse + {er} the doctor] interviewing patients?" the coordination test would apply and give "Is the nurse or the doctor interviewing patients?", giving a well-formed repair. He claimed this rule applied to all but a minority of the 957 repairs in his corpus.

In computational linguistics Hale et al. (2006) implemented the WFR in a disfluency detection system, marginally altering the rule as follows :

(2.18) "**Well-formedness rule for repairs (WFR)** A repair $\langle \alpha\gamma \rangle$ is well-formed if and only if

> there is a string β such that the string $\langle \alpha\beta$ and* $\gamma\rangle$is well-formed, where β is a completion of the constituent directly dominating the last element of α (*and* is to be deleted if that last element is itself a sentence connective)" (Hale et al., 2006, p. 163)

While their results on reparandum word detection were not competitive with other systems that will be described in Chapter 3, Hale et al. (2006) showed the inclusion of the WFR to their system improved their results.

A challenge to the WFR came from van Wijk and Kempen (1987), who investigated the rule empirically, conducting self-repair eliciting experiments. They concluded that there must be two strategies at work for computing the shape of self-repairs, one called *reformulation* and the other called *lemma substitution*. They argued that the WFR only applied for reformulation, whereas for lemma-substitution, there were more complicated processes at work involving a prosodic unit called the phonological phrase.

In terms of identifying causes, Levelt (1989) indicated that *self-monitoring* was the key to self-initiated self-repairs. These causes can be broken down, informally, into 7 self-posed questions that a speaker may ask themselves during speech production in conversation (Levelt, 1989, pp.460-462):

- Is this the message/concept I want to express now?
- Is this the way I want to say it?
- Is what I'm saying up to social standards?
- Am I making a lexical error?
- Are my syntax and morphology all right?
- Am I making a sound form error?
- Has my articulation the right speed, loudness, precision, fluency?

Central to all of these questions is the notion of a feedback loop from one's own speech signal to one's speech comprehension system, suggesting a speaker does not have full control over their articulation and formulation, requiring instead continual detection of trouble during speech production – see Figure 2.7. While the nature of the trouble encountered ranges from the articulation concerns up to high-level concerns of social propriety, the hearer of the utterance is not considered in much detail in Levelt's account, which given the nature of the questions is surprising. Having said this, the notion of self-monitoring gave the model of speech production a

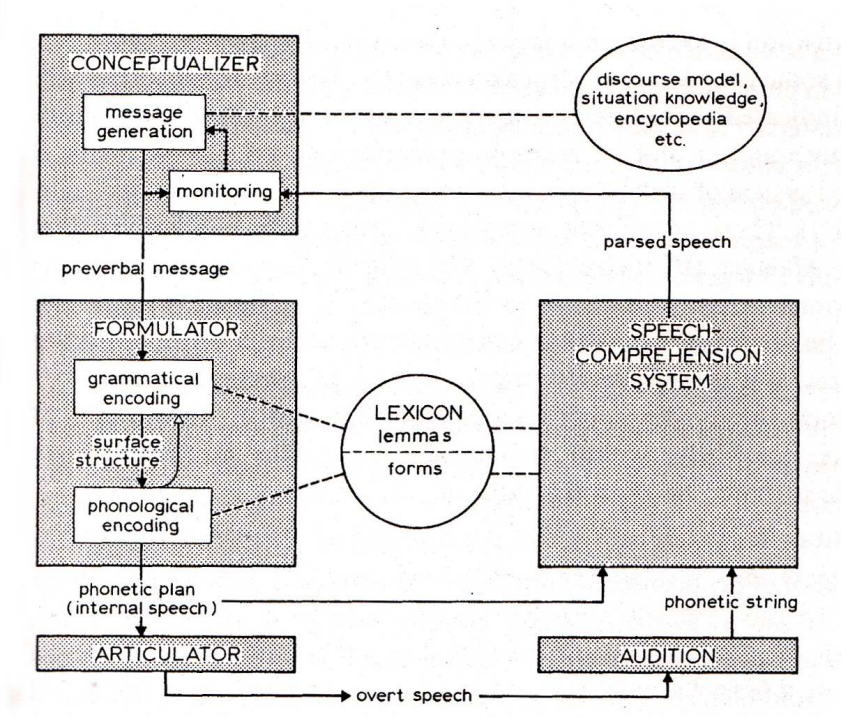


Figure 2.7: Levelt's model of the speaker with the internal feedback loop.(Levelt, 1989, p.9)

causal explanation for self-repair and has become an important concept for psycholinguistic experimentation and computational modelling (Guhe, 2007; Skantze and Hjalmarsson, 2010). The tri-partite model of *conceptualisation*, *formulation* and *articulation* also gave a neat functional decomposition of the speech production process and computational generation– I will return to this in the next chapter.

2.4.2 The role of self-repair in interaction

In contrast to the focus on individual language use in the autonomous transmission account of psychology, a more interactive stance can be taken towards self-repair that holds interesting insights. Clark (1996) claimed conversation participants manage the production and interpretation of communicative acts with the least *collaborative* effort, (that is the smallest aggregate effort of the speaker and hearer), suggesting self-repair is sometimes the optimal strategy for managing interactive situations. While the speaker may incur some processing cost in repairing the utterance, the predicted overall reduction of communicative effort is beneficial enough to make it worthwhile, concordant with the premise that "utterances are truly products of speakers and

addressees acting jointly" (ibid., p.286). Clark points out that taking the interactive approach, the distinction between self and other repair made explicit in Schegloff et al. (1977) begins to disappear, as the source of trouble in a dialogue may be difficult to attribute to a particular participant, as the failure to comprehend an utterance cannot be determined simply from failure of the speaker of the utterance to make himself clear, but from interactive trouble sources, or *grounding* trouble.

For first position self-repair, in line with Levelt and Shriberg's work, Clark (1996) describes a three interval model, consisting of an *original delivery* (utterance up to the repair point), *haitus* (interregnum) and *resumed delivery* (utterance from beginning of the repair onwards). While the intervals apply to an individual dialogue participant's utterances, the phases are discussed in terms of their role in interaction: the hiatus signals a problem to the hearer, often explicitly marked as a filled pause form of 'um' and 'uhh', which Clark and Fox Tree (2002) consider official words in English because of their communicative function, and which can signal the downgrading part of the original delivery (reparandum). In line with his views on meta-communicative signals in general, Clark's central claim is that repairs produce informative signals about the utterances themselves.

Clark distinguishes four types of resumption to categorize different repair types:

1. *Instant replacements* e.g. 'have I ever [tel- {} + talked] to you about Cookstown County Tyrone?'
2. *Trailing replacements* e.g. 'if [she'd been {} + he'd been] alive'
3. *Anticipatory replacements* e.g. 'he [thinks E-{} + thinks Ella's] worried about something'
4. *Fresh starts* e.g. [we must ha-{} +] we're{.} big enough to stand on our own feet now'

While these categories seem similar to the surface-level characterization in Shriberg (1996)'s taxonomy, Clark's explanation of the phenomena is more communication oriented. One thing he makes clear is that self-repairs can be treated as communicative signals and that a characterization of self-repair should not involve the removal of previously committed material from the common ground of the dialogue participants. He stresses that replacement does not mean obliteration of the original utterance, as the following simple example demonstrates:

(2.19) " [the interview, was {...} it was] all right."

(Clark, 1996, p.266)[4]

This illustrates that while replacement would mean "it was all right" can be interpreted as the new presentation of the utterance (i.e. the repair), the pronoun "it" is the anaphoric reference to the antecedent "the interview", so the reparandum cannot be removed from short-term memory nor be inaccessible syntactically or semantically. Not only is the repaired material accessible, it is crucial for meaning construction.

Ferreira et al. (2004)'s experiments investigate the lingering effect of the reparandum in more detail, and show that experimental subjects over 40% of the time will judge ungrammatical repairs such as "Simon says you should [drop + {uh} put] the frog" as grammatical, due to the effect of having processed the argument structure of 'drop' before 'put' is encountered. Clearly a repair's effect on incremental linguistic processing is more complex than it would initially seem.

2.4.3 Use of repair processing in comprehension

Brennan and Schober (2001) investigate the effect of disfluent instructions on participants in a simple image selection experiment, showing results which back up Clark (1996)'s idea that the identification of a repair itself has positive effects on a hearer's comprehension due to paralinguistic cues that the reparandum is a trouble source. They experiment with three types of repaired instructions: mid-word interruptions (e.g. "Move to the yel-purple square"), mid-word interruptions with a filler (e.g. "Move to the yel- uh, purple square") and between-word interruptions with no filler (e.g. "Move to the yellow- purple square"). Equivalent fluent instructions (e.g. "the purple square") and disfluency-excised instructions that replace the reparandum and interregnum with pauses of equal length (e.g. "the .. purple square") were used for comparison and to isolate the effect of the repair processing and to account for different prosodic quality of the onset of the repair phase (as their preliminary off-line study found the pitch to be higher than the reparandum word onset).

Each of the instruction types were played to subjects who had to select the target object from a visual scene of 2 objects in three of the experiments, and from 3 objects in the remaining one. To mitigate the time advantage of repaired utterances and repair-excised utterances versus naturally fluent ones, reaction times were measured relative to the onset of the word describing

[4]Core and Schubert (1999) give a similar example with "have the engine take the oranges to Elmira, um, I mean, take them to Corning", where semantic processing access to "the oranges" in the reparandum is required to resolve the meaning of "them".

the target referent (i.e. the beginning of "purple" when the target was the purple square). The accuracy of the subjects' first object selection was also calculated.

Across all four experiments the speed of response and accuracy of subjects in selecting the correct object followed a similar pattern within the disfluent instruction conditions: the fastest and most accurate responses came upon hearing mid-word interruptions with a filler ("..the yel- uh, purple square"), the next most accurate came in the between words without fillers condition ("..the yel-, purple square") and the least accurate performance came in the between-words condition ("..the yellow, purple square"); there was no difference in speed between the last two conditions. Perhaps the most important finding from all experiments was that the mid-word interruptions with filler condition resulted in faster responses from the onset of the target word than the equivalent *fluent* instruction, with no significant loss in accuracy. The other two disfluency conditions resulted in less accurate results than the fluent condition, and faster response times in two of the experiments where there was a time limit.

When they investigate repetition repairs in the last experiment (e.g. "Move to the yell- yellow square") these again yielded faster response times than fluent utterances but with a lower error rate than the replacement repairs with no fillers and between-words condition, however with slower responses and no more accuracy than the mid-word interruptions with fillers condition, which was consistently the fastest and jointly most accurate condition across all experiments.

The authors suggest the increased reaction time from the target onset word in the mid-word interruptions with fillers condition setting could have several causes: use of information that the interrupted word signals an intention to revoke the object choice, the phonetic form of the filler word, the time delay permitted by the filler word to compute the repair, the contrast in semantics or stress of the repair word, or any combination of these factors.

To factor out the possible effect of the phonetic form of the filled pause 'uh', they introduce filler-excised versions of the repaired instructions ("..the yel- .. purple square"), but with the repair onset occurring at the same time as the original instruction. This resulted in the same level of performance from subjects as the filler condition. This suggests it was not the form of the filler but the time allowed to compute that a repair has occurred that helped subjects– note when the repaired instructions have no filler or pause for an interregnum subjects were less accurate. The fact that the between-word condition lead to slower and less accurate responses suggested the indication of repair is weaker without the combination of the partial word and filler– subjects

cannot draw the information that their (virtual) instructor intended to replace the reparandum as readily.

Brennan and Schober (2001)'s experiment confirmed the use of incremental information processing in self-repair comprehension within two simple domains: one where the subject had to pick a target referent from two objects of the same shape but with different colours, and one where the subject picked from three objects, again with the varying property being colour. The effect of the 'disfluency advantage' in reaction time was attenuated in the three-object condition, however it was still positive for the mid-word interruption with filler condition. I show a summary of the experimental conditions and instruction types in Figure 2.8 and the experimental results in Table 2.9 in terms of speed and accuracy rank based on the significant interactions reported.

From a semantic processing point of view, one could conclude that in the two-object condition, subjects factor out one of two objects in computing the reparandum referent incrementally, allowing them to make the simple binary decision more quickly, whereas in the more complex domain the information content from the repair is lessened– while it helped factor out one of the objects a choice remained between the other two.

In addition to confirming the hypothesis that hearers compensate for a disfluency by computing the meaning of the repair faster relative to its onset than in an equivalent fluent utterance, the authors conclude that the less misleading the reparandum is (where a full incorrect colour word is the most misleading, followed by a partial incorrect colour word, with partial repeat repairs not being misleading at all) the more accurate hearers are in computing the meaning of the repair (or in this simple domain, resolving the referent).

2.4.4 Incremental elicitation and effect of self-repair in dialogue

As for experimental work on self-repair in dialogue domains (rather than in non-interactive instruction following), Healey et al. (2011) explored incremental processing in text-based dialogue through the DiET chat-tool methodology (Healey et al., 2003), which permits the real-time manipulation of dialogue turns during character-by-character chatroom-like conversations. The experiment elicited third position other-initiated self-repairs or *follow-up*s (Colman and Healey, 2011) from participants through the insertion of "spoof" clarification requests (CRs). The CRs were automatically generated by the system during a participant's typed turn and took the form of a repeated noun phrase (NP) that had just been typed in the turn with an additional question mark,

Type of instruction	example
Mid-word interruption w. filler	“Move to the yel- uh, purple square”
Mid-word interruption	“Move to the yel-purple square”
Between-word interruption	“Move to the yellow- purple square”
Mid-word interruption (repeat)	“Move to the yel-yellow square”
Filler+reparandum replaced w. pause	“Move to the ... purple square”
Filler replaced w. pause	“Move to the yel- .. purple square”
Reparandum replaced w. pause	“Move to the .. uh, purple square”
Fluent	“Move to the purple square”

Conditions:

Exp. 1:	2 objects.
Exp. 2:	3 objects.
Exp. 3:	2 objects. Time limit. Filler replaced by pause, reparandum replaced by pause conditions included
Exp. 4:	2 objects. Time limit. Same as Exp. 3 but with repetition repair condition included.

Figure 2.8: Instruction fluency conditions and experimental conditions in Brennan and Schober (2001)

	Speed (accuracy) rank			
Type of instruction	Exp. 1	Exp. 2	Exp. 3	Exp. 4
Mid-word interruption w. filler	1 (1=)	1 (1=)	1= (1=)	1= (1=)
Mid-word interruption	2= (4)	4= (4)	3= (6)	3= (7)
Between-word interruption	2= (5)	4= (5)	3= (7)	3= (8)
Mid-word interruption (repeat)	x	x	x	5 (1=)
Filler replaced w. pause	x	x	1= (1=)	1= (1=)
Reparandum replaced w. pause	x	x	5= (1=)	6= (1=)
Filler+reparandum replaced w. pause	4 (1=)	2 (1=)	5= (1=)	6= (1=)
Fluent	5 (1=)	3 (1=)	7 (1=)	8 (1=)

Figure 2.9: Summary of Brennan and Schober (2001)’s experimental results

appearing on the screen of the targeted dialogue participant to originate from another dialogue participant (e.g. B's fake turn on A's screen in `A: John went to town. B: John?`). There was a within-subjects manipulation of *insertion point* in that probe CRs were inserted either after a constituent boundary (e.g. after a completed NP) or within an incomplete constituent (e.g. after a determiner). There was also a between-subjects manipulation of the *origin* of the CR, in that "fake" turns in one condition were displayed on the screen as if originating from the participant's dialogue partner and in the other condition made to appear, as explained prior to the experiment to participants, as if generated by an artificial dialogue agent (chatbot).

The results in terms of *response time* showed no reliable effect between the different insertion points, however participants were quicker to respond to the chatbot's CRs. In terms of *reformulation*- that is whether the target word in the CR was paraphrased but semantically equivalent- again there was no reliable effect of insertion point, however responses to the chatbot were more likely to reformulate the target. The third comparison, which was the likelihood of a *restart* showed a reliable difference with respect to insertion point, as probe CRs inserted within an incomplete constituent were more likely to elicit restarts of from the beginning of the typed turn in comparison to the CRs at constituent boundaries. An interesting result for Human-Computer Interaction (HCI) was also found in this comparison in that participants were more likely to restart in response to a ratified dialogue partner than in response to the a chatbot's probe.

2.4.5 The effect on self-repair of the status of dialogue participants

Healey et al. (2011)'s chatbot results give an insight into the difference between interaction with side participants and interaction with primary addressees, a finding further corroborated by Eshghi (2009). Additionally, viewed from an HCI perspective, the responses to the chatbot provide an interesting parallel to the statistical corpus comparison of disfluencies in the Switchboard (SWBD) and AMEX travel agency human-human dialogues and the Air Travel Information System (ATIS) human-computer dialogues in (Shriberg, 1996). Shriberg's results showed that a human participant addressing a computational dialogue system (in the ATIS corpus) is not only far more fluent than in normal human-human conversation, but more interestingly, that this difference can be attributed to the three of Shriberg's repair types which are not classified as articulation errors– filled pauses, repetitions, and deletions. The results suggested people were less likely to insert floor-holders and less likely to repeat in speaking to a computerized dialogue system than to a human dialogue partner, which are observations consistent with the chatbot

interaction findings in Healey et al. (2011).

In both of these cases there is a systematically different behaviour in terms of orientating to artificiality, with more reformulation (and hence less openly disfluent utterances) in the chatbot condition in Healey et al.'s experiment and conversely a significant tendency to speak fluently to the system in the ATIS dialogues.

The results suggest the causes of different classes of self-repair can be attributed to interactional factors and the status of the hearer, rather than being solely due to processing demands or perception that is external to a dialogue. There is still the question of domain-specificity, and whether a more open chat-like conversation would yield different results. It is clear this experimental paradigm could be used to elicit different types of self-repair from human participants.

2.4.6 The role of self-repair in psychiatry and mental state attribution

In terms of practical use, analysing the distribution of self-repair rates has been found to be effective in analysing doctor-patient interactions in psychiatric therapy sessions and their effectiveness. Howes et al. (2012) showed that in psychiatric therapy situations between schizophrenic patients and doctors, participants are much more likely to self-repair than in normal dialogues, and do less other-repair.

Furthermore, McCabe et al. (2013) showed how self-repair rate correlated with treatment adherence for schizophrenic patients, in that those who did less self-repair but more other-repair (clarification) were more likely to adhere to treatment. Self-repair was shown to correlate with *negative symptoms* of schizophrenia, such as the 'flat effect' of displaying little interest or emotion in communication, a finding that backed up early observations of this correlation by Leudar et al. (1992).

Additionally, the degree to which expectation about mental states has effects on how hearer's interpret self-repair incrementally was found by Arnold et al. (2007). They set up an experiment similar to Brennan and Schober (2001)'s object selection task as described above, presenting hearers with fluent ("Click on the red…") and disfluent instructions with an elongated determiner and filled pause ("Click on [pause] thee uh red…), being presented with normal everyday objects and difficult-to-name novel objects. They found the disfluent instructions made novel objects more expected, as shown through eye-gaze measurements indicating on-line hypotheses of referents from the onset of the color word. However, the novelty bias was sharply attenuated when the listener was told the speaker had object agnosia (i.e. poor ability to recognize objects),

presumably due to the inference from the hearer that the speaker would have difficulty naming familiar objects, not just rare ones. This shows the interactive effect of repair and the fact it is used to make inferences about an interlocutor's mental state, rather than being ignored and filtered out as noise in a speech signal.

2.5 Summary and directions for research

Given the evidence of the pervasiveness of self-repair in dialogue and the potential for various analytical uses, it is clearly a phenomenon worthy of study. Any model of dialogue must account for it to have any cognitive plausibility. Ginzburg et al. (2014) make the analysis of disfluency to friction in physics– while an idealised model can do without it, one that pertains to model empirical evidence cannot.

Generally, there seem to be two stances in the literature on empirical approaches to self-repair as regards its psychological status, one being very interactive (CA; Clark, 1996; Healey et al., 2011) and the other more autonomous and processing focussed (Levelt, 1989; van Wijk and Kempen, 1987). However, in both cases, there is a principle of locality in self-repair. Clark summarizes this in his *principle of repair*: "When agents detect a problem serious enough to warrant a repair, they try to initiate and repair the problem at the first opportunity after detecting it"(Clark, 1996, p.284).

The statistics from Shriberg and Stolcke (1998)'s study along with Schegloff et al. (1977)'s observation of preference for self-initiated same-turn self-repair indicates optimal repair strategies are geared towards the most immediate reaction possible to a trouble source, with the slight hook in the trend for entire utterance restarts indicating the benefit of complete reformulation/starting 'afresh'. From the micro-level of sound form and word position to the larger intervals of dialogue contribution it becomes clear that self-repair takes place as quickly as possible.

Additionally, the re-use of structure as shown by Levelt's well-formedness rule shows a principle of least effort: consistent forms of self-repair make interpretation easier for a hearer. While this rule may not cover all cases, it certainly has some regularity. Also, there is empirical evidence that addressees use the information in reparanda and filled pauses to their processing advantage in reference domains (Brennan and Schober, 2001; Arnold et al., 2007) and that it is indeed necessary to retain the reparandum to compute the meaning of an expression for anaphoric

and elliptical forms in the repair phase (see example (2.19)). Computational models of reference resolution and referring expression generation that account for these results will be presented in Chapter 7.

There has been substantial work on disfluency categorization (Shriberg, 1994, 1996), but work needs to be done to create a self-repair ontology with a level of detail sufficient to inform the building of a computational model of self-repair in a dialogue system. There are slightly orthogonal proposals of self-repair types and their annotation in the literature depending on the task at hand: either more coarse-grained turn-level annotation that is interactionally focussed (Healey et al., 2005; McCabe et al., 2013) or more oriented towards automatic detection purposes (Shriberg, 1994, 1996). What is clear that an empirically solid model with appropriate detail is required, and an annotation scheme to accompany it.

One level of detail not hitherto addressed in detail is investigating precisely how the syntactic, semantic and dialogue *context* of self-repair corresponds to its form: self-repairs tend to be analysed in isolation in the more fine-grained corpus studies. There needs to be more investigation into how hearers use the incoming information incrementally to predict and process repairs, including analysis of edit terms in terms of their form and their predictive ability for upcoming repair. Corpus work investigating this will not only be useful for acquiring evidence about the context of self-repair events and their types, but could also provide a test-set of phenomena against which computational models could be evaluated. I return to this in Chapter 4, which describes a study investigating the detail required for an ontology of self-repairs in dialogue.

Chapter 3

Computational and Formal Approaches

This chapter surveys historical and state-of-the-art computational models of self-repair detection and processing and its treatment within theoretical and implemented natural language understanding (NLU), natural language generation (NLG) and dialogue systems. The commonly used repair detection evaluation techniques are introduced here, as are the principal formal and computational tools that will be used in Chapter 6. A summary of the approaches and an outline of the research questions motivating the computational and formal elements of the thesis are given in Section 3.5.

3.1 Automated processing of self-repairs

3.1.1 Detection and correction using multiple knowledge based language models

There have been several statistical language model driven systems for detecting and correcting first position self-repairs (in the computational linguistics community often referred to as *disfluency detection*), using multiple knowledge source language models which include acoustic, lexical and other information from speech. A representative sample of these approaches is surveyed here.

Heeman and Allen (1999) present a multiple knowledge source statistical language model for detecting and correcting self-repairs in the task-oriented TRAINS corpus (Heeman and Allen, 1995). This was the first time that the task of automatic speech recognition (ASR) was proposed as a joint task with tagging part-of-speech (POS), discourse markers, speech repairs and intona-

tional phrase boundaries in dialogue data, working under the premise that the outcomes to all of these sequence classification tasks were connected and would aid each other's performance.

The authors incorporate detection and correction mechanisms for speech repairs and also basic repair classification, differentiating *fresh starts* (repairs without clearly substitutive reparandum and repair phases and which restart an utterance unit), *modification repairs* (repeats and substitutive repairs) and *abridged repairs* (those consisting solely of an edit term, which contains filled pauses or discourse markers, but without reparandum or repair phases, where the following words are a syntactically felicitous continuation of the original utterance; e.g. "we need to, *um*, manage to get the bananas to Dansville more quickly" (Heeman and Allen, 1999, p. 530) and see example (2.5) in the previous chapter). The correction process for all repair types is formulated as one of successfully identifying the extent of the reparandum and the edit term, if there is one present.

Introducing the task of detecting *discourse markers* such as the italicised utterance unit-initial phrases in example (3.1) below was also novel, and they showed its interaction with repair detection and correction. They observed the various communicative functions of discourse markers, such as signalling the upcoming addition of new information (e.g. *and then* in example (3.1)) or upcoming summarisation (e.g. *so* in example (3.1)), in addition to their function as repair signals either as stand-alone edit terms, interregna or as repair onset words. They managed to identify discourse markers in their multiple knowledge source probabilistic model with high accuracy, with a recall of 0.973 and precision of 0.963.[1]

(3.1) *and then* while at Dansville take the three boxcars
so that's a total of five
(from (Heeman and Allen, 1999, p. 531))

The authors discuss the interaction of repair detection with ASR, discourse marker detection and POS-tagging. For example, repair hypotheses can be used to rule out unlikely POS assignments as certain POS combinations can only occur over a repair boundary but not in a fluent sequence, e.g. the determiner preposition sequence `DT PRP` around the interruption point + in "I can run trains [on the + in the] opposite direction" (Heeman and Allen, 1999, p. 533). The connection between repair presence and word sequence hypotheses is also clear here –"the in" is

[1] While the authors give accuracy results as percentages to 4 significant figures, here I give them in decimal form to 3 s.f. as will be done for all accuracy results in this thesis.

unlikely in a fluent sequence but is permissible at a repair onset boundary.

The authors incorporate repair tags, edit term tags and intonation phrase boundary tags as sequence models in their overall language model. The repair tag indicates the position of the start of the repair phase onset and its type, taking values of modification (*Mod*), fresh start (*Can*) or abridged (*Abr*). For edit terms, different tags indicate the extent of the edit term: a *Push* tag indicates the start, an *ET* tag between words indicates an edit term in progress and a *Pop* ends the edit term. Edit term detection allows the repair detector and corrector to work more reliably as the correspondences between reparandum and repair phases in modification repairs can be derived contiguously. An example of an utterance with a modification repair with an edit term according to this annotation scheme can be seen below:[2]

(3.2) that'll get there at four a.m. **Push** oh **ET** sorry **Pop Mod** at eleven a.m. **%**
from (Heeman and Allen, 1999, p. 546)

They build probabilistic sequence classifiers of these sequence tags and therefore redefine ASR as a task as a joint maximisation of the most likely word, POS, repair tag, discourse marker and intonation phrase boundary sequences given the acoustic signal. For each of the sequence classifiers, the authors implement decision tree learners which use local contextual information from relevant tag sequences to make decisions – this avoids estimating a large complete joint distribution but still allows the trees to select features most relevant to their task, not constraining the context only to previous sequences within their own classification task, e.g. the POS tagger could use words, repair and intonation boundary contextual information in addition to the previous POS tags hypothesised.

In their repair detection and correction classifiers they predict repair structures indicating the word correspondences between reparandum and repair phases: these correspondences include matching (repeated) words marked `m` (i.e. [I + I] =`m.m`) replacement (substitution) correspondences marked `r` (i.e. [I + you] = `r.r`), inserted words in the repair marked `x` (i.e. I [love + really love] = `m.xm`) and words deleted from the reparandum (i.e. [I +] = `x.`). One problem with the approach acknowledged by the authors was sparsity of repair structures: they report that 1,302 modification repairs (non-deletes) take on 160 different repair structures in the TRAINS corpus, with only 47 (29.4%) occurring at least twice. The correction task of identifying repairs

[2]The intonational phrase tag is a binary one that indicates whether a word ends a given intonational phrase, marked with %.

through template matching is aided by using information from the language model in the decision process to filter out false positives. They also include a feature encoding that a repair has already been detected in the utterance due to observation that in TRAINS, 35.6% of repairs overlap.[3]

Limiting their testing to transcripts, rather than evaluating the full task including ASR, they test repair detection and correction in terms of recall and precision on the transcripts. The best overall repair detection recall achieved was 0.768 with precision at 0.867, however repair correction (reparandum and edit term detection), which from an NLU perspective, is arguably the more important part of repair identification, did not perform as successfully, with an overall recall of 0.659 and precision of 0.743. Within modification repairs their performance in detection and correction was considerably better (detect r=0.809, pr=0.834; correct r=0.780, pr=0.804) than it was for fresh starts (detect r=0.486, pr=0.692; correct r=0.362, pr=0.516). Abridged repairs were processed with good accuracy (detect r=0.759, pr=0.825; correct r=0.757, pr=0.823). They also show that distinguishing fresh starts and modification repairs as separate categories was useful, showing a 7.0% boost in detection accuracy and 6.6% correction improvement over a model collapsing the distinction.

While direct comparison with other approaches explained below is not possible due to the different test data and metrics used, the apparently moderate success in reparandum detection is not surprising due to the sparsity problem. While the fact that the repeat sequence `m.m` is the most common modification repair structure may be very useful for an incremental classifier, there is a long tail in the distribution of repair structures. I assume fresh starts were particularly problematic to identify due to the lack of POS or word-level parallelism and available templates, frequent occurrences of which can be exploited in detecting modification repairs, but not for fresh starts.

In addition to the novel re-statement of voice recognition language models, a desirable property of their system is that results could in principle be obtained word-by-word incrementally, as they use Markov processes for each sequence classifier, in which classification of the current tag only conditions on the previous states rather than allowing look-ahead, however the stability of their repair detection and correction hypotheses is not discussed.

In a later system, Liu et al. (2003) also demonstrate how multiple knowledge sources including acoustic information can be used to detect disfluency interruption points, claiming this can

[3]It is not clear this is the same kind of overlap as nested and chaining repairs described by Shriberg (1994).

be best achieved by a total combination of prosodic cues, word-based and POS-based cues (Liu et al., 2003, p.957), rather than any subset of those knowledge sources. They claim the task of deciding how far to go back with the onset of a disfluency to be removed "is best found using knowledge-based rules." (ibid.). In the same spirit as Heeman and Allen (1999), they also described how "specific disfluency types can be aided by the modeling of word patterns" (ibid.). As their system architecture shows in Figure 3.1, a statistical prosody model, word-based language model and POS language model were trained to produce the best output in terms of interruption point hypotheses from speech input. Acoustic information from speech forced aligned with human transcription and ASR results were passed to a *hidden-event* language model (HELM) which was trained not just to output the best word hypothesis from the speech signal but also the most likely repair interruption points as hidden events.

They also develop a specific repetition pattern LM, motivated by the fact that specific lexical repetition patterns are sparse in the data, however as described in Chapter 2 repetition in itself is a strong indicator of repair, and as an event is far less rare than individual lexical instances of repeated words. They build the repetition LM by estimating a combination of the probability of the repetition patterns occurring in data and the probability of the 'cleaned' word sequences after the reparandum region predicted by the repetition model is removed from the word sequence. In testing, repetition events are predicted by the repetition LM and the probability for valid sequences of the repetition events are calculated. The overall probability of each sequence is calculated in the standard way for the cleaned word-based LM until the interruption point however for the repetition events, the repetition pattern LM N-gram probability is used instead of the word-based probability.

They test their model on ASR results and human transcriptions in order to avoid the effect of word errors in recognition outputs and give detailed results for the transcription condition, which had better accuracy. For detecting repair interruption points, a combined model using the word-based hidden-event LM, a similar POS-based model and a prosody model achieved a recall of 0.568 and precision of 0.813. To isolate the effectiveness of their repetition pattern language model, they test on the transcript data and achieve a recall of 0.807 and precision of 0.701 in detecting repetition repair interruption points, improving from 0.678 and 0.773 when using the word-based HELM alone.

Reparandum onset indentification using their HELM achieved a recall of 0.464 and precision

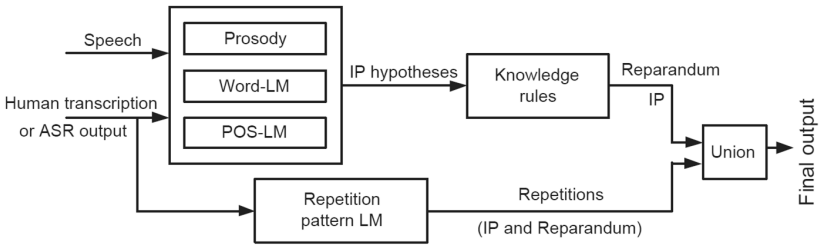


Figure 3.1: System architecture for disfluency detection and classification (Liu et al., 2003)

of 0.757, however the repetition model LM achieved improved overall accuracy (r= 0.615, pr= 0.686). They claim that the results for reparandum onset position detection would be improved by obtaining more accurate IP hypotheses.

Incorporating a disfluency model more directly into n-best list based combined ASR and statistical machine translation (SMT) system was recently achieved by Cho et al. (2014), using insights from this work and subsequent LM-based approaches. While their disfluency detection results were not competitive to other approaches mentioned here, SMT performance and ASR performance was improved due to the incorporation of the disfluency model.

3.1.2 Deterministic parsing and string editing

As stated in the introduction, the parsing community's normal implementational focus has traditionally been on written text rather than dialogue data, and for this reason it has become a technically difficult challenge to incorporate disfluencies into standard grammars and parsers. Parsing utterances with repairs was originally formulated as classifying the reparandum correctly and excising it from the parse tree of the final sentence, an approach which continues today.

Hindle (1983) proposed the first parser-first approach to the problem, introducing a deterministic parser which permitted one possible syntactic structure for a string, but invoked *correction rules* which recognized editing signals such as filled pauses and reparanda. The rules included a *surface copy editor* which simply removed from the view of the parser the left-hand string of a pair of repeated strings either side of an editing signal (i.e. interregnum). Failing that, slightly more psycho-linguistically informed repair detection was carried out by a *category copy editor*, which matched for constituent categories in its parsing buffer, and deleted the left-hand copy, and

stack copy editor which looked for copies of incomplete structures in the reparandum to expunge.

This system was elegant, however it is worth noting that one of the strategies employed, the surface deleting of the reparandum from the input string, does not make for good psychological plausibility according to the evidence in Chapter 2. If the model was considered a model for understanding speech, taking physics into consideration, let alone cognitive processes, would make it unrealistic, as the words cannot be *un*-heard, however this approach still underlies repair detection research (Rasooli and Tetreault, 2014; Honnibal and Johnson, 2014), as I will discuss below. Furthermore, Hindle's parser used known interruption points and worked on the correction part of repair processing rather than detection, however detecting interruption points is a non-trivial problem for automatic approaches.

3.1.3 Processing speech repairs with a noisy channel model

Johnson and Charniak (2004) present the first generative approach to processing self-repairs, in a noisy channel model of speech repair detection. They formulated the task of parsing utterances with repaired speech as finding the 'cleaned' utterance that is most likely one generated by an underlying fluent source language model and the most likely to generate the observed 'noisy' utterance (the 'noise' being the disfluencies)- see Figure 3.2. For their channel model, they build a S-TAG (Synchronous Tree Adjoining Grammar Shieber and Schabes, 1990) based transducer that yields complex sentences which are strings of tuples of words from the 'noisy' sentence (raw utterance with disfluencies) and corresponding words from the source sentence (clean underlying 'intended' utterance), using simple S-TAG rules.

The system is trained to yield string pairs which maximise the probability of the overall noisy channel model $P(X|U)$ yielding cleaned utterances X from raw utterance strings U. Using the Bayesian noisy channel model formulation, this is achieved by a maximisation of the likelihood of the combination of the S-TAG based channel model $P(U|X)$ generating the noisy strings and the language model $P(X)$ as in equation 3.3. The decoding task can therefore be viewed as search for the most likely underlying clean sentence $x \in X$ given the observed noisy utterance u.

$$\arg\max_X P(X|U) = \arg\max_X P(U|X)P(X) \tag{3.3}$$

The S-TAG grammar in the channel model generates sentences Z, where each $z \in Z$ consists of a sentence of tuples of the form $\langle$raw word in u, cleaned word in $x\rangle$. The second element will

be $\oslash$ (the null string) if it is classified as part of a reparandum (i.e. removed from the output), so an underlying cleaned sentence x consisting of the string of the first tuple elements will always be a substring of the noisy sequence u, the string of the second tuple elements. If the second element of the tuple is not a reparandum word then both elements have the same lexical value.

Note the TAG rules do not assign grammatical structure to words (i.e. the TAG parser is not a syntactic parser), rather they generate the strings of noisy utterances U from the underlying cleaned utterances X and yield a tree structure representing the repair-reparandum alignments such as that in Figure 3.3. The model uses the context-sensitive properties of TAG (specifically the ability to deal with crossed serial dependencies) as a way of dealing with the 'rough copy' dependencies often present in speech repairs.

The auxiliary trees used in the derivations have the tuples $\langle$raw word in u, cleaned word in $x\rangle$ as their terminal nodes, i.e. the words that compose sentences Z as described above, and simple reparandum-repair alignment rule categories for their non-terminals (*copy*, *delete*, *insert*, *substitute*), indicating the correspondence between their left and right daughter terminals. They contain the repair category alone if they have a single daughter, i.e. in the case of nodes where interregnum trees are attached. More technically, as can be seen in Figure 3.3, the non-terminals divide into three categories – N_{w_x} (the preceding word w_x is not part of a repair), $R_{w_x:w_y}$ (the preceding word in a reparandum and its corresponding word in a repair phase, if these two words are identical then it is a repetition, if they are different then there is a substitution, if w_x is $\oslash$ this is an insert and if w_y is $\oslash$ this is a delete) and the other non-terminal is I, which dominates the interregnum word sequences. Note the trees with N_{w_x} and I mothers always rightward branch in a finite state fashion, which allows the probability of these rule applications to be obtained by normal n-gram language model estimation– the authors train a bigram model for the non-repair N_{w_x} headed trees and a unigram interregnum model for I headed trees. See the derived tree for "..want a flight [to Boston, + { I mean } Denver]" in Figure 3.3.

The S-TAG parser runs in $O(n^5)$ on the length of the input sequence, which they limit to word windows of length 12 as the system stochastically predict a repair as beginning every word. A chart is used to store all the possible repair sequences. The space and time complexity issues here will be discussed in Section 3.5 and also in Chapters 4 and 5.

For their evaluation they use what has become the standard metric for evaluating repair detec-

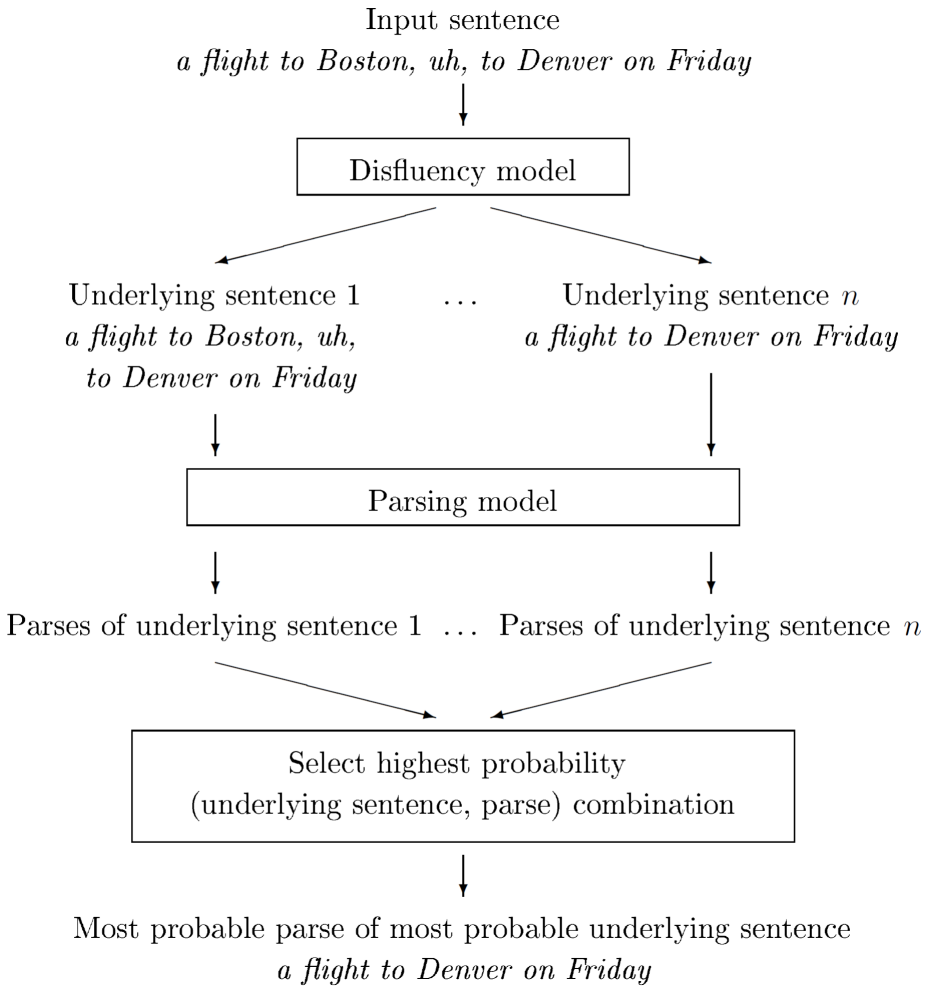


Figure 3.2: Parsing model for disfluencies from Johnson (2011).

tion systems, first introduced in Charniak and Johnson (2001)- the F-score[4] of reparandum words *rm* retrieved. To calculate the precision and recall to give this result, if we take the total number of words hypothesised as being in a reparandum as rm_{hyp}, the number of correct hypotheses $rm_{correct}$ and the total number of gold standard reparandum words in the transcript as rm_{gold} we have:

[4]This is technically the F1-measure but will be shortened to 'F-score' for the remainder of the thesis.

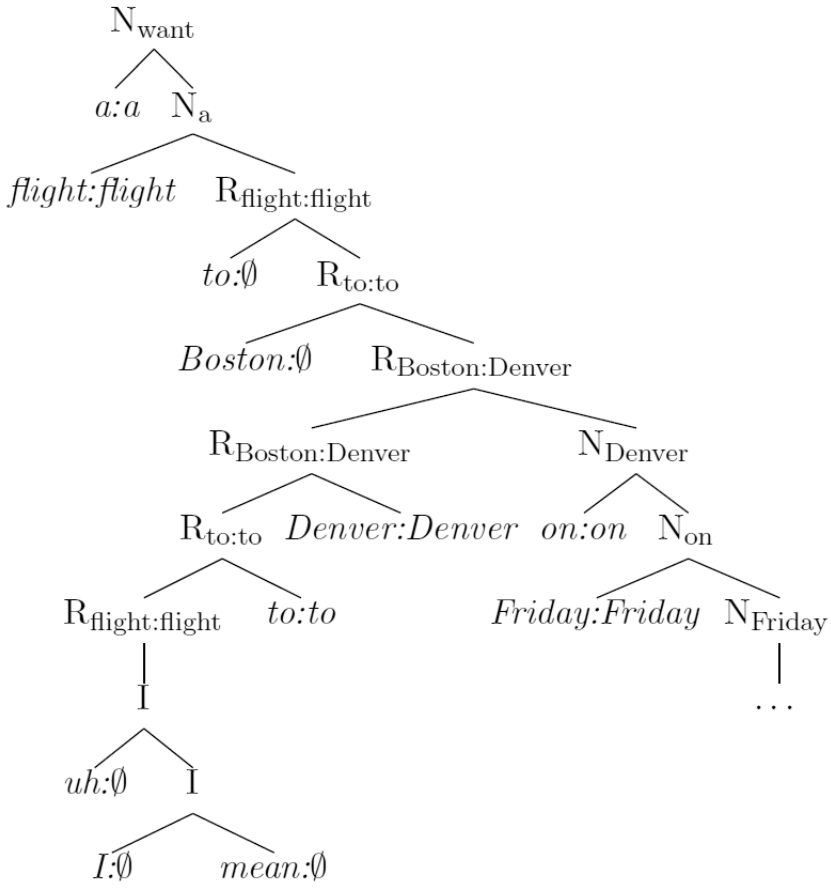


Figure 3.3: TAG-based derivation of a repaired utterance (Johnson and Charniak, 2004).

$$\begin{aligned} \text{precision} &= \frac{rm^{correct}}{rm^{hyp}} \\ \text{recall} &= \frac{rm^{correct}}{rm^{gold}} \\ \text{F-score} &= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \end{aligned} \tag{3.4}$$

In testing their model they show how the system performs best by using a statistical parser based language model for $P(X)$ with an F-score of 0.798, rather than using bigram (F-score = 0.756) or trigram (F-score = 0.768) language models. It is worth mentioning their model was not trained on overlapping repairs, which is surprising given that a grammar-based approach should be more suited to this problem than sequence labelling approaches, given their embedded

structure (Shriberg, 1994).

Incrementalising the noisy channel model

The model I consider most suitable for incremental dialogue systems in previous work is Zwarts et al. (2010)'s incremental version of Johnson and Charniak (2004)'s noisy channel repair detector, as it incrementally applies structural repair analyses and is evaluated for its incremental properties. Following Johnson and Charniak (2004), instead of using a parsing model their system uses an n-gram language model trained on roughly 100K utterances of reparandum-excised ('cleaned') Switchboard data. As above, the channel model is a statistically-trained S-TAG parser whose grammar has simple reparandum-repair alignment rule categories for its non-terminals and words for its terminals. The parser hypothesises all possible repair structures for the string consumed so far in a chart, before pruning the unlikely ones, however these are processed in a strictly left-to-right manner from the input string. It performs equally well to the non-incremental model by the end of each utterance (F-score = 0.778), and can make detections early via the addition of a speculative next-word repair completion category to their S-TAG non-terminals.

In terms of incremental performance, they report the novel evaluation metric of *time-to-detection* for correctly identified repairs, achieving an average of 7.5 words from the start of the reparandum and 4.6 from the start of the repair phase. They also introduce *delayed accuracy*, a word-by-word evaluation against gold-standard disfluency tags up to the word before the current word being consumed (in their terms, the *prefix boundary*), giving a measure of the stability of the repair hypotheses. They report an F-score of 0.578 at one word back from the current prefix boundary, increasing word-by-word until 6 words back where it reaches 0.770. These results are the point-of-departure for the work in Chapter 5.

3.1.4 Other successful machine learning approaches

There has been a competitive effort to improve accuracy for disfluency detection on transcripts since Johnson and Charniak (2004)'s work. Qian and Liu (2013) achieve the best reported performance on the Switchboard disfluency test corpus, achieving an F-score for detecting reparandum words of 0.841. They use a three step detection system using weighted Max-Margin Markov (M^3) networks: (1) detection of edit-terms/fillers/interregna (2) detection of reparandum words, and (3) refining the previous steps, using a cost-sensitive error function.

In an earlier model Georgila (2009) introduces a post-processing method of Integer Linear

Programming (ILP) to improve overall accuracy of various off-the-shelf methods, reporting an F-score for detecting reparandum onset words at 0.808 and repair onsets at 0.825 for a CRF model. While these results are impressive, the systems do not operate incrementally: they maximise the overall likelihood of tag sequences in utterances, using utterance-global constraints, rather than focussing on incremental accuracy, and so are not as suitable as other approaches discussed here for the purposes of this thesis.

3.1.5 Joint parsing and repair detection

Recently, there has been increased interest in left-to-right repair detection: Rasooli and Tetreault (2014) and Honnibal and Johnson (2014) present dependency parsing systems with reparandum detection which perform similarly, the latter equalling Qian and Liu (2013)'s F-score at 0.841. However, while operating left-to-right, these systems are not designed or evaluated for their *incremental* performance. The use of beam search over different repair hypotheses in Honnibal and Johnson (2014) is likely to lead to unstable repair label sequences, and they report repair hypothesis 'jitter' in reparandum word detection. Both of these systems use a non-monotonic dependency parsing approach that immediately removes the reparandum from the linguistic analysis of the utterance in terms of its dependency structure and repair-reparandum correspondence, which I will argue does not allow optimal integration into interesting NLU systems.

Miller and Schuler (2008) presented an earlier left-to-right parsing and repair detection model that achieved an F-score of 0.680 on the Switchboard test data. They use right-corner-transform of syntactic trees to make an EDITED headed sub-tree be derivable incrementally and achieved the best results by using a Hierarchical Hidden Markov Model (HHMM) as their parsing mechanism. The reason for the moderate success may be the encoding of repairs into a grammar causes sparsity in training: repair is a general processing strategy not restricted to certain lexical items or POS tag sequences. Again no incremental performance information is given. This was the first attempt to directly incorporate self-repair into the grammar directly, rather than orthogonally to it.

3.2 Incremental NLG and self-repair

In this section I discuss various approaches to incrementality and self-repair within natural language generation (NLG) frameworks.

3.2.1 Background: NLG architectures

Before reviewing some of the more psychologically motivated models of generation that deal with self-repairs, the standard structure of an NLG system should be considered.

The most well known functional decomposition in NLG is the separation of the *strategic* and *tactical* levels of language generation. This was first outlined by Thompson (1977), who claimed a functional modelling approach to human language production could do more than contemporary theoretical linguistics, particularly the transformational grammar tradition, in terms of the *explanatory adequacy* called for by Chomsky (1965). Thompson claimed linguistics was failing to deliver the promise of an explanation as to *how* language users produce utterances, so he formulated the problem in terms of designing a computational system capable of generating human-like linguistic behaviour.

Thompson positioned the strategic level as the more complex and interesting of the two levels, claiming that given high-level, not necessarily linguistic, information received from the rest of the cognitive model such as the *intention* to bring about some state of events through speech, a model of the *hearer* and a representation of the *propositional* content to be communicated, it could make decisions including selecting speech acts and lexical items and how best to *chunk* the information; its role was essentially to decide *what to say*. This information could then be passed on in a serial manner to the tactical level, whose easier task was to transform it into a sequence of words to send to the speech channel, using linguistic constraints applied to the lexical items; the role of the tactical component was effectively making the final decisions as to *how to say it*.

Thompson's characterization of generation reduced the import of the tactical level, claiming if the nature of it was taken as "given" (Thompson, 1977, p.656), research would be free to focus on the more "*important* encoding decisions" (ibid., p.656) of the strategic component. However, the nature of tactical generation continues to be far from given and concordance amongst theories seems hard to come by. Even more strikingly, the overall architecture of NLG systems is far from uniformally agreed upon: since the tactical-strategic distinction was made, three-part (Levelt, 1989; Reiter and Dale, 2000), multi-modular (Kempen and Hoenkamp, 1987; Neumann and Finkler, 1990; Zarrieß and Kuhn, 2013) alternatives have been proposed, with modularity being employed for reasons of computational efficiency and psychological plausibility. Furthermore, an NLG system designed for interactive purposes such as dialogue modelling requires further adjustments from standard architectures, including interleaving with parsers (Neumann, 1998;

Purver and Kempson, 2004) and interfaces to other components of a dialogue system (Skantze and Hjalmarsson, 2010), as will be discussed below.

Logical form equivalence and inputs to generation

Deciding on the most suitable representation for the inputs to a generation system is an unresolved problem. Not only is there is a lack of standardized input (Belz et al., 2010), there is the issue of deciding on which parts of the generation system should be able to access the input. The foundations of these problems have been formalized as the *problem of logical form equivalence* (Appelt, 1987; Shieber, 1988, *inter alia.*), which originated from a goal to eliminate the need for the strategic component of generation to have any grammatical knowledge.

Following Shieber (1993) and Van Deemter and Halldórsson (2001)'s explanation, the problem is roughly as follows: given that the strategic component of a generator, the *reasoner*, is non-linguistic in nature, its input to the tactical component of generation is a logical form (LF) which is not influenced by the grammar. If two LF input representations have the same meaning as far as the logic of the reasoner is concerned (e.g. $P \rightarrow Q$ and $\neg P \vee Q$ if the LFs were propositional logical forms), given the role of the tactical generator is to find the appropriate meaning-string pair in the grammar, it should generate identical strings given these two different inputs. However in the grammar within the generator each string is paired to one LF (the *canonical logical form*) for a given interpretation, without necessarily being linked to the other LFs that represent the same meaning as, or at least are *logically equivalent*, to the canonical LF.[5] This one-to-one pairing from LF to meaning is a necessary feature of generation for it to remain a tractable problem: while $P \rightarrow Q$ and $\neg P \vee Q$ may 'mean' the same thing in the logical language, if we assign P = "Bert dances" and Q = "Ernie sings", intuitively, the string "if Bert dances then Ernie sings" should not be generated by $\neg P \vee Q$, nor should "Bert does not dance or Ernie sings" be generated by $P \rightarrow Q$. Given certain equivalence rules in logic such as commutativity (e.g. $P \wedge Q \equiv Q \wedge P$) or the conditional equivalence given above ($P \rightarrow Q \equiv \neg P \vee Q$) many traditional (and formally well-behaved) logics are therefore unsuitable for providing LFs.

This problem would suggest that the notion of equivalence of two or more LFs in the input representation would need to be narrowed somewhat from logical equivalence to something closer to *syntactic identity*, and hence the strategic component must be adapted to the requirements of the tactical generator and the grammar. Shieber (1993) suggests this approach just

[5] Ambiguous strings may of course have more than one candidate canonical LF, but this is irrelevant to the problem as they are different interpretations rather than equivalent ones.

escapes the problem, claiming it is rooted deeper in the AI knowledge representation problem and the philosophical problem of meaning identity. However Shieber suggests a resolution could be possible with a linguistically oriented suggestion: it should involve a new input representation language "that characterizes exactly the semantic distinctions that ramify in natural language syntax." (ibid, p.187)

The problem of logical form equivalence continues to underlie several current directions in NLG research, in particular the proposal to agree on *common inputs* to generation systems (Belz et al., 2010) across the field. This is presented as a practical difficulty for researchers for evaluating different computational systems, given the lack of gold standard NLG input and the immense variability in input for different systems, with input forms variously being more or less specific to a particular grammar and domain of application. Although the common inputs problem is more of an engineering issue rather than a philosophical one, it is rooted in the same fundamental question as the logical form equivalence problem: *what form should the input to generation take, and which parts of the generation system should receive it?* This is a question that will be addressed in this thesis, and in particular with regard to the input to NLG in dialogue systems that include repair generation. I discuss a formalism for inputs to generation which I believe begins to meet the requirements for such a task in Chapters 6 and 7 to address this issue.

Pipeline approaches

A standard NLG architecture consists of a chain of *domain planning*, *microplanning* and *surface realization* modules, often called the "NLG pipeline" (Reiter and Dale, 2000). The processes involved in traditional generation are not generally incremental in most of the senses of incrementality discussed in this thesis, however the three-part modularization divides up the role of strategic and tactical decisions to be made and messages of different types constitute the increments passed through the system. Domain planning uses the static knowledge base of the system and the requirements of the particular generation task to construct a document plan which can be sent to the microplanner, which plans the sentence, executing such tasks such as *lexicalization* (choice of words) and *linearization* (word order), producing a more fine-grained specification to send to the surface realizer, which then makes final morphological decisions before delivering the surface form of the text – see Figure 3.4.

The standard practice is to design a system that passes fully formed messages on to the next module, so the revision of the original communicative goal will require a new generation cycle to

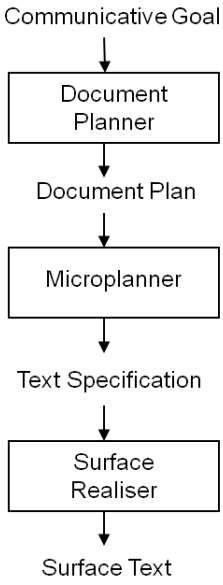


Figure 3.4: Standard NLG system architecture (Reiter and Dale, 2000, p.60)

begin afresh. The downstream stages are essentially a refinement process of the original domain planner's output. For commercial systems, the process is often driven by the generation of static texts in static domains, so a notion of real-time self-repair is not required in the architecture.

3.2.2 Incrementality in conceptualization, formulation and articulation

There is a sub-field of Natural Language Generation (NLG) research whose object of study is not the automatic production of text or synthesized speech from non-linguistic data for the benefit of system users, but computational models of cognitive processes that underlie human language production (McDonald, 1987), and this sub-field considers incrementality as a key problem in generation.

As mentioned with Thompson (1977)'s approach of functional decomposition of the production system, psycholinguistically-motivated NLG took on the task of implementing emerging psycholinguistic models of speech production. The task was formulated as the design of a multi-modular process that did not require complete input plans for sentences before beginning their production. To this end, a distinction between the different components of a generation system became important, as did the passing of incremental units between them. This approach was largely motivated by (Kempen and Hoenkamp, 1987) and (Levelt, 1989)'s separation of production into distinct *conceptualization*, *formulation* and *articulation* phases (see Figure 3.5), a

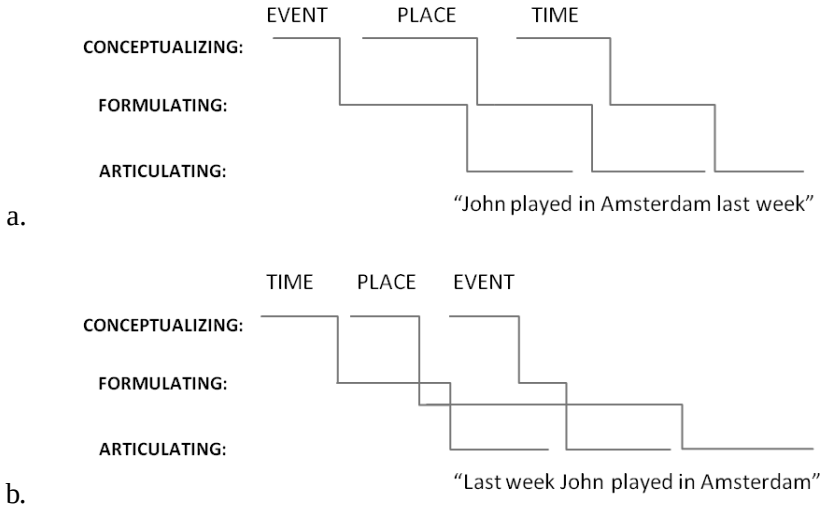


Figure 3.5: Incremental production without and with inversion of order. From (Levelt, 1989, p. 25)

psychological model which still has a bearing on NLG today.

These systems were incremental insofar as transfer of information within the generator was piecemeal, but they were not always necessarily strictly word-by-word incremental in terms of their output. The general programme of research was that input from a conceptualization module to a grammar-based formulator could be partial, as could a formulator's input to an articulator, so syntactic processes determining surface form elements like word order and inflection could begin before the entire input LF for a sentence had been received. This follows Wundt's Principle that each processing component should be triggered into activity by a minimal amount of its characteristic input (Levelt, 1989, Chapter 1.2). Neumann and Finkler (1990) describe this kind of incremental generation as "immediate verbalization of the parts of a stepwise computed conceptual structure – often called *message*)" (ibid., p. 288).

Kempen and Hoenkamp (1987) made the first detailed attempt at describing a generation implementation, introducing the Incremental Procedural Grammar (IPG) model. Schematically, IPG was driven by parallel processes whereby a team of syntactic modules worked together on small parts of a sentence under construction, with the sole communication channel as a stack object (with different constituents loaded onto it), rather than the modules being controlled by a central constructing agent. The system was designed under a premise consistent with the emerging psychological models that tree formation was simultaneously *conceptually* and *lexically* guided (van Wijk and Kempen, 1987), and that production did not take place in a serial manner. IPG was

implemented in `LISP` as a Dutch sentence generator, and was shown to be capable of generating elliptical answers to questions and also some basic self-repairs.

De Smedt (1990) took incrementality a stage further, and developed a fairly comprehensive computational model of incremental generation in which self-repair is incorporated explicitly. De Smedt developed parallelism implicit in Kempen and Hoenkamp's IPG model by implementing parallel processing *within* the formulation stage of generation, with a particular focus on incremental construction of syntactic structure in sentence generation. De Smedt proposed the Incremental Parallel Formulator (IPF), a module for grammatical encoding which could operate with input that underspecified sentences. In this case the input increments were abstract conceptual messages representing semantic conceptual relations, semantic role relations or lexical feature specifications.

The IPF operated in accordance with Kempen (1987)'s criteria for incremental generation: input from the conceptualizer should be fragmentary and not guaranteed to be sent in an order corresponding to a particular sentence's surface linear left-right word order; as a consequence, generation should be be able to proceed from the bottom of a syntactic structure upwards as well as from the top down. The IPF showed how language generation should also exploit variations in word order as made necessary, but still observe linguistic restrictions.

The formulator constructed syntactic structures by applying unification operations on 'syntactic segments', the principal units of the unification-based lexically-driven formalism *segment grammar* (Kempen, 1987; De Smedt and Kempen, 1991). Segments were TAG-like structures with two nodes (the *head* and *foot*, labeled with grammatical categories such as NP and N and containing feature structures such as *nominative (+)*), and an arc representing a grammatical function e.g. an *S-subject-NP* segment represents a subject relation between a sentence and a noun phrase. Unification operated via the combination of two compatible segment nodes into a new segment that shared their syntactic features. In tree construction terms, this is the attachment of auxiliary trees to the currently derived tree, as in a TAG. The procedure differed from the early TAG formalisms however by making a sister-node attachment operation (*furcation*) available, which allowed for different parts of a structure to be worked on in parallel before unifying them.

The IPF had two internal components: the *Grammatical Encoder*, which generated f-structures (TAG-like tree structures representing functional and dominance relationships between constituents) to which segments were attached, which in turn were used to generate c-structures

(data structures including features representing word order and other grammar specifications like case); and the downstream *Phonological Encoder*, which was responsible for executing the word ordering and correct inflection in accordance with the c-structure features. The grammatical encoding procedure could begin as soon as the first conceptual fragment entered the IPF, beginning with an empty segment *SIGN*. The formulator attempted the *unification* operation of each lexical entry segment in the lexicon with the existing structure and if successful these unified structures could be stored in the Unification Space as candidates for sending on to the Phonological Encoder, allowing multiple structures to be worked on in parallel.

De Smedt (1991) introduced a development to the IPF to allow revisions of syntactic structures in the generation procedure, providing a computational explanation for overt and covert syntactic self-repair. This was achieved by making the unification procedure "non-destructive", in the sense that the original configuration of two nodes was preserved after a unification operation, while operations on them with other syntactic constituents were still permitted as if they were unified as one structure. The accessibility to component parts of unified structures meant that no undoing of unification had to be executed at any point. The computational overhead of this extra storage and search space was not discussed.

De Smedt used the connectionist concept of *activation* in assigning real number values to the bonds of the "virtually" unified segments denoting the probability of them eventually becoming properly unified, to differentiate between strong and weak candidate structures. The author also experimented with *annealing*, whereby node activations would be set to decay over time if not unified. If a steady equilibrium was reached with a frozen configuration of segments (a state of *conformation*) the strongest remaining structure could be passed to the Phonological Encoder. The simulation of speech errors was achieved through this time-constrained annealing process: if there was not a clearly strong enough candidate or conformation after a given amount of generation time, the "incorrect" segment could be passed on. A lexical selection error such as "The next *speaker* will be given by Jonathan Slocum" (ibid.) was characterized as the presence of equally viable alternatives in the Unification Space, and possible incorrect concatenations or furcations of segments. Additionally, annealing allowed a cognitively inspired implementation whereby experiments that allowed more or less time between inputs gave different surface results, as the competing segments could optimally reconfigure with more time, simulating speech errors under time pressure.

Incremental conceptualization

While De Smedt's work on the grammatical formulation side of generation was thorough, it did not address the nature of the conceptualizer that sent the input messages to it, as atomic messages were passed to the IPF incrementally 'by hand'. Guhe and colleagues began to address this void in computational models of language production by developing the Incremental Conceptualizer (INC, Guhe and Habel, 2001; Guhe, 2007), the principle behind it being to incrementally and automatically create and send pre-verbal messages to the formulator in a cognitively motivated way. The generation task here began in a top-down manner, beginning with the incremental production of pre-verbal messages.

Guhe was interested in the idea of conceptual change in the input data and considered self-repairs from this perspective, distinguishing them from performance errors such as incorrect lexical access or misconception, which could be attributed to system malfunction. For testing, Guhe and Schilder (2002); Guhe (2007) chose a dynamic domain of a simple airport scene which had a variety of live scenarios, in order to evoke change in the input concepts which could cause both overt and covert self-repairs such as those below:

(3.5) "CK-314... uh... is delayed" [*covert*]

(3.6) "CK-314 is on time... uh... is delayed" [*overt, formulator occupied*]

(3.7) "CK-314 is on time... uh... CK-314 is delayed" [*overt, concept changed after formulation*]

A simple version of self-monitoring (Levelt, 1989) was employed in INC's *error detection* mechanism, whereby a parse of the output was compared with the planned utterance, a difference therein automatically stopping the current generation and triggering a marking of the part of the utterance to be repaired. A correction term was then generated (i.e. "uh" or "no") and the content to be corrected (the information difference) was passed to the formulator. The incremental generation of concepts in the conceptualizer was triggered by atomic perceived entities (based on a dynamically changing virtual scene at the airport), simulating real-time processing, and given a changing environment, the generator would have to be able to adapt its output quickly– this is a classic use case of incremental generation and self-repair.

A semantic underspecification formalism CLLS (*Constraint Language for Lambda Structures*) a framework for the partial description of lambda structures, was used to incrementally

compose the conceptual messages. The message generation procedure consisted of 4 operations-*construction*, *selection*, *linearization*, and *PVM(pre-verbal message)-generation*, which all operated on the *current conceptual representation* (CCR), a hierarchical semantic network that represented the internal state of the conceptualizer. The CCR was first built up by the construction process through a concept matcher linked to a concept store. The construction algorithm worked recursively with the matcher until no more complex concepts could be constructed from simpler ones, until a newly perceived entity arrived to be handled. The selection process chose the concepts to be verbalized from the CCR, which were then linearized into an appropriate order (logically, not into final word order), and PVM-generation incrementally produced a pre-verbal message by taking the first element out of a traverse buffer (a sub-structure of the CCR), and passing that part of the PVM onto the formulator, continuing in an incremental fashion.

Repairs could be triggered due to the fact that as soon as an increment was sent to the formulator it became inaccessible to the conceptualizer- generating corrections was the only way to change information. Upon new information arriving which significantly changed a concept in the PVM being sent to the formulator, the difference between the planned and actual utterance content was computed and a correction increment was generated containing information about which concept to change, which concept to be deleted by the formulator, and which information to be added by the formulator. The formulator received this correction increment, and then made decisions about how the correction was to be treated in accordance with the modular division-of-labour postulated by De Smedt (1990). Guhe and Schilder showed the consistency of their CLLS correction algorithm with the parallelism constraint commonly attributed to verb-phrase ellipsis. Informally, the lambda structure in CLLS for a correction was structurally the same as for a coordination, so it could be added to the incremental preverbal message simply as another increment. This increment could then combine with the alternation by beta-reduction in the parallel correction structure to yield a message such as CK-314($\lambda x.correction(on\_time(x), delayed(x))$), a message invoking an overt repair in the formulator such as (3.6) above, and $correction$(CK-314($\lambda x.on\_time(x)$),CK-314($\lambda x.delayed(x)$)) which would cause a more lengthy overt repair (see (3.7) above).

Guhe's work showed how concepts could be monotonically constructed and trigger repair strategies in the formulator and also developed a rudimentary semantic representation for repair concepts. These achievements fell within the larger research programme of casting the generation

of pre-verbal messages as an incremental procedure, however the work avoids the impenetrable problem of the initial autonomous generation of a concept, rather starting from the input of live-events which trigger the input of some predefined conceptual increments to the system. While it is possible to compare the verbal output of the system with that of human beings as shown in Guhe (2007), INC would be difficult to evaluate in terms of its individual contribution in a quantitative way, as pre-verbal message LFs (the input for tactical generation), do not have a widely agreed form (Belz et al., 2010, Section 3.2.1). This is not the case for measuring similarity of string outputs given a gold standard generation input or LF, as is the case for more common surface realisation tasks, for which a wide variety of metrics exist. Also, while the system operated in a dynamic and changing domain it was not interactive with users: if operating in a dialogic extension of the domain, such as describing the moving scene to a partner who could query the descriptions, more interactive requirements would be put on the conceptualizer.

3.2.3 Interleaving parsing and generation

As for more interactive approaches, notable work in interleaving generation with parsing in an incremental fashion came from Neumann (1994, 1998), who showed how the processes could be connected using a reversible grammar. The psychological motivation, as with Guhe's feedback loop, came mainly from Levelt (1989)'s production model. Reversibility of the representations for use by the parser and the generator was achieved by utilising HPSG-like attribute-value matrix objects for each utterance, termed *items* (the logical form (LF) of the sentences). In Neumann's model, the input of one module operated on output of the other (i.e. parsing: $string \mapsto LF$; generation: $LF \mapsto string$). The items uniformly contained an underlying LF, along with the corresponding string of that sentence, and depending on the component's required input, the item would have one of these fields present but not the other (the parser would take items with instantiated string variables but with uninstantiated logical forms, and vice-versa for the generator).

Following Shieber (1988), Neumann developed a processing model that could run in both the parsing and generation modules when processing items, using the Uniform Tabular Algorithm (UTA), a data-driven selection function which was a generalization of the Earley deduction scheme. The UTA algorithm had a uniform indexing mechanism for items and an agenda-based control that allowed item sharing between parsing and generation. This way partial results computed in one direction could be computed in the other, a desideratum of interactive generation that will be discussed in the initial proposal in Chapter 6. With its uniform underlying deduction

mechanism and reversible grammar, Neumann's model was computationally efficient and had the potential to be adapted for an interactive dialogue setting, but this was not done and it was only designed to parse its own utterances for ambiguity checks. Another reversible grammar formalism that has extended to dialogue modelling, Dynamic Syntax (Kempson et al., 2001), will be described below for its suitability for dialogue systems in Section 3.3.3.

3.3 Self-repair and incrementality in dialogue frameworks and systems

This section reviews dialogue approaches to self-repair and also surveys the incremental processing formalisms and tools which will be used in Chapter 6.

3.3.1 Self-repair in the Incremental Unit framework

Incremental dialogue systems enjoyed a notable theoretical and implementational development in the proposal of an abstract incremental architecture, the Incremental Unit (IU) framework Schlangen and Skantze (2009, 2011). Several interactive systems using its incremental multimodular specification have been since developed, including those with NLG and voice synthesis modules (Skantze and Hjalmarsson, 2010; Baumann, 2013). The generation of mid-utterance back-channels (Skantze and Schlangen, 2009) and interruptions (Buß et al., 2010), phenomena that require continual interaction between speech recognition, parsing, generation and voice synthesis have been shown to be more tractable problems within such an architecture.

The IU framework can be described as a network of modules, each comprising a *left buffer* for input *incremental units* (IUs), a *processor* and a *right buffer* for the output IUs. IUs have a *payload* which determines what kind of data they carry, whether it is a word, POS tag or numerical value, or anything else determined by the system designer. It is edit actions consisting of *add*, *commit* and *revoke* actions on IUs in a module's right buffer and the effect of doing so on its downstream modules' left buffers that determines system behaviour. Furthermore, the IUs can have *same level link* relations between one another if it is desirable that they should be in some way inter-dependent within a module buffer, or have *grounded in* relations between different module buffers. The buffers are defined as graphs with nodes that represent IUs, allowing for multiple hypotheses to be constructed with time-linear input and their subsequent revision. These desirable incremental properties will be exploited in the proposed formal computational model in Chapters 6 and 7.

Speech plan generation in Jindigo

Skantze and Hjalmarsson (2010) implement an incremental canned-speech based vocalizer (generation and TTS) module in the incremental dialogue framework Jindigo, the Java-implemented dialogue system based on Schlangen and Skantze's abstract specification. Their implementation does not rely on end-of-utterance silence thresholds from the ASR module before beginning the generation of a response, so the latency of response is greatly reduced compared to a non-incremental version. The chain of incremental updates occur in its buffers to allow incremental generation: *word* hypotheses are made for incoming auditory input, which are sent in real time to the interpretation module's input buffer, which in turn processes these different hypothesis to add *concepts* to the dialogue manager's input buffer, which in turn processes these to generate a `SpeechPlan` for the vocalizer. In the face of lack of commitment of complete IUs from its upstream modules, the vocalizer may start adding `SpeechSegment`s such as "eh" and "well, let's see" to allow immediate response without having to wait for complete input.

The `SpeechSegment`s, while sometimes spanning several words (e.g. "it is blue"), are semantically atomic, however word-by-word generation is achieved by further dividing the segments into word-length `SpeechUnit`s to be processed serially by the vocaliser. This incremental division gives Jindigo its mechanism for self-repair in the face of changing speech plans during generation: a cross-checking of the speech plan currently being vocalized against the new candidate speech plan gives the optimal word/unit position from which the repair can be integrated. Self-repair is therefore possible if input concepts are revised after commencing the vocalisation of a plan if commitment to it is revoked, both covertly (before synthesis) and overtly (after synthesis), and on both the segment and unit levels (see fig. 3.6).

While their model is not as clearly psychologically motivated as some of the generation work mentioned above, for instance not being syntactically oriented like Kempen and Hoenkamp, De Smedt and Neumann's frameworks and without the fine-grained semantic input of Guhe's model, Jindigo's ability to allow a maximal amount of incremental information flow between all the modules in the dialogue system allows the possibility of more interactive and responsive NLG. It not only allows parallelism within the generation process itself, but also allows for incremental dependency on other decision processes within the dialogue system in generation outcomes, and in terms of a psychological analogue, a better interface with the rest of the cognitive model. The flexibility in the specification of different module behaviours allows the testing of different

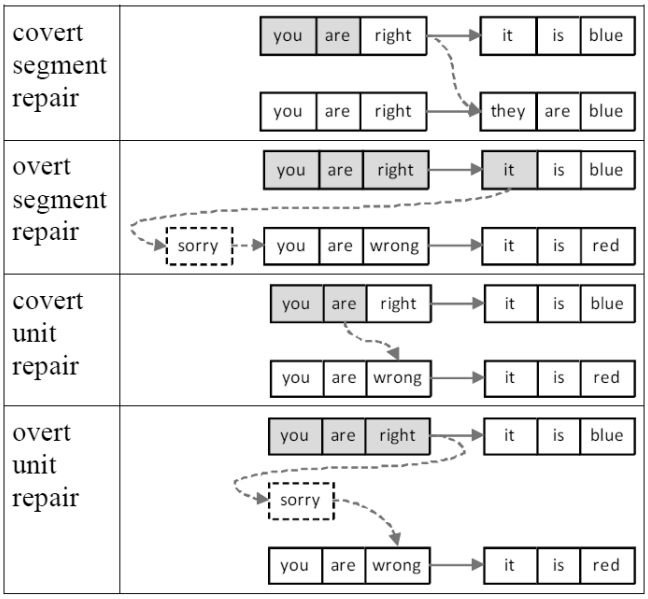


Figure 3.6: Different types of speech repairs in Jindigo vocalizer module. Shaded areas show which `SpeechUnits` have been realized, at the point of revision. from (Skantze and Hjalmarsson, 2010)

theories and implementations for individual components of speech production, and hence situates the dialogue system as a *tool-for-understanding* (see Schlangen, 2009, for an explanation of this approach).

It is also worth mentioning that in Skantze and Hjalmarsson's evaluation, using an innovative Wizard-of-Oz experiment, consistent with evidence in Aist et al. (2007) that incremental dialogue systems seem more efficient and pleasant to use than their non-incremental counterparts, they found users similarly preferred an incremental generation system over a non-incremental version in terms of ratings of *politeness*, *efficiency* and *indication when to speak*. They found no difference in user response times between the two systems, which seems to fail to give support to Brennan and Schober (2001)'s claim of increased speed in response times upon hearing corrections. However, this measurement was presumably taken from the end of system's utterances to ensure comparison across all utterances rather than from the onset of particular semantically salient words as it was in Brennan and Schober's experiments, so comparison is difficult here. The presence of self-repair certainly does not hinder response time here, in any case. Their con-

tribution is valuable in terms of the evaluation challenge for incremental NLG, as the method isolates a capability of the system that can be controlled for, exhibiting notable interactional differences.

Incremental dialogue management with self-repair capability

The Jindigo implementation of self-repair was exploratory in terms of testing interactional effects, however as it functioned within a Wizard-of-Oz setting rather than an end-to-end dialogue system, it is difficult to claim it is a generation implementation. Buß and Schlangen (2011) address the challenge of generating corrections and representing repair on a discourse level through their system DIUM, an incremental dialogue management module that functions in an implemented IU framework-based dialogue system.

DIUM addresses the need for a dialogue manager to self-repair in light of needing to produce output that conflicts with system behaviour that has already been publicly realised. It simultaneously addresses the easier problem of covert repairs, where conflicting information is not realised, by using the IU network's edit message *revoke* to remove the information that is in conflict with the current plan and also can revoke the IUs that are *grounded in* (i.e. were triggered by) the revoked IUs in the dialogue manager. It achieves its revision capability through characterising its internal information state as an IU network that allows the edit messages (`add`, `commit`, `revoke`) to operate on internal information rather than simple string output as in Skantze and Hjalmarsson (2010)'s system. The revision of internal representations is made possible by incrementalising concept frames and characterising them as an IU network themselves as in Figure 3.7. They also introduce *SemIU*, *DiscourseIU* and *DialogueActIU* incremental units to differentiate the factual content established, the issues (i.e. frames) that are required to be resolved, and the plan to form a dialogue act, respectively.

The IU state graph in the DM is altered depending on the revision strategy required. For the more complex revision strategy, the following steps are taken:

1 Handle revoked input by computing a new state of the DM's IU graph, removing *DiscourseIU*s that were *grounded in* the revoked input.

2 Check the DM's own output to determine whether projected *DialogueActIU*s that are *grounded in* revoked input have been realised into observable output.

3 If step (2) is found true, initiate explicit repair.

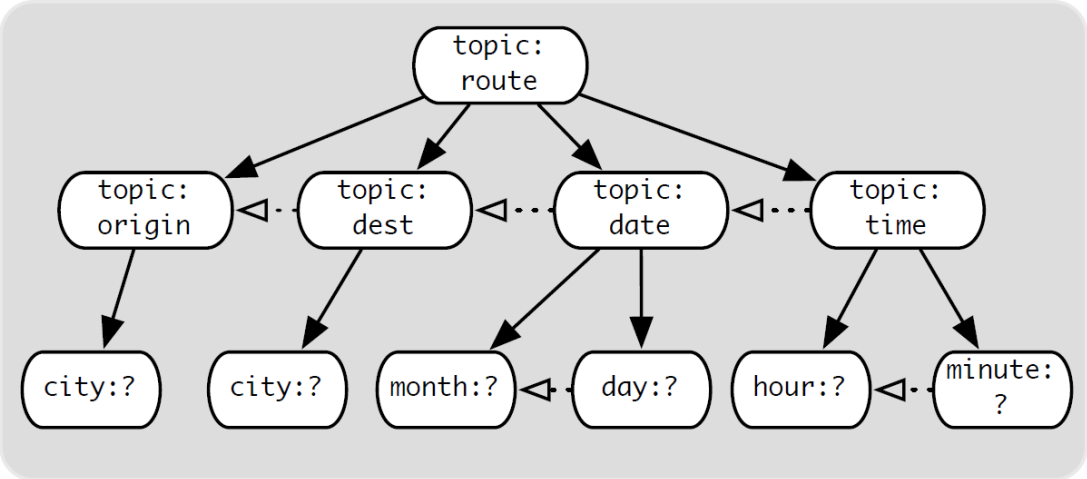


Figure 3.7: The DIUM dialogue management framework as an IU network (Buß and Schlangen, 2011)

In the final step, if it is reached, the fact a repair has been initiated is recorded through use of a novel type of *DialogueActIU* called *UNDO*. By creation of a specific repair IU this can be interpreted by down-stream realisation modules to effect the appropriate repair behaviour. The authors only suggest a preliminary strategy, one of generating an apology 'sorry about that' whilst un-highlighting anything in the visual domain that was highlighted in the previous conflicting state.

While the repair generation is limited, and not psychologically motivated, DIUM was a useful step towards incremental generation of self-repairs in an interactive system. While it does not have an NLU component capable of interpreting repairs from the user, it began to address the issue of how repair acts could be represented in a dialogue information state. A more thorough attempt at addressing this issue, albeit not one implemented in a working dialogue system, is described in the next section.

3.3.2 KoS: Dialogue semantics of disfluency in an Information State Update approach

In terms of the dialogue semantics and the interpretation of self-repair, there has only been a formal treatment relatively recently in dialogue research. The majority of work on self-repair processing, of which the above is intended to be representative, is structure-oriented, intended for practical speech applications that are not required to interpret such events by detecting them

or generating appropriate output behaviour; however modelling the *meaning* of self-repair and disfluency as construed by conversation participants (CPs) engaged in interaction is not only important for cognitive modelling (see Chapter 2), but also, as I will argue in this thesis, for more efficient and realistic dialogue systems. What follows is a brief overview of the recent interest in the dialogue semantics of disfluencies by Ginzburg and colleagues (Ginzburg et al., 2007; Ginzburg, 2012; Ginzburg et al., 2014) within the KoS dialogue framework (Ginzburg, 2012).

At the time of writing, the most recent KoS account of disfluency (Ginzburg et al., 2014) models several types of disfluency events mentioned above (particularly in line with the typology described in Section 2.5, and also attempts to unify self-repair and other-initiated repair in terms of dialogue state update mechanisms, while not conflating the two phenomena in terms of their dialogue semantics.

The authors divide disfluencies into two types: (1) *forward-looking* disfluencies, which include unfilled pauses (hesitations), filled pauses and discourse markers followed by fluent continuations (termed *covert* repairs in Section 3.3.1, or *abridged repairs* elsewhere (Heeman and Allen, 1999), Section 3.1.1) and also verbatim repeats; and (2) *backward-looking* disfluencies, which are disfluencies with a repair phase (called the 'alteration' in their work) that refers back to a reparandum, which include 'repairs' where the alteration seems to replace the reparandum such as "[We were + I was] lucky too that I only have one brother" (ibid., p. 3) and 'reformulations' where the reparandum is elaborated on, such as "at that point, [it, + the warehouse] was over across the road" (ibid., p. 4), the latter example again being a good case for preserving the semantic content of the reparandum. In terms of surface form, these types of backward-looking disfluencies are described in terms of the self-repair structure described by Levelt (1983, 1989), Clark (1996) and most precisely by Shriberg (1994) as explained in Chapter 2, however they also distinguish *fresh starts* as another category within backward-looking disfluencies (e.g. the repair phase initiating at "there" from the second interruption point + in the utterance "{ I mean } [[I, + I,] +] [there are a lot, + there are so many] songs"), which is where the alteration differs strongly from the reparandum, not exhibiting the structural coherence of repairs or reformulations.

The authors claim the desiderata for a theory of disfluencies should include the following:

1. Disfluencies are recognized immediately, for which information about their meaning is required.

2. Disfluencies have immediate discourse effects.

3. Disfluencies are related to other dialogue moves.

4. Disfluencies are in the grammar.

To begin addressing these desiderata, they show how a minimal extension to the KoS framework can achieve the incrementality required for the immediate integration of disfluent sub-utterances into the current dialogue context. I argue that the detailed requirements of a solution to the final desideratum 4 are not fully met (the authors themselves defer this challenge to future work), however the other three are addressed. I will now discuss a brief overview of KoS in order to explain the extensions to its architecture proposed to accommodate disfluency according to these three requirements.

KoS is a dialogue framework which models conversational participants (CPs) as having access to a current context, or information state, at each point in time in a dialogue, affording them the use of their own personal context and also the shared interactive context of the ongoing conversation available to all CPs, in order to make inferences and perform conversational actions. To achieve this, KoS posits a two-part information state structure for each CP: a *private* information state, and the public (or what a CP believes is public) information state called the *Dialogue Game Board* (DGB). It is the latter which is used to deal with disfluencies and their relationship to other phenomena, so an exposition of the private component is not necessary here.[6]

The DGB is represented formally within Type Theory with Records (TTR) (Cooper, 2005), a richly typed type theory that will be used later in this thesis, albeit to different technical ends. TTR (Betarte and Tasistro, 1998; Cooper, 2005) is a rich type theory which has become widely used in dialogue models, including information state models for a variety of phenomena such as clarification requests (Ginzburg, 2012; Cooper, 2012) and non-sentential fragments (Fernández, 2006). It has also been shown to be useful for incremental semantic parsing (Purver et al., 2011) and recently for grammar induction (Eshghi et al., 2013).

While I defer a full technical explanation of TTR to subsequent formal analysis chapters, at this point it is appropriate to introduce its important elements: most importantly, the fact that the central judgement in type theory $s : T$ (that a given object labelled s is of type T) is extended in TTR so that s can be a label for a (potentially complex) *record* object and T can be a *record type*, where record types can be inhabited by records in the same way as simple types can be inhabited

[6] See Larsson (2002)'s thesis for a thorough treatment of the difference between private and shared information states, and the interaction between them during task-oriented dialogues.

by simple witnesses – in the case of KoS, typically, *s* could be a label for a conversational state and *T* could be a conversational state type (Ginzburg, 2012; Cooper, 2012), or *s* could be an utterance and *T* an utterance type. Establishing that *s* is a witness for *T* by judging that $s : T$ is true can be seen as a classification of a situation, which is the central action in natural language understanding in the view the authors present: type classification is used for judging phonetic, syntactic, semantic and discourse-level situations (and situations which involve combinations of these) as pertaining to types of dialogue situation on various levels- e.g. if an utterance event is correctly classified by a type generated by the dialogue grammar, the utterance can be thought of as being correctly parsed or interpreted.

Technically, record types can be viewed as partially ordered sets of *fields*, which are individual type judgements of the standard form $s : T$, allowing them to be represented graphically as attribute-value matrices such as in (3.8) representative of the DGBtype. For the purposes of discussion here, I take all possible conversational states to be records of type DGBType:

$$DGBType \equiv \begin{bmatrix} \text{spkr} & : & \text{Ind} \\ \text{addr} & : & \text{Ind} \\ \text{utt-time} & : & \text{Time} \\ \text{c-utt} & : & \text{addressing(spkr,addr,utt-time)} \\ \text{Facts} & : & \text{Set(Proposition)} \\ \text{Pending} & : & \text{list(locutionary Proposition)} \\ \text{Moves} & : & \text{list(locutionary Proposition)} \\ \text{QUD} & : & \text{poset(Question)} \end{bmatrix} \tag{3.8}$$

The role of the relevant fields of the DGBType in terms of their function in dialogue interaction are explained briefly below:[7]

- **spkr,addr**: the speaker and addressee for the current turn.
- **Facts**: set of commonly agreed upon facts, i.e. the shared knowledge conversational participants utilize during a conversation- more operationally, this amounts to the information $x \in X$ that a CP can use embedded under the presuppositional operators ". . .Given that *x*" and "Since we know that *x*".
- **Pending**: a list of ungrounded locutionary propositions (abbreviated 'LocProp'- propositions encoded as TTR records that individuated by an utterance event and a grammatical type that classifies that event).

[7] The utterance timing and turn taking status fields utt-time and c-utt can be understood as implicitly being present in the DGBTypes I show from here on; see (Ginzburg, 2012) for details

- **Moves**: a list of grounded utterances (also encoded as LocProp records).[8] To focus on the most recent move, **LatestMove** is a field in a DGBType often used to distinguish the final element of *Moves*.

- **QUD**: ('questions under discussion') : a partially ordered set (poset) that specifies the currently discussed questions. Each question in QUD constitutes a 'live issue' rather than necessarily a question that has been posed verbally, that is to say an issue introduced for discussion not yet downdated by resolution or abandonment. A query *q* updates QUD with *q*, whereas an assertion *p* updates QUD with *p*? (= whether p?) and a question being maximal in QUD ('MaxQUD') corresponds to the current topic under discussion. The way in which the ordering on QUD changes as new questions are added often resembles the behaviour of a FIFO (first-in-first-out) stack, due to the nature of embedded questions– questions about/relevant to existing questions are dealt with before or in tandem with the original questions on which they are dependent– however, QUD is not constrained to behave only in this way.

KoS's division of propositional content between Facts, Pending, Moves and QUD creates a neat compartmentalisation of the potentially large amount of information available to CPs. The functional division allows *conversational rules* to be formulated as mappings of the form $DGBType \rightarrow DGBType$ which can be specified to operate on specific components. Like the DGB itself, conversational rules can be encoded in TTR as a record type, in this case one with two fields with embedded record types as their types (values), one representing the *preconditions* which is the type constraint that permits the rule to be applied to the current DGB state (via a *supertype* relation or subsumption check against the current conversational state DGB record) and the *effects* which represent the changes to the DGB referenced in the preconditions, stipulated to hold upon application of the rule. The authors abbreviate the full form of these two embedded record types to only represent the elements of the DGB that change, omitting the elements of the DGB that remain invariant;[9] I will follow this convention here. The conversational rule record type can be represented as in (3.9) and an example definition of a specific conversational rule

[8] Ginzburg (2012) describes how it is empirically important to maintain the structure of the utterances grounded rather than just store their semantic content. This is is an important point which I will develop in Chapter 6.

[9] Technically the 'carrying over' of the entire DGB from the state before the application to the one after it is achieved by a function that returns a merge operation of the current DGBType with the effects DGBType, however, this is not important here.

type is given in (3.10) in the abbreviated form mentioned above.

(3.9) $\begin{bmatrix} \text{pre} & : & \text{PreCondSpec} \\ \text{effects} & : & \text{ChangeCondSpec} \end{bmatrix}$

(3.10) *Ask QUD-Incrementation* $\equiv$

$$\begin{bmatrix} \text{pre} & : & \begin{bmatrix} \text{q} & : & \text{Question} \\ \text{LatestMove.sit.content = Ask(spkr,addr,q)} & : & \text{IllocProp} \end{bmatrix} \\ \text{effects} & : & \begin{bmatrix} \text{QUD} = \langle \text{q, pre.QUD} \rangle & : & \text{poset(Question)} \end{bmatrix} \end{bmatrix}$$

As can be seen in (3.10), the type constraint on LatestMove in the preconditions of *Ask QUD-Incrementation* is not on the entire LatestMove LocProp but on a specific component of it, namely the embedded content ('cont') field of its situation ('sit') field, this is given that the LocProp objects in the Moves and Pending lists are records of the structure in (3.11).

(3.11) LocProp = $\begin{bmatrix} \text{sit} = u & : & f \\ \text{sit-type} = T_u & : & f \end{bmatrix}$

T_u has the form of a record type with fields representing various fields pertaining to the current utterance's context, including the field cont, which classifies the current utterance as being of a given IllocProp type, such as Ask(spkr,addr,q). Operationally, the rule *Ask QUD-Incrementation* in (3.10), when the type constraints in its pre field are satisfied, namely that the LatestMove (utterance) has been classified as having Ask(spkr,addr,q) as its illocutionary content, pushes a question q onto QUD, making it QUD-Maximal.

Using the KoS framework, to meet the desiderata for disfluency the authors give an account which is a variation on that which they use for other-repair clarification requests. The principle behind KoS is that in the aftermath of an utterance a variety of questions can be asked of it which are available to the addressee of the utterance, which can be potential update functions to context as just shown. The nature of these questions can range from the low-level attention ones (did the speaker speak?) to phononological recognition (did the speaker say word w?) to higher-level semantic inference (what did the speaker mean by utterance u?). In the incremental version of KoS they present, these questions can be predictive and posed after each word, for instance the issue 'what will the speaker say next after w?'. These planning questions can be pushed on to QUD (made QUD maximal) and both dialogue participants are aware of them. Given this, they define backward-looking disfluency and forward-looking disfluency update rules from the hearer's perspective as resolving these questions on a word-by-word basis. The question $\lambda x MeanNextUtt(pre.spkr, pre.u0, x)$ where $pre.spkr$ is the current speaker, $pre.u0$ is some utterance or word in context and x the propositional content they wish to contribute next can be

posed on a word-by-word basis. This gives the bases for the forward looking rule which may fire on recognition of an edit term or repeat as in (3.12).[10]

(3.12) *Forward Looking Utterance Rule* ≡

$$\begin{bmatrix} \text{pre} & : & \begin{bmatrix} \text{spkr} & : & \text{Ind} \\ \text{addr} & : & \text{Ind} \\ u0 & : & \text{LocProp} \\ \text{Pending} = \langle u0 \ldots \rangle & : & \text{list(LocProp)} \\ \text{LatestMove} = \text{FLDEdit(spkr,}u0\text{)} & : & \text{IllocProp} \end{bmatrix} \\ \text{effects} & : & \begin{bmatrix} \text{q} & : & \lambda x \mathit{MeanNextUtt}(\mathit{pre.spkr}, \mathit{pre.u0}, x) \\ \text{QUD} = \langle \text{q, pre.QUD} \rangle & : & \text{poset(Question)} \end{bmatrix} \end{bmatrix}$$

They show it is possible to model "Show flights arriving in uh Boston" (Shriberg, 1994) by means of a move being recognized upon the production of 'uh' as FLDEdit(A,B,'in'), i.e. A is having a forward-looking problem after the word 'in'. This triggers the Forward Looking Utterance rule (3.12) such that $\lambda x \mathit{MeanNextUtt}(A, {}'\mathit{in}', x)$ is pushed on to QUD to become QUD-maximal. "Boston" can then be interpreted as answering this question.

For backward-looking disfluencies, the key semantic question created by them is not about what follows, but what the speaker *meant* by some previous content $u0$: this can be encoded as the question $\lambda x \mathit{Mean}(\mathit{pre.spkr}, \mathit{pre.u0}, x)$. This allows a backward looking appropriateness repair rule to be formulated as in (3.13) where this question is pushed on to QUD.

(3.13) *Backward Looking Appropriateness Repair* ≡

$$\begin{bmatrix} \text{pre} & : & \begin{bmatrix} \text{spkr} & : & \text{Ind} \\ \text{addr} & : & \text{Ind} \\ u0 & : & \text{LocProp} \\ \text{Pending} = \langle u0 \ldots \rangle & : & \text{list(LocProp)} \end{bmatrix} \\ \text{effects} & : & \begin{bmatrix} \text{q} & : & \lambda x \mathit{Mean}(\mathit{pre.spkr}, \mathit{pre.u0}, x) \\ \text{QUD} = \langle \text{q, pre.QUD} \rangle & : & \text{poset(Question)} \end{bmatrix} \end{bmatrix}$$

This means the example "take that book [in, + { I mean } from] the shelf" can be modelled. After "take that book in", the authors claim the Backwards Looking Appropriateness Repair rule (3.13) can be licensed allowing $\lambda x \mathit{Mean}(A, {}'\mathit{in}', x)$ to be pushed on to QUD and "I mean from" resolves the question.

The authors draw the parallels between self-initiated editing phrases (interregna) and clarification requests (CRs) as cues for repair, extending interregna not just to discourse markers and edit terms as discussed above, but also to self-posed questions. They do this by making an adjustment to KoS in allowing CRs and editing signals and their following corrections to occur mid-utterance, accommodating incrementality by allowing the DGB word-by-word updates to its

[10] These rules are slightly simplified here for illustration.

PENDING component. They also show the potential for fine-grained meaning in dialogue that TTR can provide on a word-by-word basis, an approach which is explored in Chapter 6. However, while this begins to provide a general dialogue model and one which takes the semantic update functions of self-repairs very seriously, the relationship of these updates to incremental parsing and generation processes is not made explicit and they "defer to future work the important tasks of specifying a grammar that can incorporate incremental parsing and interpretation of disfluency-containing utterances and the identification of reparanda." (ibid., p 15)

3.3.3 Dynamic Syntax (DS), DS-TTR and DyLan: incremental semantic construction for dialogue processing

The above incremental generation and dialogue frameworks and systems have variously developed incremental syntactic construction during generation (Kempen and Hoenkamp, 1987; De Smedt, 1990), incrementally changing inputs to generation (Guhe, 2007; Skantze and Hjalmarsson, 2010) or else incrementally updating discourse semantics (Ginzburg et al., 2014), however they do not detail *how semantic content is built up incrementally*, or at least how partial structures in parsing and generation can be converted into maximal semantic content in real time. To facilitate this capability, an incremental grammar is needed that not only has the quality of reversibility, but also has the ability to generate semantic states that can be reasoned with by other modules during generation and interpretation processes. In this section the background and motivation to the formal tools used in this thesis are given, which are then re-purposed and extended in Chapters 6 and 7.

Strong incremental interpretation and incremental representation

The psycholinguistic evidence reviewed in Section 2.4.2, particularly the evidence on restarting, retracing and reformulating (Healey et al., 2011), suggests models of human incremental production and understanding, and indeed self-repair processing within these, would need the capacity to construct the maximal amount of semantic information one at least a fine-grained a level as word-by-word. A formal grammar that could be used for such modelling purposes in a dialogue system, as Neumann (1998) showed, would also need the quality of reversibility in that representation available in interpretation should be available for generation too.

A theoretical distinction that nicely brings out this kind of incrementality is explained by Milward (1991), who points out the difference between a linguistic system's capacity for *strong in-*

cremental interpretation and its ability to access and produce *incremental representation*. Strong incremental interpretation is defined as a system's ability to extract the maximal amount of information possible from an unfinished utterance as it is being produced, particularly the semantic dependencies of the informational content (e.g. a representation such as $\lambda x.like'(john', x)$ should be available after parsing "John likes"). Incremental representation, on the other hand, is defined as a semantic representation being available for each substring of an utterance, but not necessarily including all possible information such as semantic dependencies (e.g. having a representation such as $john'$ attributed to "John" and $\lambda y.\lambda x.like'(y,x)$ attributed to "likes" after processing "John likes"). While strong incremental interpretation is more obviously required for an adequate account of the semantics of dialogue and self-repair, the incremental representation requirement becomes stronger once we look at the time-critical mechanisms of dialogue processes required, where access to information as to *how* the incremental information was constructed becomes essential.

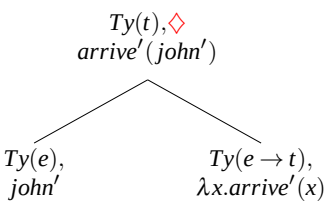
DS grammar and parsing formalism

A formalism presented here which begins to satisfy the incremental criteria just described is Dynamic Syntax (DS Kempson et al., 2001; Cann et al., 2005, *inter alia*), an action-based and semantically oriented incremental grammar formalism that defines grammaticality as parsability. The DS lexicon comprises *lexical actions* keyed to words, and also a set of globally applicable *computational actions* (equivalent to syntactic rules), both of which constitute packages of monotonic update operations on semantic trees, and take the form of IF-THEN action-like structures. In traditional DS notation, the lexical action corresponding to the word *John* has the preconditions and update operations in (3.14).

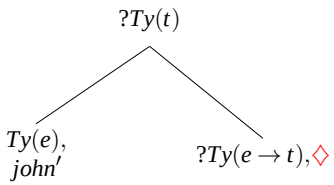
(3.14)

IF	$?Ty(e)$
THEN	$\texttt{put}(Ty(e))$
	$\texttt{put}(Fo(john'))$
ELSE	$\texttt{abort}$

(3.15)



(3.16)



In DS parsing, the semantic trees upon which the actions operate represent terms in the typed lambda calculus, with mother-daughter node relations corresponding to semantic predicate-argument structure, with no independent layer of syntax represented- see (3.15) above. For this reason, from an interpretation point of view, there is no need to augment syntactic structures with semantics due to a representation of semantic dependency being available directly from parsing. Beginning with an axiom tree with a single node of requirement type $?Ty(t)$, parsing intersperses the testing and application of both lexical actions triggered by input words and the execution of permissible (Kleene* iterated) sequences of computational actions, with their updates monotonically constructing the tree. The pointer object (◇) indicates the node under checking and development.

Successful parses are sequences of action applications that lead to a tree which is complete (i.e. has no outstanding type requirements ($?Ty(..)$) on any node, and has type $Ty(t)$ at its root node as in (3.15)) with a compiled semantic formula. Incomplete *partial* structures such as (3.16) above are also maintained in the parse state as words are scanned in the input, so they are still available to be updated upon the next input word string: this is central to the principles at the heart of DS: *underspecification* and *update*. While in the standard DS parsing model there is no compiled single formula for incomplete trees, an issue addressed in Chapter 6, the semantic dependencies between formulae indicated by position in the tree allow a maximal amount of semantic information to become available incrementally in the parsing process, allowing for strong incremental interpretation as well as incremental representation.

DS generation as parsing and goal subsumption checking

As Purver and colleagues (Otsuka and Purver, 2003; Purver and Otsuka, 2003; Purver and Kempson, 2004) demonstrate, a model of tactical DS generation can be defined neatly in terms of the DS parsing process and a subsumption check against a *goal tree*. The input goal tree is defined as a complete and fully specified DS tree such as (3.15), and the generation of each word consists of attempts to parse each word in the lexicon given the trees under construction in the parse state. For successful lexical action applications, a *subsumption* check is carried out to make sure there are no nodes in the trees under construction that contain decorations absent in the goal tree, and

failed trees and their parse paths are removed from the parse state- see Figure 3.8 for a comparison of parsing and generation in DS. Otsuka and Purver (2003)'s initial context-independent generation model uses a set of DS *parser states* to characterize a generator state, where a parser state is a pair of the word input consumed so far and a set of associated partial trees.

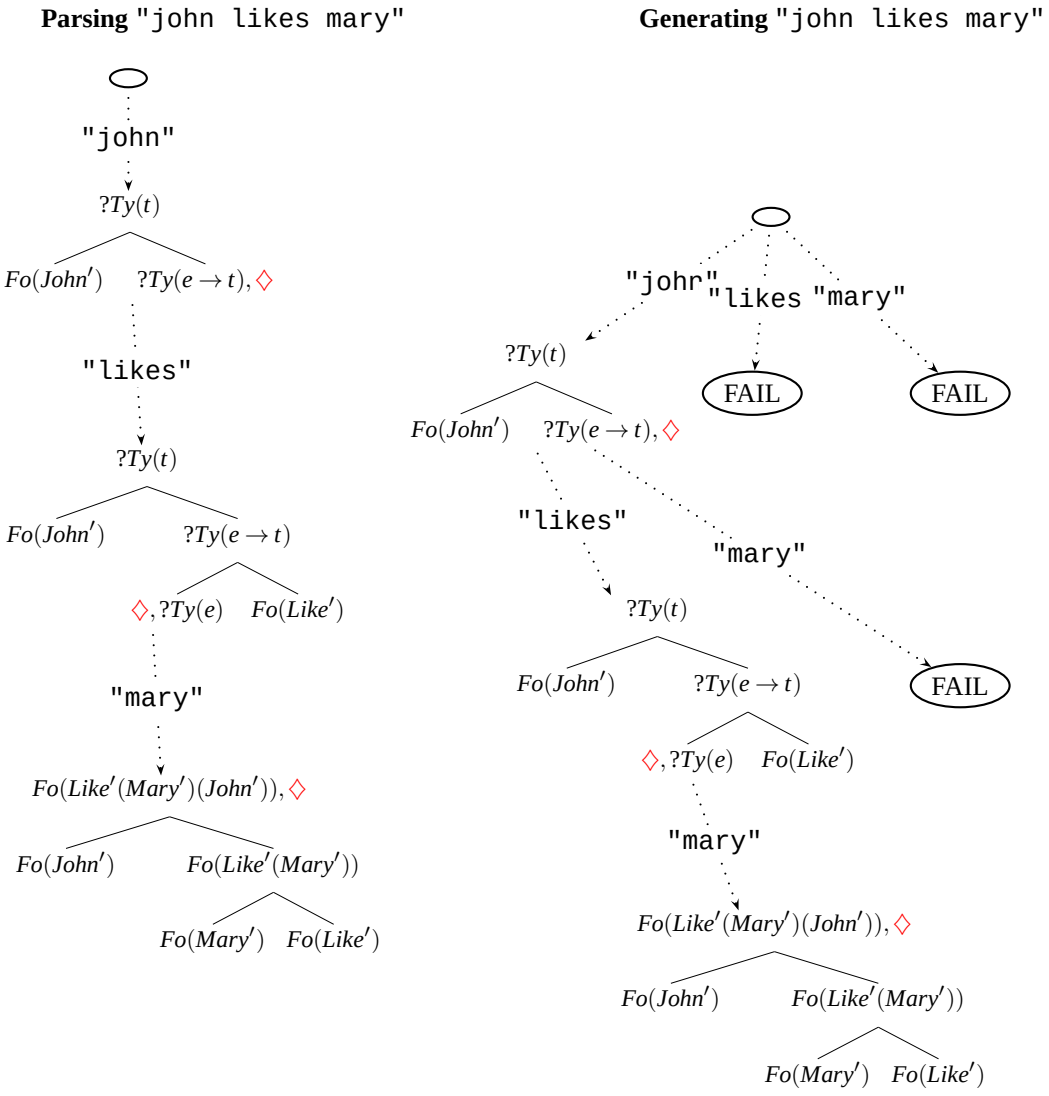


Figure 3.8: Parsing/generating "john likes mary" from Purver and Kempson (2004)

As DS generation proceeds by a 'parse and test' method, where all words in the lexicon accepted by each parse state are used to extend that parse state (i.e. they develop the trees under construction), the subsumption test is carried out to return only those parse paths with trees that subsume the goal tree. There is formal elegance in the model from an NLG perspective in that lexicalisation and linearization (or in psycholinguistic terms, formulation and word ordering) are not separate processes: each word in the lexicon is tested for its applicability at each point of

possible tree extension, and if accepted by the generator it is both selected and realized in the output string in one single action.

In terms of overall efficiency, despite the incremental testing of each word in the lexicon, the DS generation protocol does not need to generate all possible strings before testing, nor does it require case-by-case specific syntactic versions for lexical items dependent on their position in the utterance being generated. Otsuka and Purver (2003) point out that given a lexicon size of L and number of words in the output string W, the number of possible paths is not in fact L^W, but to closer to $N \times L$. This is because the parse of each candidate word is constrained by the trees in a generator state that is pruned of all trees incompatible with the goal tree on a word-by-word basis. However, the problem of computational efficiency for sizeable lexica still remains and the authors suggest some possible heuristics to overcome this using goal tree features. Additionally, different search algorithms for generation are suggested, with a depth-first search returning only the first suitable string found, abandoning other possible parse paths, and a breadth-first search returning all the possible candidate words that could follow sub-strings in the generator state. Work needs to be addressed as to how this could be defined more concretely computationally, allowing different search strategies and also taking psycholinguistic plausibility into consideration; this is something that will be investigated later in this thesis.

While a functional model of self-repair is not proposed in the DS literature, there is a self-monitoring facility inherently present in the generation procedure. The strings generated are selected through parsing, so there is no need for a feed-back loop to the parser as proposed in Levelt (1989)'s psychological model and featured in (Neumann, 1998)'s reversible system; in DS,"self monitoring comes built-in, as parsing is the building block for generation" (Otsuka and Purver, 2003, p. 98). This nice facility in the DS framework will be used for modelling the self-monitoring in self-repair in later chapters.

Context models in DS for incremental dialogue modelling

As stated above, similarly to Neumann (1998)'s framework, DS can be characterized as a reversible grammar, as the input for generating a string is the semantic tree that would be derived from parsing that string. However, there is an additional component in DS parsing and generation that can be obtained from its grammar apart from the LFs and strings, which is the maintenance of the procedures that are employed to construct trees. This is possible due to the fact that actions are first class citizens of the system. Making use of this feature of the grammar, Purver and

Kempson (2004) describe how a thorough-going notion of context in generation and parsing can be explicitly expressed in terms of the incremental development of a set of *parser tuples*, each of which constitute a triple $< T, W, A >$ of a tree (T), a word-sequence (W) and the sequence of actions (A), both lexical and computational, that are employed to construct the trees. This characterization of context can be used to account for several speech production and dialogue phenomena.

For modelling purposes Purver and Kempson (2004) use a simple implementation whereby context is limited to spanning back to the beginning of the immediately previous utterance (one which has yielded a complete parse, roughly equivalent to a sentence). At the beginning of the parse of the second sentence, the parser's context consists of both a parser tuple $< T_0, W_0, A_0 >$ of the previously parsed sentence and another tuple $< T, W, A >$, where T is the initialized axiom and W and A are empty word and action sequences.[11] The modification of the parser state to a generator state is minimal, as it is characterized as a pair of a goal tree G and a set X of partial string(S)-parser state(P) pairs (where P is a set of $< T, W, A >$ tuples, initially with only one member, as per parsing).

The authors use this detailed context apparatus for modelling several production and dialogue phenomena:

- *Anaphora and Ellipsis* The generation of strict readings of verb phrase ellipsis (VPE) such as "John likes his donkey and Bill does too" $\rightarrow$ *Bill likes John's donkey* is achieved through normal generation, but the lexical action for the auxiliary 'does' allows tree nodes to be decorated with *metavariables*, which are underspecified semantics which recover suitable tree decorations (in verb phrase ellipsis that is a $Ty(e \rightarrow t)$ complete node with its formula from context). Anaphors work similarly, but with $Ty(e)$ metavariables being used instead. *Wh*-questions also project metavariables which can be updated so a fragment may be generated to complete a tree in the generator context.

  Sloppy VPE readings, where "John likes his donkey and Bill does too" $\rightarrow$ *Bill likes his own donkey*, the strategy is different, as the semantic formula at the top of the tree drives the re-running of *computational* actions in context as opposed to re-using semantic formulae.[12] While this is implementationally perspicuous for parsing ellipsis, the generation of

[11] This second tuple is the single inhabitant of the parser's state before any actions are triggered, which, after multiple triples being formed through the parsing process, will eventually result in returning a single tuple $< T_1, W_1, A_1 >$.

[12] A formal implementational example action of REGENERATION is given in Purver et al. (2006) and

such structures is not fully detailed– some additional mechanisms would be needed for a generator to chose an elliptical phrase over a non-elliptical semantically equivalent one.

- *Minimizing lexicon search* While generation details are not fully detailed, DS authors claim the computationally expensive 'parse and test' task at each stage of generation can be reduced in complexity by using the immediate context to search for appropriate lexical actions. In the case of ellipsis and anaphora resolution the action sequences required to complete trees are in the immediate context (in the same utterance or the one previous), and the authors point out the prevalence of these phenomena in dialogue may be due to this efficiency benefit, and *alignment* benefits for interlocutors as postulated by Pickering and Garrod (2004).

- *Alignment and Routinization* The recoverability and re-use of actions allows several psycholinguistic phenomena to be modelled with the storage of actions in context, and this not only applies to within speaker affects, with ellipsis and anaphora, but also between speaker re-use of actions, resulting in lexical, semantic and syntactic alignment (Pickering and Garrod, 2004).

- *Shared utterances (compound contributions)* In cases where it can be argued that speaker and hearer add to the same semantic structure despite a speaker switch mid-utterance, this can be modeled in the DS context model easily due to the close connectivity of the parsing and generation processes. In terms of the generation model, in generating the completion of an utterance started by the interlocutor, given a dialogue system capable of such deduction, in a situation where the goal tree G is deduced and an initiation to vocalize is signalled by the dialogue manager,[13] G can be set as the goal tree, with the parser state replacing the generator's parser state (i.e. the P in $< G,\{S,P\}>$). The testing of lexical and computational actions will then commence on the tree under construction from the parser state and successful applications to the tree will result in the words completing the utterance and subsuming the goal tree. As can be seen in Figure 3.8, the trees built up are the same in parsing and generation.

Kempson et al. (forthcoming).

[13]The generation of the goal tree itself (the conceptualisation stage, most likely completed by a dialogue manager) is not modelled in DS as the account is tactical rather than strategic. Poesio and Rieser (2010) show how a formal dialogue model can account for the completion of another dialogue participant's utterance from mechanisms on the strategic level, in their account through inferring the intended plan of the speaker.

- *Transition from speaker to hearer* Modelling speaker change transitions again falls out of fact that the parse states in generation and in parsing are directly interchangable. So, as a hearer of a continuation in a shared utterance rather than the speaker, it is the transfer of *P* to the parser that makes the integration of the two parts of the utterance possible through incremental semantic construction. The possible mismatches between the parse states of different interlocutors are not addressed in this model, but clearly this needs attention for a realistic model.

DyLan: DS context as a graph in the IU framework

The procedural context of DS parsing is made more explicit in its integration into the dialogue system Jindigo (Skantze and Hjalmarsson, 2010, see Section 3.3.1 above) within the `DyLan`[14] NLU module (Purver et al., 2011), where internally a parse state is characterized as a Directed Acyclic Graph (DAG), following Sato (2011), with DS *actions* for edges and *trees* for nodes. This move gives the DS framework a more fine-grained incremental representation than was previously possible with the original model of context. The characterization also allows an exploitation of Jindigo's graph-based buffers, particularly the interface with word graphs sent from a voice recognition (ASR) module: `DyLan` incrementally attempts to parse word hypothesis edges as they become available, and parse paths in the DAG are *grounded in* the corresponding word edges of the ASR graph in Figure 3.9 there is only one word hypothesis, 'john', but in reality there could be several candidates, each spawning their own corresponding part of the parse DAG.

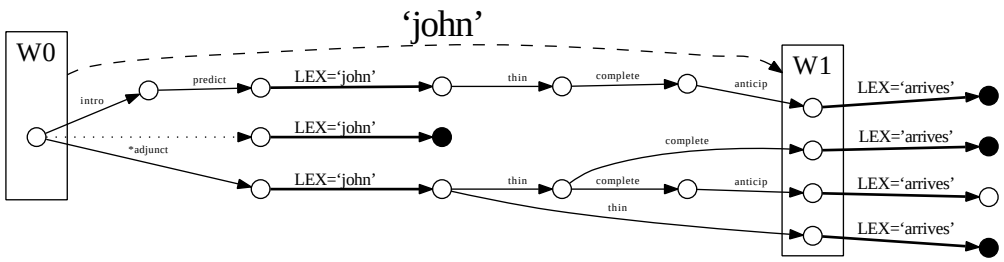
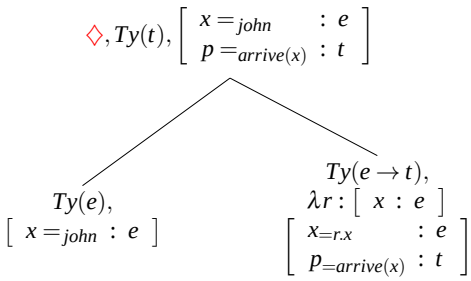


Figure 3.9: DS context as a DAG, consisting of parse DAG (circular nodes=trees, solid edges=lexical(bold) and computational actions) with overarching corresponding word graph (rectangular nodes=tree sets, dotted edges=word hypotheses) with word hypothesis 'john' spanning tree sets W0 and W1.

[14] 'DYnamics of LANguage'.

The other addition to DS in DyLan is the incorporation of TTR, which can be seen in (3.17). TTR record types decorate the nodes of the tree as opposed to simple atomic formulae, and following Cooper (2005), each field in the record type contains a variable name, a value (after the =), which can be null for *unmanifest* fields, and a type (after the colon) which represents the node type of the DS tree at which the formula is situated if it is a simple type, or else the node at which β-reduction will give its result (e.g. type t for a predicate at a $Ty(e \rightarrow t)$ node).

(3.17) *"John arrived"* $\longmapsto$



The DS-TTR adaptation is made to provide representations that can interface with domain conceptual structures. For type complete trees, through functional application of functor nodes to argument nodes beginning with the right corner node, root nodes of complete trees are decorated with a compiled TTR record type, which can be checked against specified TTR formulae representing system domain concepts. The integration of TTR allows more fine-grained lexical and pragmatic information to be represented on the tree, as well as providing representations that can easily interface with domain concepts in information-state motivated dialogue systems. The theoretical efficacy of the approach for incorporating locutionary and dialogue participant information into DS processes shown in Purver et al. (2010) in the modelling of *compound contributions* (see Howes et al., 2011, for details of the empirical phenomenon). Various extensions of DyLan are required for it, as a model and dialogue system, to be able to deal with self-repair as will be discussed in Chapter 6. DyLan and the DS-TTR parser are well positioned as a point of departure and at the time of writing, only the incremental RMRS (Robust Minimal Recursion Semantics) parser described by Peldszus et al. (2012) has competitive incremental semantic construction capabilities.

3.4 Evaluation of incremental dialogue processors and their self-repair capabilities

Baumann et al. (2011) perspicuously formulate the challenge evaluating the *incremental* aspects of a processor, which amounts to more than simply comparing its utterance or input-final results

with a non-incremental counterpart:

> "What needs to be measured is *what happens when*, as simply comparing the final results of incremental and non-incremental settings is not enough." (Baumann et al., 2011, p.115)

While I will not go into full detail here as the metrics will be explained in the analysis chapters when needed, they divide their metrics into *similarity*, *timing* and *diachronic* measures. It is the last kind which is the most novel, that is, the evolution of incremental hypothesis over time, for which measures such as Edit Overhead (see Baumann et al. (2011)) were devised.

In terms of self-repair evaluation, the only incremental metrics described in the literature are the *incremental accuracy* and *time-to-detection* scores discussed in Zwarts et al. (2010). All other approaches only evaluate reparandum word detection, and not the whole structure of the repair. These limited metrics are currently not sufficient for the purposes of incremental dialogue systems, and this will be addressed in Chapter 5.

3.5 Summary and directions for research

3.5.1 Detection and classification

While Heeman and Allen (1999)'s approach is left-to-right word-by-word incremental due to its integration within n-gram language models, its success rate (with a caveat for non-comparability to other systems) for resolving and correcting speech repairs appears to be considerably lower than other approaches, possibly due to sparsity of repair template examples. Liu et al. (2003)'s model shows how explicit rules about repetition forms can help resolve the interruption point and reparandum onset better than statistical models alone, but this only extended to simple repeats. The problem with data-driven machine-learning approaches is the sparsity of data for rarer forms of self-repair, which suggest a knowledge-rich rule based system should be incorporated.

Johnson and Charniak (2004)'s TAG-based Noisy-Channel model uses explicit rules, and shows how this can be incorporated as a pre-parsing step that uses other parsing technology to filter out reparandum and interregnum words. However, the model may not be seen as psychologically realistic, given its excising of repaired material. In evaluating their earlier edit-detection system, the authors state they "do not care where the EDITED nodes appear in the tree structure produced by the parser" (Charniak and Johnson, 2001, p.6). The internal structure of the

EDITED sub-trees is not a concern, as the principle task is deletion at the string level. From a dialogue point of view, as discussed in the previous chapter, this is not a satisfactory approach. An interpreter module should have access to the edited syntactic structure, not only for possible future reference to it, but for giving the correct procedural context to aid processing the up-coming sentence (Clark, 1996; Core and Schubert, 1999; Brennan and Schober, 2001). The approach is not incremental in the sense of limiting re-computation and having stable output, as it assigns the repair structure at the end of parsing the utterance (and then consequently re-ranks it).

In the now popular automatic disfluency detection task, while incremental systems exist, they use over-prediction, large chart storage and filtering (Zwarts et al., 2010; Heeman and Allen, 1999). A parsing chart used solely for disfluency structures positing every possible repair path can grow in the order of $O(n^4)$ in the length of the utterance n, when considering all boundary points of the three phases of a repair. Also, Zwarts et al. (2010)'s TAG parser has a run-time complexity of $O(n^5)$. This complexity blow-up seems cognitively implausible, particularly given the relative sparsity of repairs. In addition, these approaches cannot easily deal with processing embedded repairs realistically, as a stack of charts would be required, further increasing complexity– consequently these are ignored in training (Johnson and Charniak, 2004). Rather than positing all possible repair alignments, intuitively, a listener is almost certain an utterance is a non-repair before the repair onset, so a backwards search mechanism employed upon interruption point detection seems more plausible: the corpus study in the next chapter will tell us exactly how plausible. A more strictly incremental detection should improve responsiveness (time-to-detection) too.

For evaluation, this thesis adds the extra stipulation that systems should be evaluated for their *incremental* (at least word-by-word) performance, and not just for their performance on complete utterances. While Heeman and Allen (1999) take the most strictly incremental approach to building their system, their evaluation is for top interpretations of entire utterances, not evaluating the evolution of their output through time. While Rasooli and Tetreault (2014) and Honnibal and Johnson (2014) achieved state-of-the-art utterance final results and operate left-to-right, again no evaluation of incremental performance is given.

Most importantly the latter systems only evaluate on the Penn Treebank EDITED words (reparandum tokens), which by design are annotated to help parsers, making the task rather circular. Evaluating on detecting the entire repair structure as proposed by Shriberg (1994) is the

first step to interpreting the meaning of self-repair automatically. New evaluation metrics to deal with this will be proposed in this thesis.

3.5.2 NLU, NLG and dialogue models

In incremental NLU, there is no established computational model for processing or representing repairs, as only formal dialogue models have been proposed (i.e. Ginzburg et al., 2014). Given this, well defined incremental models of how dialogue participants jointly construct meaning on a word-by-word basis will require an interface to an incremental semantic grammar and semantic inference system that uses it– Chapters 6 and 7 address this. Particularly, the sub-problem of representing *parse states* in line with the empirical data, and representing the context of incremental parsing in such a way as to maximise repair detection is a major concern of this thesis.

Incremental generation has been extensively worked on, and much of the work has been informed by, and indeed informed, cognitive models of production. Several desiderata of incremental generation systems arise from this, include parallelism in formulation, feedback and partial and underspecified inputs. Several of the systems have modelled self-repair explicitly, and (De Smedt, 1990)'s IPF account, Guhe's INC and Skantze and Hjalmarsson's speech generation model make it a more central, rather than peripheral, ability. What they lack however is an incremental semantics that is well defined. Given that speakers can recover words from the reparandum straightforwardly, repaired elements of a string must be as accessible to a parser and generation system as to fluent parts of the utterance, yet simultaneously should receive a different discourse status. No model of generation and dialogue meets these challenges. Chapter 6 describes a formal model built with the premise that the *representation* of the processing context in incremental parsing and generation is an important factor for characterising self-repair formally and meeting these challenges while I return to probabilistic generation in Chapter 7.

The semantics of a repair is clearly one in which order of repair sequences and timing matter. We need to include in any model of dialogue a record of the processing history that CPs are utilising in building up representations such that the difficulties in articulation, formulation and conceptualisation and their resolution are part of this– this is the only way CPs can compute each other's conversational state more accurately. Contra to traditional semantics, and to some extent syntax, order is as important as form. To adhere to Ockham's razor, this additional apparatus must of course be incorporated into a dialogue system in the most efficient way possible.

3.5.3 The desiderata for incrementality in dialogue models

With the different types of incrementality described above at hand, I conclude the chapter by summarizing the different types of incrementality required for models of self-repair and the dialogue models and systems they are housed in, which applies both to automatic tasks and formal models.

One kind of incrementality abounds in NLP (Natural Language Processing, rather than psycholinguistically motivated computational linguistics), which is borne out in parsing and generation algorithms rather than in the grammars they use. This incrementality is the inspiration for chart parsing (Kay, 1973): for example the Cocke-Younger-Kasami (CYK) algorithm is a process that incrementally hypothesises the syntactic structure of a sentence, where partial results of the computation can be stored on a word-by-word basis to maximise efficiency in a dynamic programming chart, and no computation is done more than once. However, the structures exhibited at any point in the parse may not give the full amount of syntactic information available if the grammar formalism is not inherently incremental, and dynamic programming in general is non-monotonic due to the final solution not being directly predictable from the solutions to subproblems. Hale (2001); Hale et al. (2006)'s approach to parsing is similarly motivated. The kind of incrementality these algorithms exhibit is still important however, and is what I will term the *minimization of re-computation* requirement. This is illustrated in Sundaresh and Hudak (1991)'s diagram in Figure 3.10 (rotated here for consistency with other diagrams). The diagram shows how a function f (e.g. a parser) works on some input (e.g. a word string) to output some result, and if only one part of the input from time i changes at time $i+1$, i.e. d goes to d', there would ideally be an avoidance of re-computation of f's application to the other parts of the input a, b, c which have not changed from time i. This desideratum of a dialogue system or model puts requirements on its architecture and algorithms which will be discussed in the coming chapters, and evaluation techniques will be presented to capture this.

Another kind of processing incrementality mentioned above is Wundt's Principle, that each processing component should be triggered into activity by a minimal amount of its characteristic input. As described for psycholinguistically motivated NLG models, this is inspired in part by working memory limitation in human language production: processing is less demanding if information is delivered and received in piecemeal chunks between different modules, particularly in systems where minimal input can cause processing to begin in different modules of the speech

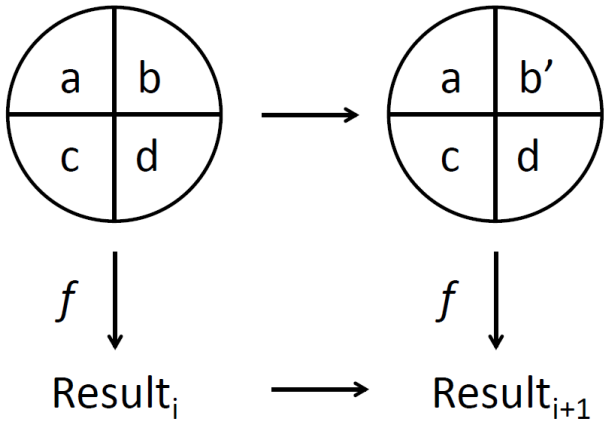


Figure 3.10: Mimimization of re-computation in incremental processing (Sundaresh and Hudak, 1991)

system. The incremental multi-modular production model developed by Levelt (1983), Kempen and Hoenkamp (1987), Neumann (1994) and others has been a powerful one in the *autonomous processing* camp of psycholinguistics. The principle has extended to interactive systems, particularly the IU framework (Schlangen and Skantze, 2009, Section 3.3.1).

Two other complementary types of incrementality mentioned above are strong incremental interpretation and incremental representation (see Section 3.3.3). A system may exhibit one of these types of incrementality but not the other: this is more clear in the case of a system producing incremental representation but not yielding strict incremental interpretation of left-right input: in such a system functional application of $\lambda y.\lambda x.like'(y,x)$ to $john'$ would not necessarily be carried out to give the maximal amount of semantic information possible $\lambda x.like'(john',x)$ despite representations corresponding to each word becoming incrementally available– bottom-up chart parsing may suffer from this. Conversely, in another system the maximal interpretation for a partial utterance may be available incrementally, but if the incoming words add to a context via constraints– for example the incremental updating of a Discourse Representation Structure (DRS, for details see Kamp, 1981)– it may not be possible to determine which word or sequence of words was responsible for which part of the semantic representation without stipulating extra machinery, and therefore the procedural or construction elements of the context may be irretrievable. This recoverability of the update effects caused by given words becomes important in the context of repairing previous update effects, and the revocation of word hypotheses in a dialogue

system situation where the word hypotheses may be unstable as more acoustic information becomes available– see (Schlangen and Skantze, 2011). Both of these types of incrementality are important for dialogue models, and for self-repair in particular.

It is possible to distinguish the level and type of incrementality a formal grammar may exhibit. Less incremental grammars are those which only yield *connected* tree structures at the end of a parse, rather than these being available continually during word-by-word parsing. However, these can be modified with extra-mechanisms to get word-by-word semantic interpretation, such as Milward (1995)'s incremental CCG-driven semantic parser, but extra machinery is needed to achieve this. Similarly, in chart generation (Shieber, 1988; Kay, 1996), incrementality is exhibited in consuming the generation input (a logical form), as partial results are stored and no hypothesis is made twice, fulfilling the minimization of re-computation requirement, however the output is not restricted to being left-right word-by-word incremental in the way the string is realised.

Some grammar formalisms have made incremental representation a focus, as they yield connected structures (partial syntactic trees) from sub-sentential input, such as psycholinguistically-motivated tree-adjoining grammar (PLTAG, Demberg and Keller, 2008), (top-down) probabilistic context-free grammar (PCFG, Roark, 2001), Robust minimal recursion semantices (RMRS) parsing (Peldszus et al., 2012) and Dynamic Syntax (DS, Kempson et al., 2001, see Section 3.3.3). Given the varying degrees of incrementality in both representation and in linguistic mechanisms, a major task befalling incremental models of parsing and generation is to devise the optimal representations for syntax, semantics and dialogue state representation and the best algorithms for translating between these. For example, in interpreting input word-by-word, the key mechanisms can be seen as functions transforming input from layers closer to perception to the higher level more abstract representations on a word-by-word basis: in Figure 3.11 see the functions from syntax $\rightarrow$ semantic representation $\rightarrow$ dialogue state (transitions a and b) and also the time-linear transition function f between these representations. The trade-off between the expressive power of the representations and the minimization of complexity of these transition functions is presumably what motivates the multitude of grammar formalisms, semantic representations, dialogue state representations and the parsing, generation and dialogue management algorithms that have been proposed. The selection of the most natural and efficient choices for representation languages and functions given the challenge posed by self-repair forms a central concern of the

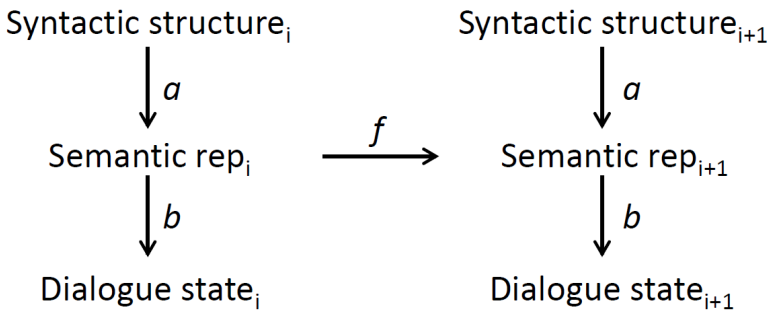


Figure 3.11: Syntax-semantics-dialogue state interface for incremental processing. Adapted from Milward (1995)

formal modelling and computational implementation part of this thesis.

The approach outlined in Chapters 6 and 7 makes use of and modifies a grammar which exhibits desirable incremental properties without needing additional mechanisms in parsing such as those in (Milward, 1995). It achieves this by as far as possible collapsing transitions *a* and *b* in Figure 3.11 into one transition, where a unitary representation can be used. The key ingredients to this are the formalism DS-TTR (Purver et al., 2011; Eshghi et al., 2012), Sato (2011)'s discussion of DS parsing algorithms, (Purver and Kempson, 2004)'s left-right incremental DS surface realizer and the IU framework (Schlangen and Skantze, 2009), which collectively provide the point of departure. Extensions to these formal and implementational tools are proposed to allow the incremental processing of self-repairs within a dialogue model.

In summary, I posit the following incremental desiderata of our interpretation, generation and dialogue management modules, all of which also apply to the self-repair processing capacity as much as to the processing of non-repaired utterances:

- Incremental representation (at least word-by-word): recoverability of update effects to context triggered by words must be total, to allow for revision of input in real-time dialogue models.

- Strong incremental interpretation (at least word-by-word): the maximal amount of semantic information built-up so far as the model consumes its input words must become available immediately.

- Wundt's Principle: processors should be able to begin their work on a minimal charac-

teristic amount of input, e.g. input should be words for parsers to allow strict left-right processing, input should be partial logical forms for generators.

- Efficiency: fast, (if possible synchronous) processing chains between syntax, semantics and dialogue state updates.
- Minimization of re-computation: no computation should be made twice wherever possible as new input is consumed.
- Stability of output through time: minimization of change in output hypotheses as utterances are consumed.

Chapter 4

Empirical Corpus Study of Self-Repair in Dialogue

This chapter[1] describes a corpus study on the Switchboard corpus (Godfrey et al., 1992). I analyse the distribution of self-repair surface form types in dialogue and, motivated by an incremental approach, analyse the ways contextual features of an ongoing utterance predict the repair forms. I present two self-repair ontologies, one based on surface form and the other based on dialogue function, to address lacunae in previous corpus studies.

4.1 Introduction

The aim of the corpus study is to inform the building of two different computational mechanisms: (1) an incremental self-repair detector and classifier which works strictly word-by-word incrementally (2) incremental parsing and generation modules that can interpret and generate self-repair phenomena appropriately.

As described in Section 3.5, the state of the art in these mechanisms falls short in this task in various ways. Approaches have either used statistical language models which do not provide the appropriate constraints on self-repair, and lack interpretable results for building generation systems (e.g. Heeman and Allen, 1999; Qian and Liu, 2013) or else those using more generative models either suffer from data sparsity (Miller and Schuler, 2008) or do not take a strictly incremental approach that is required for integration with on-line dialogue processing (Johnson and Charniak, 2004). The state-of-the-art incremental approach (Zwarts et al., 2010) suffers from latency in detection and delay in computing the analysis as it is a word-by-word version of the in-

[1] This is an extension of Hough and Purver (2013).

herently non-incremental (Johnson and Charniak, 2004) TAG-based noisy channel model: it still requires access to entire utterances to compute global alignment probabilities, running in $O(n^5)$ time in the length of the utterance, in addition to the language model and re-ranker's processing requirements.

As none of the current systems focus on evaluating the structure or interpretive classification of the repair, a necessary mechanism for dialogue processing as argued for by experimental evidence (see Section 2.4), I address this lacuna here. Also, overlapping (chaining or nested, complex) repairs have not had a thorough treatment in corpus analyses despite the ontology set out by Shriberg (1994). Also, repairs involving partial words have not been previously addressed in a comprehensive way syntactically, formally or computationally, although there has been acoustic modelling work in this area (Liu, 2004). Partial words are standardly removed from training and testing in the Switchboard disfluency detection task from (Charniak and Johnson, 2001) onwards.

4.1.1 Surface form and incremental context

To address the above, this study examines the relationship between an utterance's context and the presence, form and function of self-repairs in dialogue. Under the assumption of incremental time-linear processing outlined in Sections 3.5.3 and 3.5, this work builds towards an interpretable incremental model that can predict the most likely repair form and its interpretation from four positions in a repair: (1) the optional editing signal (interregnum), (2) the repair onset, (3) during the repair, and (4) at the repair end boundary. I assume these tasks all interact with detecting the reparandum onset, so the interaction between that task and the other tasks will also be part of the model. The over-prediction of reparandum onsets in the absence of a repair indication is something I wish to avoid for reasons of computational complexity that current systems suffer from, resulting in slow-run times, unnecessary memory use, latency and psychological implausibility. The design of the automatic repair detection and classification system described in Chapter 5 and the dialogue system models in subsequent chapters will be informed by this study.

4.1.2 Terminology

For the purposes of this corpus study, I use the self-repair annotation scheme first proposed by Shriberg (1994) and the Switchboard disfluency corpus annotation protocol (Meteer et al., 1995), reprised from (2.4):

$$\text{John} \underbrace{[\ \text{likes}}_{\text{reparandum}} + \underbrace{\{\text{uh}\}}_{\text{interregnum}} \underbrace{\text{loves}\ ]}_{\text{repair}} \text{Mary} \tag{4.1}$$

In addition to this structural vocabulary, from here on I term the *repair onset* to be the first word after the (possibly null) interregnum, and the *interruption point* as the transition labelled '+' after the reparandum-final word. The first study investigates the distribution of the different forms of repair based on the affordances of this structure, while the others investigate the interaction between context and these surface forms. The edit terms that constitute most repair interregna have a characteristic vocabulary, a fact Heeman and Allen (1999)'s system exploited to detect discourse markers, a subset of them, with almost perfect accuracy. Here I also investigate context in terms of edit term presence to test their predictive ability of upcoming repair onsets.

4.2 Corpus preparation

4.2.1 Corpora

I use the section of the Switchboard corpus (Godfrey et al., 1992) that has been both dialogue act tagged (Jurafsky et al., 1997; Shriberg et al., 1998) and annotated for disfluencies (Meteer et al., 1995). This constitutes 1126 transcripts of different telephone conversations between pairs of native American English speakers from various regions of the United States between 1990 and 1992. The conversations are loosely task-directed: each pair chose from a variety of topics of conversation in a set list (e.g., 'exercise and fitness', 'consumer goods') and were connected to each other by an automatic switchboard. The conversations are between strangers, ranging from one-and-a-half up to ten minutes in duration, with an average of six-and-a-half minutes (Calhoun et al., 2010).

Each repair is marked as described above, with *reparandum*, *interregnum* and *repair* intervals (Meteer et al., 1995). The only compulsory interval for a repair is the reparandum, as those considered deletes do not have repair phases. All repairs have a repair onset however, which is the word directly after the possibly null interregnum. I will also consider stand alone edit terms and their relationship to interregna forms here.

Note that below, (4.3) has an embedded repair structure, being counted as 2 repairs in this case, and (4.4) is the only example below with an interregnum, which is also given a disfluency

type– here ‘D’ within the curly brackets indicates a discourse marker. The other two types that appear below will be labelled ‘E’ (editing phrase) and ‘F’ (filled pause); when these three are considered together I will call them *edit terms*. I do not consider asides, marked ‘A’ in Switchboard as edit terms, as these can be considered parenthetical constructions that are not considered disfluencies by many accounts of the grammar. When they appear at a repair onset they will be grouped into the repair phase.

(4.2) “. . . but my kids are only elementary [grades, + levels] right now”
(Switchboard conversation number sw4325, repeating (2.1) above)

(4.3) “ [[I guess + I k-,] + I think] it’s got some relevance”
(sw4330)

(4.4) “And I find that for [a normal, + {D you know, } everyday things.] It’s really very easy to work on”
(sw4356)

Penn TreeBank III syntactic tree representations are available for 650 transcripts (Marcus et al., 1999), which, as I aim to investigate syntactic context, provides a sub-corpus I will use for various tests below. The treebank files constitute the data used in the standard Switchboard disfluency detection task, divided into training, held-out and test sets (Charniak and Johnson, 2001, onwards)– in this chapter I am careful not to use the test set, as this will be used in evaluating an automatic repair detector in Chapter 5. The principal three corpora comprise the following:

`SW_train`: the standard Switchboard training data (all conversation numbers sw2*,sw3* in the Penn Treebank III release)

`SW_non_PTB`: the non-Treebank dialogue act tagged Switchboard files[2]

`SW_heldout`: the same as the standard held-out data (PTB III sw4[5-9]*)

The demographics of the speakers, the number of repairs annotated and the size of these three corpora can be seen in Tables 4.2, 4.1 and 4.3. I also make use of a few files from `SW_PTB_rest`, the remaining treebank files not used in the standard training testing or heldout sections (PTB III sw4154 - sw4483), but not for quantitative purposes.

[2]These files are not in the training, held-out or test data of the standard task as described by (Charniak and Johnson, 2001, inter alia.)

Transcripts	496
Words	≈ 651,000
Utterances	≈ 100,000
Different topic prompts	64
Speaker gender	M=415 (≈ 42%), F=577 (≈ 58%)
Speaker age (mean)	(≈ 38yrs) <20: 0% 20-29: 25% / 30-39: 36% 40-49: 22% / 50-59: 15% / 60-69: 2%
Total repairs annotated	22,691
First-position/cross-turn repairs	21,427 (94.4%)/1,264(5.6%)

Table 4.1: Corpus statistics for `SW_train`, the Penn Treebank III parsed Switchboard Corpus (Training files)

Transcripts	476
Words	≈ 631,000
Utterances	≈ 96,600
Different topic prompts	64
Speaker gender	M=381 (≈ 40%), F=571 (≈ 60%)
Speaker age (mean)	(≈ 39yrs) <20: 0.1% 20-29: 20% / 30-39: 37% 40-49: 18% / 50-59: 22% / 60-69: 3%
Total repairs annotated	20,095
First-position/cross-turn repairs	18,921 (94.2%)/ 1,174 (5.8%)

Table 4.2: Corpus statistics for `SW_non_PTB` the non Penn Treebank III dialogue act tagged Switchboard Corpus

Transcripts	52
Words	≈ 48,900
Utterances	≈ 6,400
Different topic prompts	27
Speaker gender	M=76 (≈ 73%), F=28 (≈ 27%)
Speaker age (mean)	(≈ 41yrs) <20: 0% 20-29: 27% / 30-39: 16% 40-49: 13% / 50-59: 43% / 60-69: 0%
Total repairs annotated	2,261
First-position/cross-turn repairs	2,251(95.6%)/ 100 (4.4%)

Table 4.3: Corpus statistics for `SW_heldout`, the Penn Treebank III parsed Switchboard Corpus (Heldout files)

In terms of the demographic of speakers there are some differences between `SW_train` and `SW_non_PTB`, and `SW_heldout`, however for the purpose of general analysis of human-human phone conversations between unfamiliar participants of varying ages and gender, the differences should not be critical here (presumably with participants from a normal population distribution of mental health)– though see (Bortfeld et al., 2001) for comparisons of disfluency rate against age and gender. The tables show the ratio of cross-turn self-repairs is roughly the same in each subcorpus– note these are not always third position self-repairs as described by Schegloff et al. (1977); Schegloff (1992) (see Section 2.2), but are those where the structure of a repair extends across multiple utterance units where the interlocutor of the participant making the repair has contributed or overlapped an utterance within the repair.

Here I report different tests on each of these corpora and combinations of them. For the repair type distribution study I use the standard Switchboard disfluency training corpora `SW_train` plus the non-Treebank Switchboard files `SW_non_PTB` giving a total of 972 transcripts, ∼196,600 utterances, ∼1.28M words– (see Tables 4.1 and 4.2). For the syntactic context study I use `SW_train` (see Table 4.3). For the final semantic annotation and qualitative analysis of the dialogue function of repairs I use sections of `SW_train`, `SW_heldout` and `SW_PTB_rest`.

4.2.2 Aligning trees with transcriptions

The first data preparation stage overcomes the issue of aligning the different versions of Switchboard with idiosyncratic tags, corrections and transcription conventions. Although an integration effort was made through the NXT Switchboard Annotations from the Linguistic Data Consortium (LDC), an XML resource linking different versions (Calhoun et al., 2010), the problem for repair analysis persists. Despite this recent effort, NXT was not suitable here due to the lack of repair phases in the disfluency annotations. Instead, considerable pre-processing had to be carried out on the dialogue-act tagged and disfluency-annotated files as collated by Potts (2011).

The principal pre-processing task consisted of automatically matching the words in the transcripts to their corresponding leaves in the Penn Treebank III parse trees (a task which also equates to matching them to their POS tags which always form parent nodes to the words). There are slightly different numbers of trees than there are utterances, and tree-utterance mapping can be one-to-many and many-to-one. I developed an algorithm to match tree leaves (and POS tags) to words to deterministically ensure a mapping wherever possible. Some re-annotation of the corpora had to be carried out manually to ensure the words aligned. It systematically ig-

nored punctuation and disfluency markers, and non-linguistic utterance units constituting transcriptions such as `<Throat_clearing>` and `<Laughter>` . I checked manually that the tree-mapping algorithm had sufficient precision and recall over the words in all utterances by checking random samples, and was satisfied with the mappings.[3]

4.2.3 Manual repair annotation

Considerable re-annotation of self-repairs was carried out manually according to the Switchboard disfluency annotation manual (Meteer et al., 1995) to resolve the mismatch between the Penn Treebank III trees and the original disfluency tags. The mismatches that required re-annotation were those where a treebank EDITED node is a mother of a leaf node but the word corresponding to that leaf node in the transcription is not part of a reparandum. As these amount to inter-annotator disagreement between the treebank and transcript annotators, in each case I made a decision to preserve the treebank annotation of an EDITED node (siding with the treebank annotator) and annotate the rest of the repair structure, or discard it (siding with the transcript annotator). A decision was made not to side with the treebank annotation where I judged the EDITED node annotation has been done for yielding a felicitous 'clean' tree (one which conforms to standard phrase structure conventions when excised of reparanda and interregna) rather than annotating a self-repair. This was a rare case however and the majority of the 2500 words with EDITED mothers not marked as reparanda in the transcripts were retained.

4.2.4 Global corpus repair rates

For each subcorpora I extract all the self-repairs based on the annotations and the number of self-repairs extracted from each one can be seen in Tables 4.2 – 4.3.

Over all the sub-corpora, the mean base-rate likelihood of a given word beginning a repair onset is 0.0393, that is on average once every 25.5 words of speech.[4] The overall rate for the corpora that an utterance (where utterances include those where overlapping from the interlocutor

[3]The word→tree and word→POS mappings will be made available for public release and can easily be integrated into the resources of the Python NLTK sourced versions of Switchboard corpus tools Potts (2011).

[4]Here I exclude the first word of every utterance that is not a continuation, as you cannot begin a disfluency repair initiation across these boundaries– in Table 4.4 this is the repair onset:word rate in the third column. Without this cleaning the overall rate (repair:word rate (raw) in the table) is 0.0338 or 29.6 words. $\sim$100 repairs' repair onset occur at the same word as an embedded delete repair, so this simple division of the number of repairs annotated over the number of words is slightly larger than the probability of a repair onset occurring, though insignificantly so..

Corpus	repair:word rate (raw)	repair onset:word rate	repaired utterance rate
SW_non_PTB	0.0318	0.0370	0.167
SW_train	0.0349	0.0406	0.180
SW_heldout	0.0463	0.0529	0.238

Table 4.4: Global repair rates in subcorpora

has taken place) is 0.176. See Table 4.4 for the separate rates for each subcorpus. I do not distinguish between repairs crossing utterance boundaries and those marked within an utterance unit, treating them both as first-position within one continual stream, however the difference between these two types would be interesting to consider in further study.

These figures are comparable to Colman and Healey (2011)'s, lying between the rates reported in free conversation and those in task-specific dialogues, which given Switchboard is loosely task-directed seems consistent. While I acknowledge this rate could be idiosyncratic to this domain– see Shriberg (1996), Colman and Healey (2011) and Faust and Artstein (2013) for discussion of how domain can affect overall rates– I consider this sufficient data to build a domain general self-repair mechanism for dyadic interaction.

4.3 Hypotheses

From the previous work on repairs in dialogue described in Chapter 2 and after a qualitative pre-analytic observation of the transcripts, I arrived at several hypotheses about the self-repair phenomena in Switchboard. Some can be seen as investigatory constraints and aims and some as concrete testable predictions.

Hypothesis (1)- Self-repairs have a systematic surface form.

I assume self-repairs are processable in dialogue (and able to be annotated consistently in transcripts) due to some structural similarity to one another. The extent of this similarity and difference to fluent parts of an utterances will be investigated.

Hypothesis (2)- Position of interruption point contributes to predicting the type of self-repair.

This hypothesis assumes that distinct repair types have different processing effects. There is a lack of clarity about what the relationship is between the position of a word in the utterance and the probability of the different types of repair. If possible the study should use a more fine-grained than the *initial* and *medial* utterance distinction which shows some skewing in the distributions of the different types in (Shriberg, 1994)

> *Hypothesis (3)- The presence and type of an edit term contributes to predicting the presence and type of self-repair.*

This hypothesis is that the type and surface form of edit terms can be used to predict the presence and type of repairs more accurately than previously has been shown. Heeman and Allen (1999) showed an interaction between discourse markers, a subset of edit terms, and self-repairs, but their results were not interpretable enough to produce an interregnum vocabulary which could predict types of repairs. Johnson and Charniak (2004)'s unigram model for generating interregna does not consider their context, which is unsatisfactory for an incremental processing account: there should be a relationship between the form of the interregnum and the form of the following repair if speakers have little trouble processing them. Interaction with trends found in Hypothesis (1) could also be interesting.

> *Hypothesis (4)- The syntactic context of an utterance (the partial tree) can contribute to predicting the form and structure of a self-repair.*

A principal empirical aim is to investigate how and to what extent the information in partial syntactic structures as they are built up incrementally can predict repair forms. The investigatory question can be formulated as more technically as: which features of a connected syntactic tree that spans the utterance string up to (and including) the repair onset can make predictions about up-coming repairs, or the form of repairs if one is detected?

> *Hypothesis (5)- Processing context (fluency of ongoing utterance) can help predict the occurrence and type of a repair.*

The effect of overlapping repairs/repair clusters has not been fully explored, while Shriberg (1994) suggested a 'synergy' effect for overlapping repairs, as there is a heightened probability of repair during a repair or in the same utterance as a repair, this has not been comprehensively

modelled. A more systematic model may be able to predict more accurately the changing probability in a repair occurring, given a processing context (how fluent the utterance has been so far).

> *Hypothesis (6)- Self-repairs can be interpreted by interlocutors and annotators as having a particular dialogue function.*

This final hypothesis is one which addresses the meaning of self-repairs as constructed by dialogue participants. An annotation study and qualitative analysis of the data are needed to address this.

4.4 Methodology

The principal corpus mark-up tasks to test the hypotheses were: (1) categorise repair types within a consistent taxonomy; (2) define syntactic context appropriately for partial and incomplete utterances using their Penn Treebank parse trees (3) define elements of interest for processing context (4) define a semantic ontology for repairs. I explain the first three parts of the methodology below, and introduce the last one in Section 4.6.

4.4.1 Repair surface form types

For the repair surface form type study I extracted 42,786 self-repairs from `SW_train` and `SW_non_tree`, based on the merge of the Treebank and transcript annotations (see Table 4.4), making this to my knowledge the largest number of self-repairs studied in a corpus.

Repair form taxonomy by alignment

To investigate the distribution of the different types of repair, I follow Johnson and Charniak (2004) in their use of weighted minimum string edit distance alignment, for an algorithmically defined surface form taxonomy. The aligner classifies the 42,786 examples. It operates by mapping each reparandum word to a repair word, where each word must receive at least one alignment with the lowest possible cost.

In addition to Johnson and Charniak's alignment categories I introduce REP[complete], which aligns prefix→complete word relations such as "j- + just" and its inverse REP[partial] which aligns complete word→prefix such as "just + j-". I used the costs as in Table 4.5.

Function	**Alignment type**	**Cost**
delCost	DEL	4
insCost	INS	4
subCost	REP	0
	REP[complete]	1
	REP[partial]	1
	SUB[same POS]	2
	SUB[same POS class]	5
	SUB[arbitrary]	7

Table 4.5: Costs for reparandum-repair alignments.

The costs were chosen to ensure that 'weaker' substitutional relations are replaced by stronger ones: e.g. REP[complete] and REP[partial] are a close approximation to phonological onset alignments they should be selected as stronger alignments over SUB[same POS]. Following Johnson and Charniak (2004), the weights are also designed so an arbitrary substitution will always be chosen over an insert and a delete– this reduces spurious ambiguity, reducing the number of possible alignments.

I implement weighted minimum edit distance alignment with back-trace storage of possible alignment sequences, using the popular dynamic programming algorithm that runs in polynomial time (Needleman and Wunsch, 1970; Wagner and Fischer, 1974). I follow Jurafsky and Martin (2009)'s explanation of this algorithm. For a reparandum of length m and a repair of length n, a tabular dynamic programming algorithm computes the lowest cost alignment between the reparandum and repair in $O(m \times n)$ time and in $O(m \times n)$ space as it functions by populating a $(m+1)$ by $(n+1)$ simple edit distance matrix, using a simple recurrence relation algorithm. In using the matrix D, where $D[i, j]$ corresponds to the alignment score between the prefix $w_0 - w_i$ of the reparandum and prefix $w_0 - w_j$ of the repair phase, the algorithm can iterate over all cell positions, using recurrence relations to populate the cell with a score and a pointer position. See Algorithm 1.

The pointer matrix stores a set of $< pointer, alignment >$ tuples in each cell to indicate the

Algorithm 1 Weighted Minimum Edit Distance Alignment for Self-Repairs

Input: *reparandum, repair*

$m \leftarrow$ Length(*reparandum*)
$n \leftarrow$ Length(*repair*)
$reparandum \leftarrow \oslash + reparandum$ ▷ Add the empty string as the first word.
$repair \leftarrow \oslash + repair$
Initialise distance matrix $D[m+1, n+1]$ and pointer storage matrix $ptr[m+1, n+1]$

$D[0,0] \leftarrow 0$ ▷ Initialise distance between empty string and empty string.
for *i* from 1 to *m* **do** ▷ Initialise distance of other zero'th row and column cells.
  $alignment \leftarrow$ null
  $D[i,0] \leftarrow$ delCost($reparandum_{i-1}$,*alignment*)
  $ptr[i,0]$.append($<\uparrow, alignment>$)
for *j* from 1 to *n* **do**
  $alignment \leftarrow$ null
  $D[0,j] \leftarrow$ insCost($repair_{j-1}$,*alignment*)
  $ptr[0,j]$.append($<\leftarrow, alignment>$)

for *j* from 1 to *n* **do** ▷ Main recurrence relation computation.
  for *i* from 1 to *m* **do**
    $alignment \leftarrow$ null ▷ Best alignment relation variable gets value from choice function.

$$D[i,j] \leftarrow Min \begin{cases} D[i-1,j] + \text{delCost}(reparandum_{i-1}, alignment) & (del) \\ D[i,j-1] + \text{insCost}(repair_{j-1}, alignment) & (ins) \\ D[i.j] + \text{subCost}(reparandum_i, repair_j, alignment) & (sub) \end{cases}$$

$$pointer = \begin{cases} \uparrow & \text{iff } (del) \\ \leftarrow & \text{iff } (ins) \\ \nwarrow & \text{iff } (sub) \end{cases}$$

    ▷ Select corresponding pointer.
    $ptr[i,j]$.append($< pointer, alignment >$) ▷ Update the pointer matrix accordingly.

$Alignments \leftarrow \{x : \text{Reverse}(x) \in \text{BackTracePaths}(ptr[m,n] \rightarrow ptr[0,0])\}$ ▷ Now get all possible complete alignments by backtracing over the pointer matrix.

Return $< D[m,n], Alignments >$

direction from which its score originated and what exact alignment operation was used. In the first study, the atomic alignment types correspond to those described above: e.g. comparing "j-" at position i and "just" at position j would yield $<\nwarrow$,'REP[complete]'$>$ at cell $ptr[i,j]$ in the pointer matrix. Algorithm 1 backtraces depth-first through the pointer matrix from $ptr[m,n]$ to $ptr[0,0]$ to return the set of valid alignment sequences, such those in Table 4.6 as well as the (weighted) minimum edit distance cost. When faced with multiple alignments, I decide to favour those with the best *incremental score* as the alignment is computed left-to-right (best-first), as I

feel this is more appropriate for testing the efficacy of this approach for incremental self-repair processing.

4.4.2 Context

Syntactic context through tree paths and constituency

The task of characterizing syntactic context is challenging, given the lack of incremental representations for the parse trees in Switchboard, and more generally the lack of a standard approach for incremental tree representation for incomplete structures. It was decided that a measure of syntactic constituency of the words in the utterance up to the interruption point could be characterized by *tree path length* between the leaves, that is the distance from the leaf (terminal node) mapped to the word in question to the leaf mapped to its preceding word– see Figure 4.1 for an example.

This annotation was automated with an algorithm using Python methods devised within the constraints of the Python NLTK `tree` class, using the tree map created, as described above, to bottom-up search for the common mother node between the leaf node word and its predecessor. It overcomes the problem of 'EDITED' non-terminal nodes, which under a strict incrementality assumption would not be available to a hearer when processing on-line, by ignoring the path directly above the edited node in the tree path total length, and ignoring the path to the node directly below it if it leads to a node which has the '-UNF' (unfinished) suffix and a stem identical to a node above the edited node. This is a way of simulating a connected tree at the time of processing the word, factoring out their EDITED status before the repair onset has been processed. Notice the repair phase "I don't" has the same tree path length values as the repeated reparandum phase in Figure 4.1.

Processing context

I am interested in the incremental processing context of an utterance so far, that is to say, the level of fluency that has been exhibited by the speaker up to a given point in the utterance. To do this I measure several factors which are easily calculated from the data:

- The binary feature of whether a repair has been produced in the utterance so far
- The presence of an edit term at the given point in the utterance
- The distance from the utterance start in words

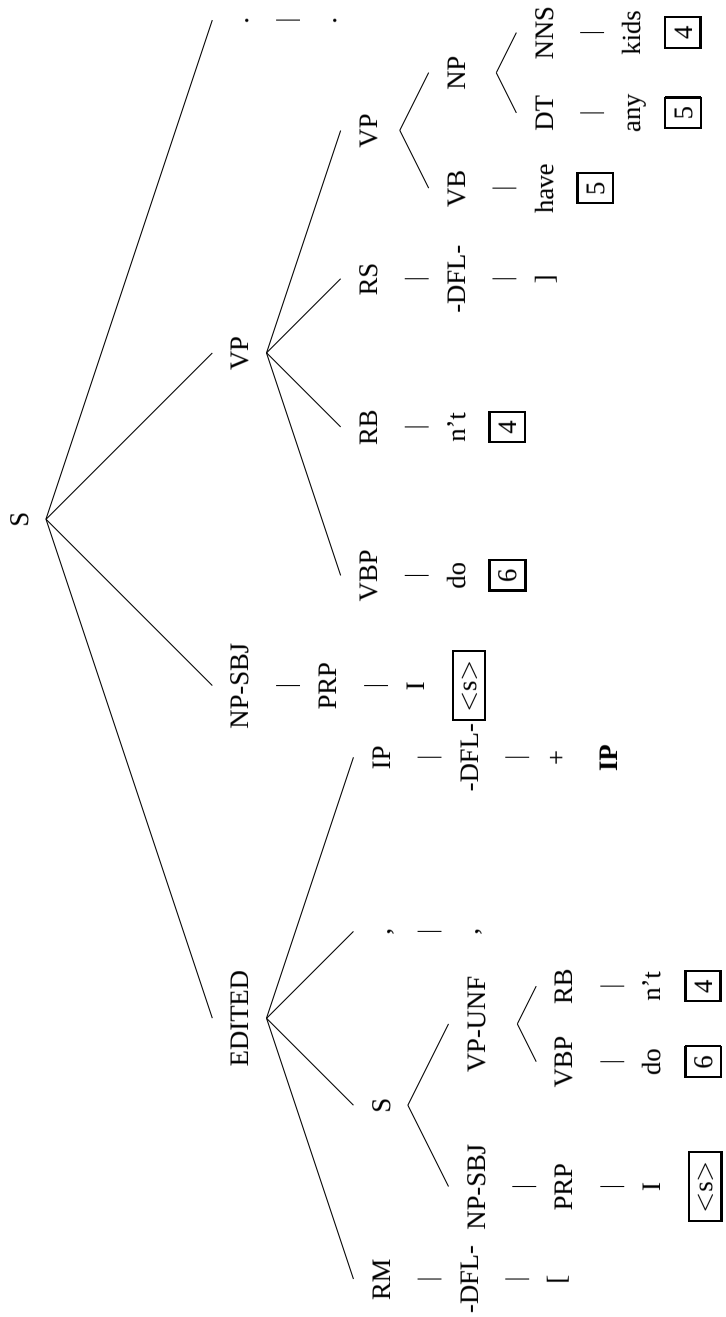


Figure 4.1: Annotation of tree path length between a word and its preceeding word

4.5 Results and analysis

4.5.1 Repair surface form types

The most frequent aligned structures extracted are shown in Table 4.6: I split the structures between the broad classes of verbatim repeats (all repeat alignments), substitutions (all other alignment types with a repair phase) and pure deletes (no repair phase annotated), in order to get the most prototypical deletes as judged by the Switchboard annotators.

1,994 different alignment sequence types were found, with only 31.0% of types occurring at least twice, a figure similar to Heeman and Allen (1999)'s reported 29.4% of templates within the TRAINS corpus. As can be seen, the majority of types are within substitutions (1,974), which have a long tail of types – the 10 example substitutions shown only cover roughly half of the tokens all substitution occurrences. Deletes were the rarest, conflicting with Shriberg (1996), but mainly due to my definition covering pure deletes only.

While building a rule-based repair grammar is not what I advocate in this thesis, it is worth noting the observed alignment sequences can be compressed into 194 different operation sequence pairs such as [$SUB(r_{m-i}$-$R_{n-j})$ REP (r_m-R_n)], in this case representing a substitution alignment from i words back from current reparandum index m to a repair word j words back from current repair index n, followed by a repetition alignment between the current indices. In terms of coverage, due to the sparsity of most alignment sequences, the strength of Johnson and Charniak (2004)'s generative S-TAG grammar approach over a template based one (Heeman and Allen, 1999) becomes clear – for example the approach allows the most frequent repair type, repeats, to have high likelihood within a repair 'grammar', regardless of their length.

Reparandum and repair lengths

First-turn repairs tend to be very short, with a mean reparandum length of 1.581 words (pop. st.dev = 1.108).[5] As with the distribution of many linguistic phenomena, their length distribution can be characterized as an inverse power law with a good fit: a function $y = 0.9248x^{-2.595}$, where x is the reparandum length in words and y is the average relative frequency of that length, has a goodness-of-fit R^2 = 0.9697 up to length 6. Reparanda of 1 or 2 words account for 90.8% of repairs and lengths 1-3 account for 96.5%. Repeats (mean=1.217 words) and deletes (mean=1.372 words) are significantly shorter than substitutions (mean=2.068 words), which also exhibit a

[5] I count partial words as one word tokens in these calculations.

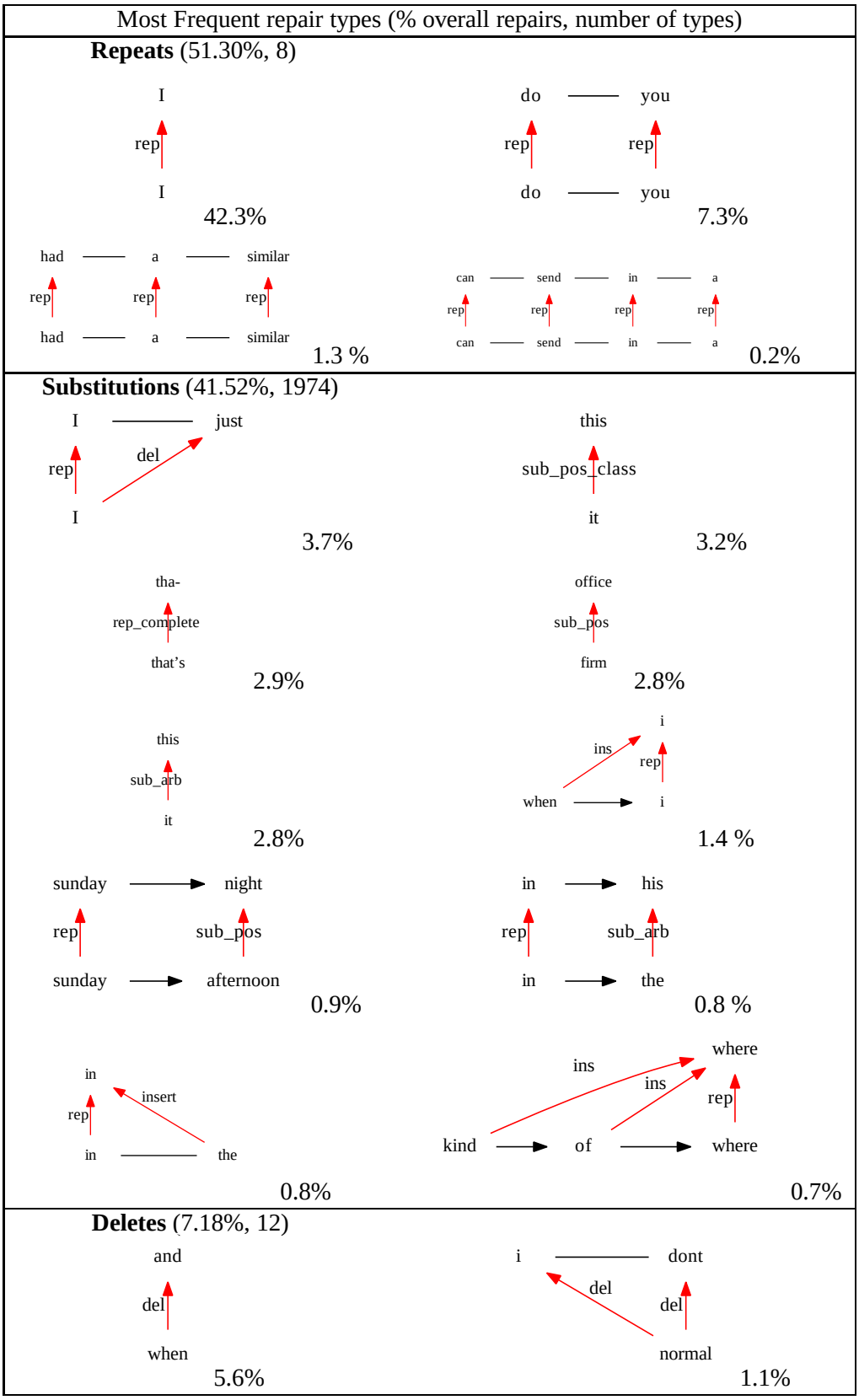


Table 4.6: Distribution of the most frequent repair disfluencies in Switchboard

Class	% of repair tokens	number of types	+interregnum (% of class)
Repeats	51.29%	8	12.47%
Substitutions	41.52%	1974	18.52%
Deletes	7.17%	12	22.91%
ALL	100.00%	1994	15.77%

Table 4.7: The tokens, types and interregnum presence for the three repair classes

Class	mean reparandum length (std.)	power curve for length
Repeats	1.217 (0.533)	$y = 1.9873x^{-4.780}$, $R^2 = 0.9273$
Substitutions	2.068 (1.428)	$y = 0.7292x^{-1.949}$, $R^2 = 0.8895$
Deletes	1.372 (0.962)	$y = 0.9143x^{-2.993}$,$R^2 = 0.9891$
ALL	1.581 (1.108)	$y = 0.9248x^{-2.595}$,$R^2 = 0.9697$

Table 4.8: Mean reparandum lengths and the power curve fit over all lengths

shallower power-law decay – see Table 4.8 for the figures and Figure 4.2 for the plots.[6]

With the vast majority of reparanda being 1-3 words long, a very local model of context could be used to capture them. As mentioned, previous approaches using sequence-based language models in combination with repair grammars and templates have had some success, but there is scope for incorporating repair detection more directly into an n-gram model, though not necessarily through Hidden Event Language Models (HELMs) (Liu et al., 2003; Georgila, 2009), which require bigger values of n and more training data. Furthermore, as Shriberg and Stolcke (1998) showed, the likelihood of retracing back one more word in retraces decays logarithmically with the number of words into a fluent word sequence, so the need to store all possible reparandum sites before having heard an interruption point seems unnecessarily complex: a locally triggered recovery mechanism does not have far to backtrack. Repeats and deletes are frequently short so their repair onset and complete reparandum will often fall within a bi- or tri-gram: for example, presuming perfect interregnum and edit term recognition, a trivial repeat-word feature $w_i = w_{i-1}$ captures 42.3% of all repairs. Use of such local alignments may yield high precision for those

[6]These figures are slightly different to Hough and Purver (2013) due to the fact these are the raw values rather than lengths calculated on 'cleaned-up' repairs whereby internal embedded repairs are removed. I thought this was more realistic in terms of modelling the surface forms.

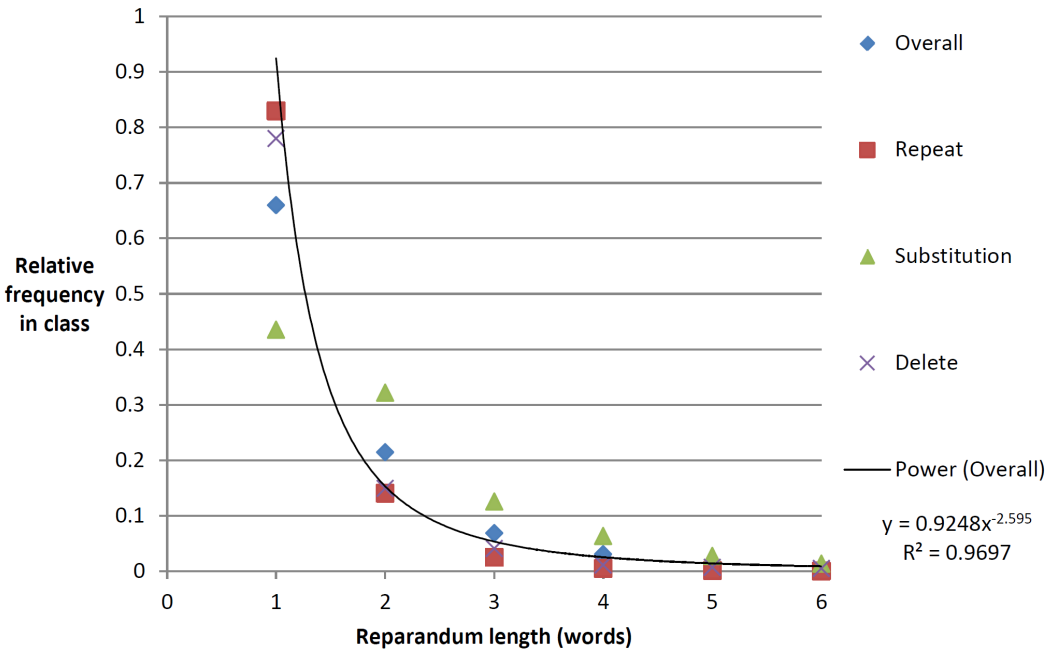


Figure 4.2: Reparandum length decay for each type and power law for all repairs

types, but we need a more general way of detecting interruption points in a local n-gram context which can also capture longer repairs, as will be discussed below.

4.5.2 Context

Processing context: position of repair onset and type

To investigate how likely the three types of repair are at each point in the utterance, I simply normalised the occurrence of each repair onset type by the number of times that utterance position is present in the corpus. The results of this for utterance positions from 2 to 30 can be seen in the graph in Figure 4.3 (note a repair never occurs on the first word as they by definition do not cross utterance units). The results support Shriberg (1994)'s more coarse-grained analysis of utterance position in that repeats are more likely utterance-initially, however our distinction between deletes and substitutions in a well defined algorithmic ontology as described above shows some interesting differences: the probability of substitutions peaks between 4-6 words then tails off sharply, while the probability of a delete rises steadily through the utterance.

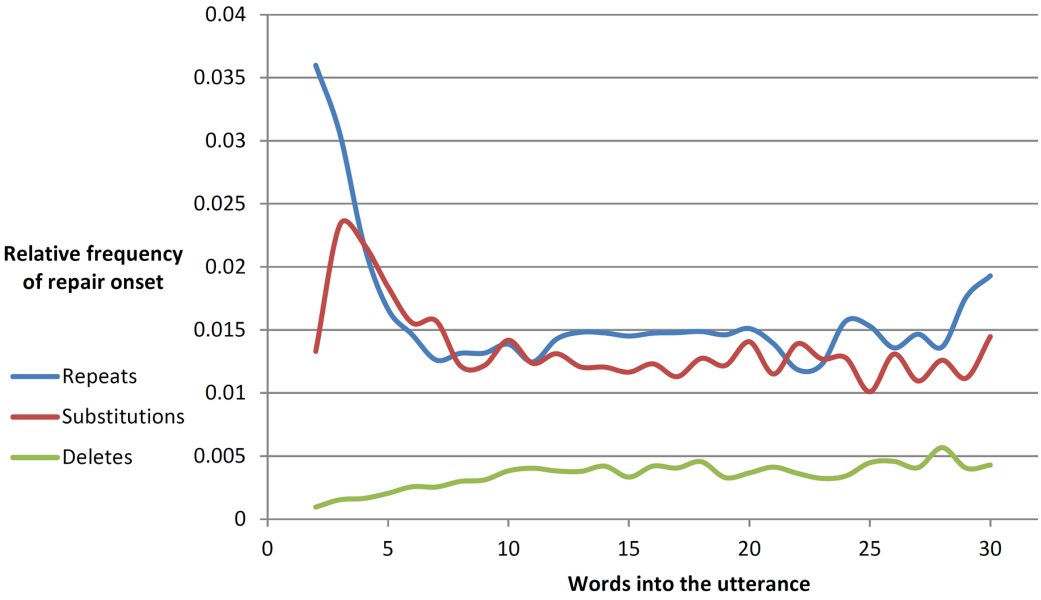


Figure 4.3: The effect of utterance position on likelihood of repair onsets of each type

Processing context: embedded repairs and repair contagion

11.9% of all repairs are embedded inside a longer structure – this divides between 9.9% chaining repairs, embedded within the reparandum phase as in (4.5), and 2.0% nested within the repair phase of a longer repair.[7] While these appear to need more complex resolution mechanisms, which is presumably why they are ignored in the training phase and evaluation of automatic disfluency systems, they need not be processed as hierarchically embedded structures by listeners on-line. They are frequently short, with mean reparandum 1.28 words long (std=0.67), and so can be resolved very locally, again in a short n-gram context, and may provide an immediate feature for following repair onsets. Intuitively an interruption point indicates speaker trouble, so the likelihood of a consequent interruption point in the following word transitions increases.

(4.5) " [[This, + it,] + they] are really... "

Embedded chaining substitution (sw3389)

In terms of the predictivity of these repairs, in the chaining case there is a boost in probability from the base-line repair probability with p(repair2onset|repair1end)=0.110 whilst in the nesting case the probability of a repair happening is less likely p(repair2onset|repair1onset)=0.020.

[7] While Shriberg (1994)'s thesis and Meteer et al. (1995)'s annotation attempted to formalise these, they remain a problem for consistency of annotation- it is not always clear whether they should be annotated as nested or chaining.

There is also an overall effect of contagion if there has been a repair in the current utterance already. If there is a repair the probability of an up-coming repair rises from the normal repair probability discussed in Section 4.2.4. In `SW_non_PTB` and `SW_train` this goes up from 0.174 to 0.269.

With these observations of intra-utterance processing context, there is clearly a sensitivity to a repair already having been made locally.

Processing context: partial words as interruption point indicators

The most reliable lexical indicator of a repair onset is a preceding partial word. According to the transcripts, the likelihood of a repair onset following a partial word that is not utterance-final is 0.935, boosting the likelihood significantly more than the presence of an interregnum, as will be discussed below. Furthermore, the remaining 0.065 of probability mass for continuations, upon inspection, look like mis-transcriptions or approximation to colloquialisms which are in fact complete words. Reparandum-final partial words are present in 10.4% of repairs. Furthermore, the completion of a single partial word is one of the most frequent repair structures (2.9% of all repairs, see Table 4.6). The probability of the partial word being a deleted reparandum also rises from the overall average rate 0.072 to 0.171.

This is clearly a very useful feature for detection and classification. Charniak and Johnson (2001) posit an optional phase between the reparandum and the interregnum called the 'free-final', consisting of a sequence of partial words of any length, which, when used as a training feature for an edited words classifier, can improve the detection of repairs. Subsequent work does not use partial words in an attempt to simulate a more realistic testing situation for dialogue systems. While I cannot make direct predictions here without the acoustic data, I investigate how a simple word completion predictor could be a fair approximation to an annotator's incremental processing in Section 5.6.2.

Processing context: interregna and edit terms

I investigated the rate of interregnum presence, the interregna forms most predictive of repair, and the effect of interregnum presence on predicting repair surface form type.

In Switchboard, only 15.77% of repairs have an interregnum, so it is not a strong repair indicator, a surprising result given their important role in formal and empirical models (Ginzburg, 2012; Ginzburg et al., 2014). However, if one is identified correctly, its presence signals information about the type of upcoming repair: the likelihood of a substitution rises to 0.489

form	p(repair|form)	p(form|repair)
(fluent word)	0.039	0.842
“uh”	0.155	0.071
“you know”	0.100	0.037
“um”	0.061	0.011
“I mean”	0.074	0.005
“well”	0.080	0.005
“or”	0.017	0.003
“like”	0.014	0.003
“yeah”	0.038	0.002
“oh”	0.005	0.002
“actually”	0.025	0.001

Table 4.9: Distribution of interregnum forms in first-position self repairs in Switchboard

(compared to 0.415 in general, see Table 4.7), the likelihood of a delete rises to 0.105 (from 0.072), while the likelihood of a repeat drops to 0.406 (from 0.513). There are more substitutions with interregna than repeats in raw frequency and significantly more relative to their class size (3291/17767 (18.52%), versus 2739/21918 (12.47%) $\chi^2_{(1)}$=278.5,p$<$0.0001), and significantly more deletes have interregna than substitutions (704/3072 (22.91%), versus 3291/17767 (18.52%) $\chi^2_{(1)}$=32.6,p$<$0.0001).

Interregna share a virtually identical vocabulary to editing signals in the more common *abridged* (Heeman and Allen, 1999) or *forward-looking* (Ginzburg, 2012; Ginzburg et al., 2014) repairs which comprise an editing signal followed by a fluent continuation to their preceding context, rather than a disfluent one. Focussing here on interregnum vocabulary distributions, I obtain the probabilities in Table 4.9, showing the predictive power of the vocabulary item and its relative frequency within all repairs. The filled pause “uh” and discourse marker “you know” are the most indicative, increasing the probability of a repair from the base rate to 0.155 and 0.100 respectively. These two items are also the most frequently occurring within repairs (7.0% and 3.7% of repairs have them, respectively). The surprising fact is the lack of predictive power even the most frequent interregna forms have to predict repair. This means interregnum presence does not provide a reliable feature for detection on its own. However as it has significant interaction with repair type, it is a useful feature for repair classification or interpretation.

Repair class	Mean length with interregnum (std.)	Mean length without interregnum (std.)
repeat	1.274 (0.670)	1.209 (0.510)
substitution	2.201 (1.639)	2.041 (1.376)
delete	1.370 (0.787)	1.379 (1.013)
overall	1.738 (1.330)	1.554 (1.061)

Table 4.10: Effect of interregnum presence on mean reparandum length

Processing context: interaction of interregnum presence and reparandum length

There are some interesting interactions between the repair surface forms and their context. Particularly, there is an interaction between reparandum length and interregna presence. Overall, the length of the reparandum of repairs with an interregnum is significantly longer (mean=1.738 words) than those without one (mean=1.554 words). Repeats and substitutions with interregna are longer than their non-interregna counterparts, while there is little difference for deletes– see Table 4.10 for the figures.

Syntactic context: reparandum onset prediction

In terms of word-by-word syntactic context of utterances, there are interesting interactions which are relevant to the hypotheses. The notion of syntactic context used here is a measure of constituency, which was calculated using the tree path length measure explained in Section 4.4.

By way of visualisation, Figure 4.4 shows the results. There is a difference between the distribution of syntactic path lengths for the overall corpus (for all words) and that for reparandum onset words. Word leaf nodes of tree path length 4 to their predecessor are repaired from less than their predicted frequency in the overall corpus, where this probability mass shifts mainly to repairing from the beginning of the tree structure (<s>). This interacts with the probability of repairs being higher utterance-initially as shown in Figure 4.3, but shows one of the reasons why this might be the case, which is systematic avoidance of repairing from the middle of a constituent. This is supported by the fact speakers are also marginally more likely to repair from other stronger constituent boundary points, at path lengths 6,7,8 and 9. This suggests that repairing from the middle of a constituent is dispreferred, a result that seems consistent with Levelt (1989)'s theory of repair form and Healey et al. (2011)'s experiment.

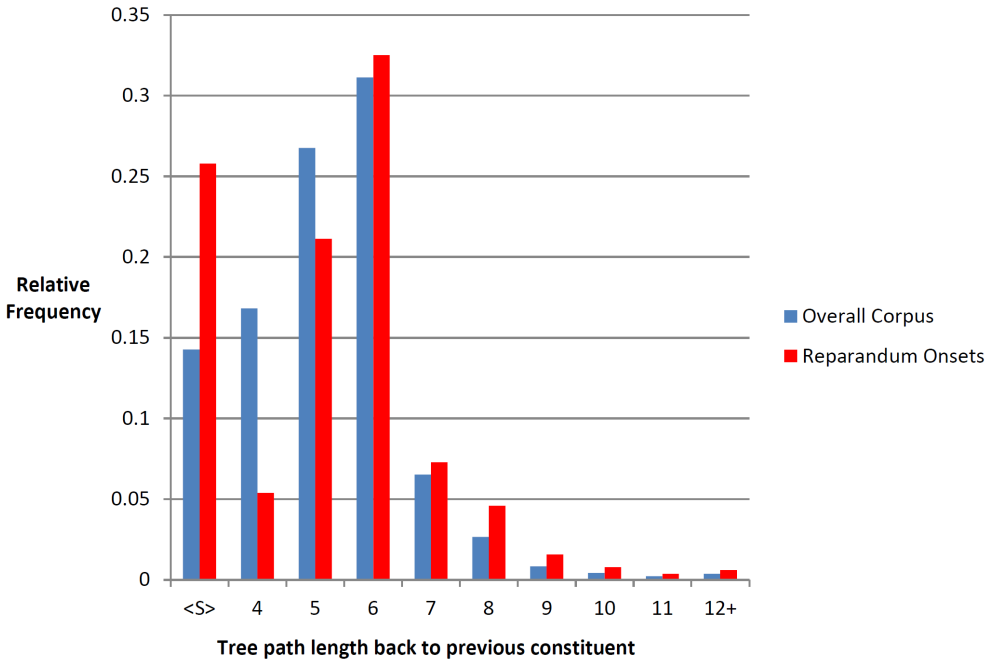


Figure 4.4: Comparison between distribution of tree path lengths over entire corpus and those in reparandum onset positions in Switchboard PTB heldout data

4.6 Qualitative observations of substitutions and deletes

In addition to the surface form and syntactic based analysis above, I also investigate the meaning of self-repairs for dialogue, assuming that there are regularities in their dialogue function.

Given the basic ontology of repair forms I describe above, I assume the different surface forms are related to different classes of dialogue function. I also assume that dialogue participants are sensitive to the function of a repair for several reasons. One may infer someone is having forward-looking difficulty during an edit term or repeat, or may make direct use of semantic dependencies between reparandum and repair phases in substitutions, while deletes may have a more cancelling effect on the reparandum and one may draw more pragmatic and turn taking inferences about utterance-initial deletes (restarts). The ability to recognize these types may help alignment and aid inference about the speaker's mental state. Also, recognizing whether your dialogue partner sits either side of the statistically significant divide between "repeaters" and "deleters" (Shriberg, 1996) may help alignment.

I analysed substitution and delete surface form repairs from random transcripts from `SW_train`,

`SW_heldout` and `SW_PTB_rest`, ignoring straight repeats as they seem to all be indications of forward looking problems as per Ginzburg et al. (2014)'s formulation.

4.6.1 Substitution functions

Firstly, within substitutions, I noticed a regular function of *paraphrasing*, or lexical choice change, where the semantics of the utterance remains largely unchanged from repair to reparandum while the lexis used does change– see (4.6) and the second repair in (4.7).

(4.6) "[It's just part + it's just something] that turns up in the other parameters that they test for"
(sw4330)

(4.7) "And, {F uh,} [it's a + it's a] legal [firm + office]"
(sw4325)

Another regular function observed was that of *adding information* to the reparandum, whereby it appears the speaker does not intend to revoke the reparandum material, but elaborate on it for some communicative effect. See (4.8) where the speaker does not merely assert that their clients are big oil companies, but revokes any commitment to the fact that they are not be big. (4.9) shows the added information that the speaker revokes the assertion that they are discussing an objectively nice environment. The additional asserted information in repairs here would not have been explicit in a cleaned up version of the utterance without the reparandum and its contrast to the repair phase.

(4.8) "So, a lot of our clients are [oil companies, + big oil companies]"
[Extra assertion: the companies are not just normal size, they are big.]
(sw4330)

(4.9) "[It's + it's] a [nice, + {F uh } relatively nice] environment"
[Extra assertion: it is not neccessarily a nice environment objectively, only relatively.]
(sw4619)

Also, in line with the idea of an appropriateness repair (Levelt, 1989), substitutions can indicate a *change in register*, for example from literal to metaphorical, as in (4.10), or as in (4.11), from a more colloquial lexical choice (presumably "dump" here) to a more formal register ("landfill").

(4.10) "There's [too many manage, + too many chiefs] and not enough Indians"
(sw4171)

(4.11) "A guy went to [a d- + a landfill,] dug down five feet and dug up a phone book from {D like } nineteen sixty"
(sw4358)

Commitment to knowledge may also be communicated via the difference between the repair and reparandum, by strengthening (4.12) or weakening (4.13) the commitment to propositional content.

(4.12) "[[and + and] I think that + and I know] Massachusetts has. . . "
(sw4358)

(4.13) "{F Uh, } [I know + {F uh } I believe] it was last year that they actually collected the old phone books" *(sw4358)*

Another example of a commitment to knowledge change which also involves anaphoric reference to material in the reparandum is in (4.14), where "they" is resolved by "the bottles".

(4.14) "and [I think the bottles were + {D like, } I know they were] at least ten cents apiece"
(sw4329)

There are other functions of substitutions that occur regularly, such as quantification changes, polarity changes and reference changes– these will be shown below.

4.6.2 Delete functions

As for deletes, the construed meaning is often one of a cancelling or *abandonment* effect where the speaker seems to over-write the reparandum as the repair onset has no parallelism to it - see (4.15) - (4.18)

(4.15) "and you know it's {D like} [you're, + {E I mean }] employments are contractual by nature anyway"
(4330)

(4.16) "I guess [with, +] Israel is a perfect example"
(sw2252)

(4.17) "[Yo- +] a microwave, a VCR, a answering machine"

(sw4360)

(4.18) "but I used to work [and +] when I had two children"

(sw4325)

Restarts from the beginning of an utterance unit such as (4.17) are rare because the Switchboard manual instructs annotators not to mark them as repairs, but as two separate utterance units. Recovering all the restarts was beyond the scope of this thesis, and also not part of the data preparation for the automatic repair detection in the next chapter, however it is worth noting they are frequent and have the extra information that the speaker has abandoned their current contribution completely and begun afresh.

4.6.3 Towards a dialogue functional taxonomy of self-repair

Based on these observations, I propose a taxonomy of the meaning of substitution and delete repairs, excluding restarts.

Out of 216 utterances with repairs marked as substitutions or deletes in Switchboard I and another annotator marked up each utterance under a coarse-grained binary decision of whether the repair was driven by correcting the reparandum (substitution) or whether driven by overwriting the reparandum (delete), rather than re-apply the bracketing structure used in the other studies. We also applied a more fine-grained taxonomy I devised based on observations such as those just mentioned.

The fine-grained annotation scheme used was as in Table 4.11 with an example for each one, which includes some of the examples above and examples not heretofore discussed have the conversation number.

Annotators were allowed two tags from the more fine-grained scheme and only one from the more coarse-grained choice of substitution or delete. The agreement scores for any matching tag was only 57.6% for the fine-grained scheme and while the agreement for the coarse grained annotation was 83.8%, when factoring out frequency of class using Cohen's κ score this was only 0.517. The low agreement for the coarse-grained annotation is surprising, however this was in fact higher than each of the annotators' agreement with the original Switchboard annotations where non-repeat repairs without a repair phase annotated are taken as deletes, and those with a repair phase a substitution (both ≈ 0.4). After discussion, it appeared most of the disagreement

Tag	Example
SyntacticReformulation	"[Is it + {E I mean } does it]. . . " *(sw4171)*
ArticulationProblem	". . . walk the halls [and see + [an- + and] see] all these people. . . " *(sw4619)*
Paraphrase	"[It's just part + it's just something] that turns up in the other parameters..."
LexicalChoiceChange	". . . but my kids are only elementary [grades, + levels] right now"
AddingInfo	"So, a lot of our clients are [oil companies, + big oil companies]"
Abandonment	"I guess [with, +] Israel is a perfect example"
CommitToKnowledgeChange	"[[and + and] I think that + and I know] Massachusetts has. . . "
RegisterChange	"There's [too many manage, + too many chiefs] and not enough Indians"
ReferenceChange	"[We, + {F uh, } she] goes more often than I" *(sw4619)*
QuantificationChange	". . . and I do [the work on, + most of the work on] that myself" *(sw4356)*
EventChange	"[We don't + we haven't] been doing lay-offs" *(sw4171)*
AgencyChange	". . . [when he's not + when she can't] keep control of him" *(sw4325)*
PolarityChange	"And [they have a + they don't have any kind of] pension plan..." *(sw4171)*

Table 4.11: The fine-grained annotation scheme for repair function

between annotators was caused where one thought a repair made use of the reparandum and the other did not judge this to be the case, and the fine-grained disagreements subsequently followed from this. Most of the disagreement between us and the Switchboard annotations was caused by our judgement that a repair was a delete while Switchboard annotators annotated them as having a repair phase and therefore as a substitution. I would argue from this finding that deletes have been under-annotated in favour of substitutions.

4.7 Discussion

I now discuss how my initial hypotheses have been confirmed or not, conclusions arising from the results, and the limitations of the study and plans for further work.

4.7.1 Confirmation of hypotheses

Hypothesis (1)- Self-repairs have a systematic surface form.

This is the case, however there is a long tail to substitutions which means a string alignment approach to surface form is not adequate.

Hypothesis (2)- Position of interruption point contributes to predicting the type of self-repair.

This study shows there is an effect of position, with repeats being more likely to occur utterance-initially, and substitutions and deletes less so, with the probability of substitutions peaking at 4-6 words in to the utterance. The probability of a delete gradually rises through the utterance, while after the initial disparity substitutions and repeats remain roughly equally probable.

Hypothesis (3)- the presence and type of an edit term contributes to predicting the presence and type of self-repair.

The presence of edit terms alone is surprisingly under-predictive of repair, with the most predictive form ‘uh’ only raising the probability of an upcoming repair onset to 0.155. However, if an interregnum is correctly identified as one, it helps predict type. Deletes are more likely to have interregna than substitutions, and substitutions are more likely to have them than repeats.

Hypothesis (4)- the syntactic context of an utterance (the partial tree) can contribute to predicting the form and structure of a self-repair.

This is true to a degree– there is still a sparsity problem for scaling this to an automatic detector, however the results in terms of tree path length are encouraging. Speakers are less likely to repair material beyond the main constituent boundary in the partial tree they are constructing.

Hypothesis (5)- Processing context (fluency of ongoing utterance) can help predict the occurrence and type of a repair.

This is very much the case, with repair contagion being found within utterances, and also in chaining (embedded) repairs.

Hypothesis (6)- Self-repairs can be interpreted by interlocutors and annotators as having a particular dialogue function.

The annotation of the meaning of self-repairs is problematic, however progress is being made. The repair phase of the bracketing structure in the Switchboard annotations may not always be suitable for defining the function of the repair as a replacement of the reparandum by the repair, and deletes seem to be under-annotated. This may be one of the reasons the task of classification, or even assigning the bracketing structure as a whole has been avoided in evaluation in automatic repair detection in the literature. What is clear is that repairs can perform a range of dialogue functions and annotators can come to some agreement within the taxonomy proposed, however there is surprising disagreement even at a coarse-grained level.

4.7.2 The interpretation of self-repair and the problem of annotating function

The strikingly negative result from this study on the annotation disagreement over repair types may be more interesting than it would first seem. The lack of use of the audio data during annotation may have contributed to the disagreement. As not only our function annotation, but the original Switchboard repair annotations were done after transcription (Meteer et al., 1995), so clearly more work can be done to improve the resources available to repair annotators. On the other hand, this may be evidence for a more *gradient* effect of self-repair interpretation, whereby it is not clear whether a given repair only functions to cancel commitments to part of the utterance or whether it uses the reparandum to elaborate further content. The following examples show possible alternative interpretations (italicized) to the Switchboard annotations:

(4.19) "and [there's, + *?]* it's] completely generic."

Substitution or delete? (sw4619)

(4.20) "a matter where priorities are [at, +] placed.*?]*"

Delete or substitution? (sw4360)

The first type of disagreement shown in (4.19) where Switchboard annotation suggests a substitution where we suggest a delete, is the most common disagreement. The bracketing structure used may need modification in future work, as its affordances mean annotators may be tempted to use the repair phase more than they should. The higher agreement scores based on categorization are a useful start in this direction, however introducing gradient categories may help further here as recent work on grammaticality judgements has shown (Lau et al., 2014).

4.7.3 From string alignment to incremental information processing

Several bits of evidence point to the fact that instead of viewing repair interpretation as a string alignment problem, it is more fruitful to see it as an incremental information processing one, where speakers try to minimise the amount of revocation of information possible in ongoing talk.

This is made clear from the distributions of the repair types. Repeats are the most common, and add the least information (in fact no new information), acting as stalling devices much like isolated edit terms, and so Ginzburg et al. (2014)'s view of them being forward-looking problem indicators seems appropriate. Given the inverse power law of reparanda lengths, speakers try to repair as soon as possible as per Clark's principle of repair (Clark, 1996, p.284), however I would add to this imperative of locality that this is done *covertly* when possible.

While a striking correlation, the fact that mid-utterance partial words predict a repair with near certainty may be a non-argument, as mid-utterance partial words *are* in fact reparandum ends– the signal that the speaker is having trouble is made as clear as possible by not completing the word and starting the next one. These minimise the information change by revoking commitment to a potentially problematic bit of information.

Finally, there is clearly sensitivity by the speaker to previous repair, as shown in the contagion and embedding effects, and also use of edit terms to signal the nature of the upcoming repair. Preference for interregna in more severe information changing repairs, deletes and substitutions, shows they can be indicators of up-coming information revision. The interaction of reparandum length and interregnum presence also supports this, which suggests the more severe the upcoming

information change, the greater the need to signal this with an edit term, which also allows time for the re-computation required for the new material in the repair.

4.7.4 Limitations and future work

One of the limitations here is not using the audio data due to time constraints. This will clearly influence how transcribers and annotators interpret the repairs. More can be done with syntactic context, and using a parser with connected tree structures partially available, such as a PCFG parser (Roark et al., 2009) or a Dynamic Syntax parser (Purver et al., 2011) could be a next step.

A more thorough experimental evaluation of annotation using different variables could investigate how annotators disagree and agree on repair form more comprehensively. Clearly there is gradience in the interpretation, but investigating precisely how and why this happens would give us an understanding of how people interpret repairs in dialogue.

4.8 Conclusion: Consequences for models of self-repair

Given the results of this study, I conclude that only using 'rough copy' dependency between reparandum and repair phases is a restricted view of repair detection and interpretation: an incremental information processing view of these processes is more realistic, and may lead to better results in automatic repair detection, as the next chapter investigates. While regularity in surface form exists, for instance with short repeats being the most common, adherence to alignment for a model of repair is generally inadequate for the long tail of substitutions.

Edit terms prove to be less predictive of a repair onset than hoped, as they are more often forward-looking disfluencies as stand alone edit terms. This is surprising given their prevalence in the literature on repair. Edit terms can be seen as conventionalised signals of trouble in communication, but their default meaning appears not to be one to prime for a repair, rather just a signal that the speaker is uncertain how to proceed. Other features, such as the fluency level and information content of the utterance so far, must therefore be relied on to detect repair onsets by hearers and machines.

Furthermore, the annotation work suggests self-repair classification may have been avoided for understandable reasons– there is a tendency for inter-annotator disagreement and it is not clear a purely categorical way of classifying repairs (other than verbatim repeats) is a good way forward. There are a number of different functions that repairs perform, and any NLU or NLG

system capable of understanding and generating repairs in a realistic way must account for them. There is extra meaning computed on-line caused by repairs, both semantic and pragmatic, which is not available from simply removing the reparandum from the input string and re-parsing or re-generating the phrase. A context-sensitive approach to building these systems is clearly better than one which removes parts of the utterance before processing cleaned utterances.

Chapter 5

Strongly Incremental Self-Repair Detection

This chapter[1] presents STIR (STrongly Incremental Repair detection), a system that detects self-repairs and edit terms on transcripts incrementally with minimal latency, addressing problems from the previous approaches outlined in Chapter 3 and cognitive processing insights from empirical evidence in Chapters 2 and 4. STIR uses information-theoretic measures from n-gram models as its principal decision features in a pipeline of classifiers detecting the different stages of repairs. The measures can be used to model self-repair detection in terms of time-linear incremental information processing. Detection results on the Switchboard disfluency tagged corpus show utterance-final accuracy which improves on state-of-the-art n-gram model based incremental repair detection, and has considerably better incremental accuracy, faster time-to-detection and less computational overhead. STIR's performance is evaluated using incremental metrics and novel repair processing evaluation standards are proposed.

5.1 Introduction

To re-introduce the task at hand for automatic systems, I reprise (4.1) below as the structure a repair detector should be capable of recognizing:

$$\text{John} \underbrace{[\ \text{likes}}_{\text{reparandum}} + \underbrace{\{\text{uh}\}}_{\text{interregnum}} \underbrace{\text{loves}\ ]}_{\text{repair}} \text{Mary} \tag{5.1}$$

[1] Much of the work presented here is included in Hough and Purver (2014c) and Howes et al. (2014).

As discussed in previous chapters, from a dialogue systems perspective, it is not only the detection of repair presence but also appropriate assignment of the entire structure that is vital for robust natural language understanding (NLU). Downgrading the commitment of reparandum phases and assigning appropriate interregnum and repair phases permits computation of the user's intended meaning. As discussed above, for implementation into incremental dialogue systems (see e.g. Rieser and Schlangen, 2011, Section 3.3), left-to-right operability on its own is not sufficient and repair detection should operate without unnecessary processing overhead, and function efficiently within an incremental framework, meeting as many of the incremental criteria set out in Section 3.5.3 as possible.

In line with the principle of strong incremental interpretation (Milward, 1991), a repair detector should give *the best results possible as early as possible*. As discussed in 3.5, with one exception (Zwarts et al., 2010), there has been no focus on evaluating or improving the *incremental performance* of repair detection.

In this chapter I present STIR (STrongly Incremental Repair detection), a system which addresses the challenges of incremental accuracy, structure assignment, computational complexity and latency in self-repair detection, by making local decisions based on relatively simple information-theoretic measures of fluency and similarity. Section 5.2 summarizes the challenges posed and explains the general approach; Section 5.3 explains STIR in detail; Section 5.4 explains the experimental set-up and introduces evaluation metrics, some of which are novel; Section 5.5 presents and discusses STIR's results on Switchboard; Section 5.6 investigates STIR's domain-generality and practical use in psychiatry applications and Section 5.7 concludes.

5.2 Challenges and Approach

In this section I summarize the challenges for incremental repair detection: computational complexity, repair hypothesis stability, latency of detection and repair structure identification. In 5.2.1 I explain how I address these.

Computational complexity Approaches to detecting repair structures often use chart storage (Zwarts et al., 2010; Johnson and Charniak, 2004; Heeman and Allen, 1999), which introduces a computational overhead: if considering all possible boundary points for a repair structure's 3 phases beginning on any word, for prefixes of length n the number of hypotheses can grow in the order $O(n^4)$. Exploring a subset of this space is necessary for assigning repair structures as

in (5.1) above, rather than just detecting reparanda: the (Johnson and Charniak, 2004; Zwarts et al., 2010) noisy-channel detector is the most successful system that applies such structures (and the only one that does so using a generative model) but the potential run-time complexity in decoding these with their S-TAG repair parser is $O(n^5)$. In their approach, complexity is mitigated by imposing a maximum repair length (12 words), and also by using beam search with re-ranking (Lease et al., 2006; Zwarts and Johnson, 2011). In order to include full decoding of the repair structure, which as argued in the previous chapter is necessary for full interpretation, whilst also taking a strictly incremental and time-critical perspective, reducing this complexity by minimizing the size of this search space is crucial.

Stability of repair hypotheses and latency Using a beam search of n-best hypotheses on a word-by-word basis can cause 'jitter' in the detector's output. While utterance-final accuracy is desired, for a truly incremental system good intermediate results are equally important. Zwarts et al. (2010)'s time-to-detection results show their system is only certain about a detection after processing the entire repair. This may be due to the string alignment inspired S-TAG that matches repair and reparandum phases: a 'rough copy' dependency only becomes likely once the entire repair has been consumed. The latency of 4.6 words to detection from the repair onset and a relatively slow rise to utterance-final accuracy of reparandum word detection up to 6 words back from the current word is undesirable given repairs have a mean reparandum length of $\approx$1.5 words (Shriberg and Stolcke, 1998, Chapter 4).

Structure identification Classifying repairs has been ignored in repair processing, despite the presence of distinct categories (e.g. repeats, substitutions, deletes) with different pragmatic effects.[2] This is perhaps due to lack of clarity in definition: even for human annotators, verbatim repeats withstanding, agreement is often poor (Shriberg, 1994, Chapter 4). Assigning and evaluating repair (not just reparandum) structures will allow genuine repair interpretation in future; however, work to date evaluates only reparandum detection.[3]

5.2.1 An incremental information-theoretic approach

To address the above challenges, I propose an alternative to (Johnson and Charniak, 2004; Zwarts et al., 2010)'s noisy channel model. While the model elegantly captures intuitions about paral-

[2] Though see Germesin et al. (2008) for one approach, albeit using idiosyncratic repair categories.

[3] The 'correction' measures presented by Heeman and Allen (1999) do not provide a comparable result for assigning the correct reparandum-interregnum-repair positions- see Section 3.1.1.

lelism ('rough copy' dependency) in repeat and substitution repairs and models fluency, it relies on string-matching, motivated in a similar way to automatic spelling correction (Brill and Moore, 2000): it assumes a speaker chooses to utter fluent utterance X according to some prior distribution $P(X)$, but a noisy channel causes them instead to utter a noisy Y according to channel model $P(Y|X)$. Estimating $P(Y|X)$ directly from observed data is difficult due to sparsity of repair instances, so a transducer is trained on the rough copy alignments between reparandum and repair. This approach succeeds overall because repetition and repeat-containing substitution repairs are very common; but I assume, given the evidence, repair as a psychological process is not driven by string alignment (see Chapters 2 and 4), and mid-utterance deletes, restarts and rarer substitution forms will be given low likelihood and are likely to be missed in statistical alignment approaches– see Chapter 4's alignment repair type distributions. Furthermore, the noisy channel model assumes an inherently utterance-global process for generating (and therefore finding) an underlying 'clean' string –much as similar spelling correction models are word-global– instead a very local perspective is taken here.

In accordance with psycholinguistic evidence (Brennan and Schober, 2001), I assume characteristics of the repair onset allow hearers to detect it very quickly and solve the continuation problem (Levelt, 1983) of integrating the repair into their linguistic context immediately, before processing or even hearing the end of the repair phase. While repair onsets may be preceded by interregna, these are not reliable signals, as shown in Chapter 4, occurring in only $\approx$16% of repairs, a figure supported by Heeman and Allen (1999)'s observation of discourse markers in repairs. Due to this lack of lexical signal for repair, I formulate repair onset detection here as driven by recognition of departure from fluency, which I predicted could be achieved by using information-theoretic features of word sequences derived incrementally from a language model. This approach is in line with recent psycholinguistic accounts of incremental language comprehension– see Keller (2004); Jaeger and Tily (2011).

Considering the time-linear way a repair is processed and the fact speakers are exponentially less likely to trace one word further back in computing the reparandum extent as utterance length increases (Shriberg and Stolcke, 1998) and the shortness of repairs, backwards search seems to be the most efficient reparandum extent detection method.[4] Features determining the detection

[4] I acknowledge a purely position-based model for reparandum extent detection under-estimates prepositions, which speakers favour as retrace starts and over-estimates verbs, which speakers tend to avoid as a restart point (Shriberg and Stolcke, 1998, Chapter 2), preferring to begin the utterance again. Healey et al. (2011) also demonstrate this experimentally.

of the reparandum extent in the backwards search can also be information-theoretic: entropy measures of distributional parallelism can capture not only rough copy dependencies, but distributionally similar or dissimilar correspondences between sequences. Finally, when detecting the repair's end point and its structure, distributional information allows computation of the similarity between reparandum and repair. I argue a local-detection-with-backtracking approach is more cognitively plausible than string-based left-to-right repair labelling one, and using this insight should allow an improvement in incremental accuracy, stability and time-to-detection over string-alignment driven approaches in repair detection.

5.3 STIR: Strongly Incremental Repair detection

Given this motivation, the prototype system STIR (Strongly Incremental Repair detection) takes a local incremental approach to detecting repairs and isolated edit terms, assigning words the structures in (5.2). It includes interregnum recognition in the process of edit word detection, due to the inclusion of interregnum vocabulary within edit term vocabulary (Ginzburg, 2012, Chapter 4), a useful feature for repair detection (Lease et al., 2006; Qian and Liu, 2013). STIR therefore assigns the following structures:

$$\begin{cases} ...[rm_{start}...rm_{end} + \{ed\}\ rp_{start}...rp_{end}]... \\ ...\{ed\}... \end{cases} \tag{5.2}$$

Rather than detecting the repair structure in its left-to-right string order as in (5.2), STIR functions as in Figure 5.1, where the top graph shows the input stream of words on top of the edge, a processing graph is in the middle, and the output repair and edit term tags are beneath the words in the top graph: it first detects edit terms (possibly interregna) at step T1; then detects repair onsets rp_{start} at T2; if one is found, backwards searches to find rm_{start} at T3; then finally finds the repair end rp_{end} at T4. Step T1 relies mainly on lexical probabilities from an edit term model; T2 exploits features of divergence from a fluent language model; T3 uses fluency of utterances without the hypothesised reparanda (as per the noisy-channel intuition) and parallelism between repair and reparandum phases; and T4 the similarity between distributions after reparandum and repair end points (indicated by the dotted edge between S3 and S4 in Figure 5.1). Each stage integrates these basic insights via multiple related features in a statistical classifier as will be explained.

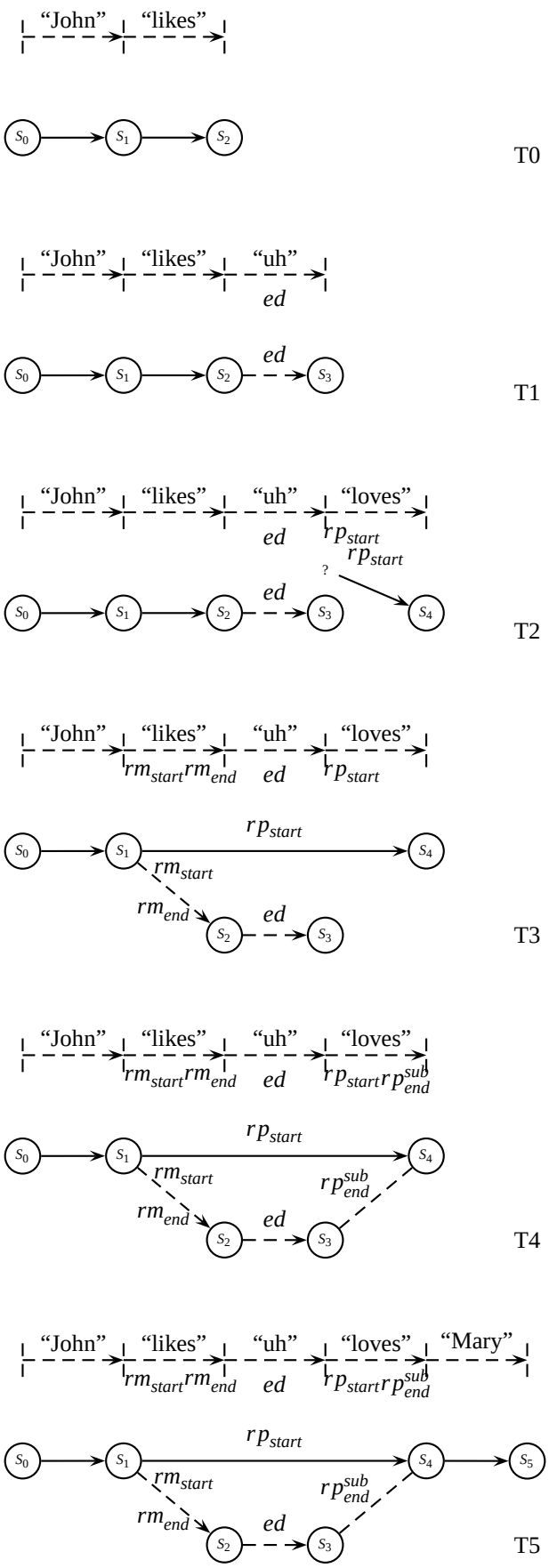


Figure 5.1: Strongly Incremental Repair Detection

5.3.1 Enriched incremental language models

STIR derives the basic information-theoretic features required using n-gram language models, as they have a long history of information theoretic analysis (Shannon, 1948) and provide reproducible results without forcing commitment to one particular grammar formalism. Following recent work on modelling grammaticality judgements (Clark et al., 2013), I implement several modifications to standard language models to develop our basic measures of incremental fluency and uncertainty.

For our main language models two trigram models with Kneser-Ney smoothing (Kneser and Ney, 1995) are trained: one on the words and one on the POS tags of the standard Switchboard training data (`SW_train` from the last chapter, all files with conversation numbers beginning sw2*,sw3* in the Penn Treebank III release).[5] I follow Johnson and Charniak (2004) by cleaning the data of all edit terms and reparanda, to approximate a 'fluent' language model, trained on a total of ≈100K utterances, ≈600K tokens. I call these probabilities p_{kn}^{lex} and p_{kn}^{pos} for the two models below, and if referring to the same calculation being used for both models I suppress the superscripts.

After obtaining the probability values one can derive *surprisal* as the principal default lexical uncertainty measurement s (equation 5.3); and, following Clark et al. (2013), the (unigram) Weighted Mean Log trigram probability (WML, eq. 5.4) – the trigram logprob of the sequence divided by the inverse summed logprob of the component unigrams (apart from the first two words in the sequence, which serve as the first trigram history). As this thesis adopts a local approach to repair detection I restrict the WML measures to single trigrams (so only weighted by the inverse logprob of the final word). This approach was found to be much more effective for detection as it factors out global utterance-level probability. While use of standard n-gram probability conflates syntactic with lexical probability, WML gives an approximation to *incremental syntactic probability* by factoring out lexical frequency.

$$s(w_{i-2}\dots w_i) = -\log_2 p_{kn}(w_i \mid w_{i-2}, w_{i-1}) \tag{5.3}$$

$$WML(w_0\dots w_n) = \frac{\sum_{i=2}^{n} \log_2 p_{kn}(w_i \mid w_{i-2}, w_{i-1})}{-\sum_{j=2}^{n} \log_2 p_{kn}(w_j)} \tag{5.4}$$

[5] In pre-processing POS tags which have a many-to-one relation to words are concatenated into one token. This was for practical purposes but had no significant effect on results.

Distributional measures To approximate uncertainty, one can derive the entropy $H(w|c)$ of the possible word continuations w given a context c, from $p(w_i|c)$ for all words w_i in the vocabulary – see (5.5). Calculating distributions over the entire lexicon incrementally is costly, so this is approximated by constraining the calculation to words which are observed at least once in context c in the language model training, $w_c = \{w|count(c,w) \geq 1\}$, assuming a uniform distribution over the unseen suffixes by using the appropriate smoothing constant, and subtracting the latter from the former– see eq. (5.6).

Manual inspection showed this approximation to be very close to fully calculated entropy; the trie structure of the n-gram models as they are stored in the implementation allows efficient calculation. I also make use of the Zipfian distribution of n-grams in corpora (Manning and Schütze, 1999) by introducing a pre-processing step after the language model has been trained which computes and stores entropy values for the 20% most common trigram contexts observed in training, leaving entropy values of rare or unseen contexts to be computed at decoding time with little search cost due to their small or empty w_c sets.

$$H(w|c) = -\sum_{w \in Vocab} p_{kn}(w|c) \log_2 p_{kn}(w|c) \tag{5.5}$$

$$\begin{aligned} H(w|c) \approx & \left[-\sum_{w \in w_c} p_{kn}(w|c) \log_2 p_{kn}(w|c) \right] \\ & - [n \times \lambda \log_2 \lambda] \\ & \text{where } n = |Vocab| - |w_c| \\ & \text{and } \lambda = \frac{1 - \sum_{w \in w_c} p_{kn}(w|c)}{n} \end{aligned} \tag{5.6}$$

Given this method of approximating distributions, it is similarly efficient to approximate the Kullback-Leibler (KL) divergence (relative entropy) between distributions in two different contexts c_1 and c_2, i.e. $\theta(w|c_1)$ and $\theta(w|c_2)$, by pair-wise computing $p(w|c_1)\log_2(\frac{p(w|c_1)}{p(w|c_2)})$ only for words $w \in w_{c_1} \cap w_{c_2}$, then approximating unseen values by assuming uniform distributions in both distributions. Using p_{kn} smoothed estimates rather than raw maximum likelihood estimations avoids infinite KL divergence values. Again, I found this approximation sufficiently close to the real values for STIR's purposes.

Incremental processing graphs

I employ incrementally constructed directed acyclic graph (DAG) structures such as those in Figure 5.1 with lexical word and POS values and tuples of numeric probability and distributional values stored on the edge labels, akin to word confusion networks in speech recognition. I introduce these structures to satisfy another notion of incrementality set out in Section 3.5.3– the minimisation of re-computation. Storing the unigram probabilities, cumulative *WML*, *p* values and the entropies of continuation $H(w_i|w_{i-2}, w_{i-1})$ at each node *i* as the graphs are constructed word-by-word allows STIR to incrementally calculate these values efficiently for the entire prefix without full re-computation due to the Markov property of n-gram models. This also allows access to previously processed sub-sequences of arbitrary lengths and positions when faced with local ordering changes in the DAG as in reparandum identification (step T3 in Figure 5.1).

5.3.2 Individual classifiers

This section details the features used by the 4 individual classifiers in STIR. To investigate the utility of the features used in each classifier I obtain values on the standard Switchboard heldout data (PTB III files sw4[5-9]*: 6.4K utterances, 49K words) and report feature ranking results, in addition to some mean values and distributions for interesting features. While the edit term (ed) and the repair onset (rp_{start}) classifiers only have one possible training set of features, due to the classifier pipeline, as will be explained, the reparandum start (rm_{start}) and repair end (rp_{end}) classifiers have several possible training sets depending on the cost function setting of each of the classifiers. To reflect the manual feature engineering process, I report the measures obtained for the best overall cost function setting on the heldout data.[6]

Edit term detection

In the first component, I utilise the observation that edit terms have a distinctive vocabulary, training a bigram model on a corpus of all edit words annotated in Switchboard's training data. The classifier simply uses the word surprisal s^{lex} from this edit word model, and the trigram surprisal *s* and syntactic fluency *WML* from the word and POS standard fluent models of Section 5.3.1. I hypothesised that *WML* would be lower for trigrams with an edit term in them, and coupled with the higher probability of the edit words within the p^{lex}_{kn} edit term model (and therefore lower s^{lex} measures), this would give sufficient information to the classifier. A good separation was

[6]The meaning of 'best overall' setting is given technically as the one which achieves the highest Total Score (TS) measure described in Section 5.4.1.

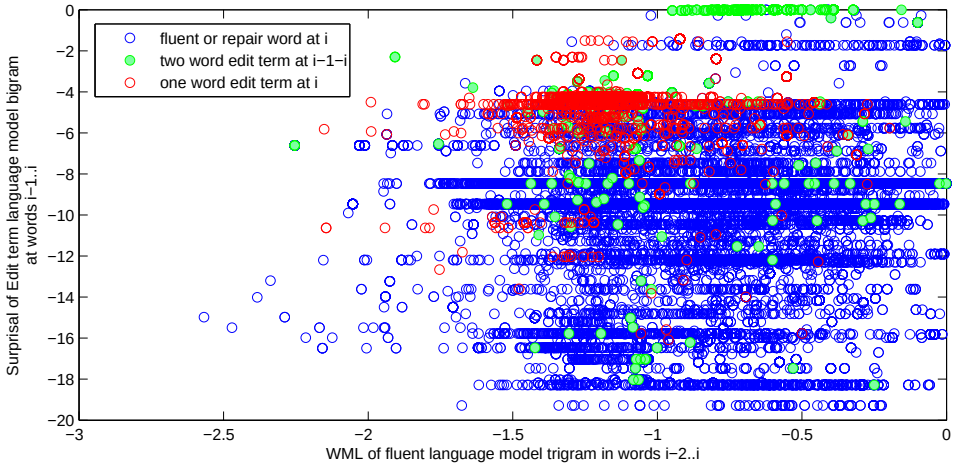


Figure 5.2: WML^{lex} fluent model (X axis) and the inverse s^{lex} edit model measures (Y axis) for the heldout data for one-word edit terms at w_i and two-word edits at $w_{i-1} - w_i$

indeed found for edit terms in the final position w_i and in the span $w_{i-1} - w_i$ in the local trigram history– see Figure 5.2 where the clustering of the one-word (red) and two-word (green) edit terms relative to the values for fluent trigrams (blue) can be seen.

Given the neat separation of values, I restrict the task to classifying at the current position w_i, one, both or none of words w_i and w_{i-1} as edit terms. I found this simple approach effective and stable, detecting edit term words with an F-score of 0.938, performing marginally worse though detecting a broader range of phenomena than Heeman and Allen (1999)'s discourse marker detector. Some delayed decisions occur in cases where s^{lex} and WML^{lex} have similar values in both the edit and fluent language models before the end of the edit, e.g. "I like" $\rightarrow$ "I $\{like\}$ want...". Words classified as *ed* are removed from the incremental processing graph (indicated by the dotted *ed* edge in Figure 5.1) and repair hypotheses are removed if their repair onset coincides with a delayed edit hypothesis at w_{i-1} –this is an aspect of the system which could cause output instability or 'jitter'.

Repair start detection

Repair onset detection is arguably the most crucial component: the greater its accuracy, the better the input for downstream components and the lesser the overhead of filtering false positives required. I use Section 5.3.1's information-theoretic features s, WML, H for word and POS

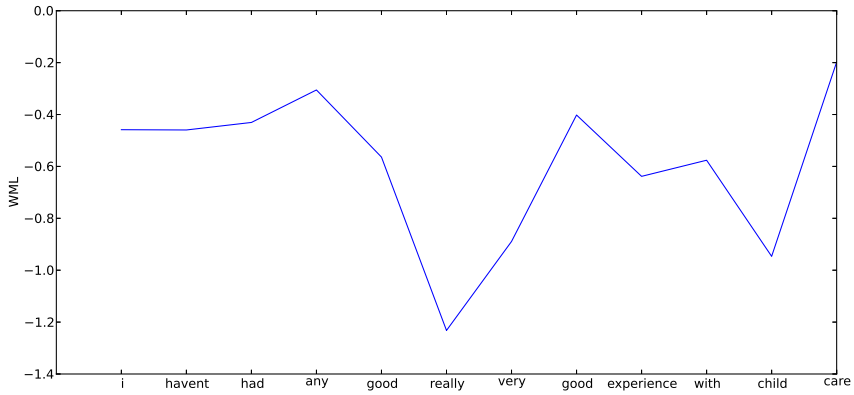


Figure 5.3: WML^{lex} values for trigrams for a repaired utterance exhibiting the drop at the repair onset

models, and introduce 5 additional information-theoretic features: ΔWML is the difference between the WML values at w_{i-1} and w_i; ΔH is the difference in entropy between w_{i-1} and w_i; *InformationGain* is the difference between expected entropy at w_{i-1} and observed s at w_n, a measure that factors out the effect of naturally high entropy contexts; *BestEntropyReduce* is the best reduction in entropy possible by an early rough hypothesis of reparandum onsets within 3 words; and *BestWMLboost* similarly speculates on the best improvement of *WML* possible by positing rm_{start} positions up to 3 words back. I also include simple alignment features: binary features which indicate if the word w_{i-x} is identical to the current word w_i for $x \in \{1,2,3\}$. With 6 alignment features, 16 language model information-theoretic features and a single logical feature *edit* which indicates the presence of an edit word at position w_{i-1}, rp_{start} detection uses 23 features– see Table 5.1. It is worth emphasising here that actual lexical or POS *values* (i.e. words and POS tags) are not used at all in the feature sets, but only these numerical probability derived measures.

I hypothesised repair onsets rp_{start} would have significantly higher s values (lower lexical-syntactic probability) and lower *WML* (lower syntactic probability) than other fluent trigrams. This was the case in the Switchboard heldout data for both measures, with the biggest difference obtained for WML^{lex} (non-repair-onsets: -0.736 (sd=0.359); repair onsets: -1.457 (sd=0.359)) – see Figure 5.4 for the density plot for each class. In the POS model, entropy of continuation H^{pos} was the strongest feature (non-repair-onsets: 3.141 (sd=0.769); repair onsets: 3.444 (sd=0.899)).

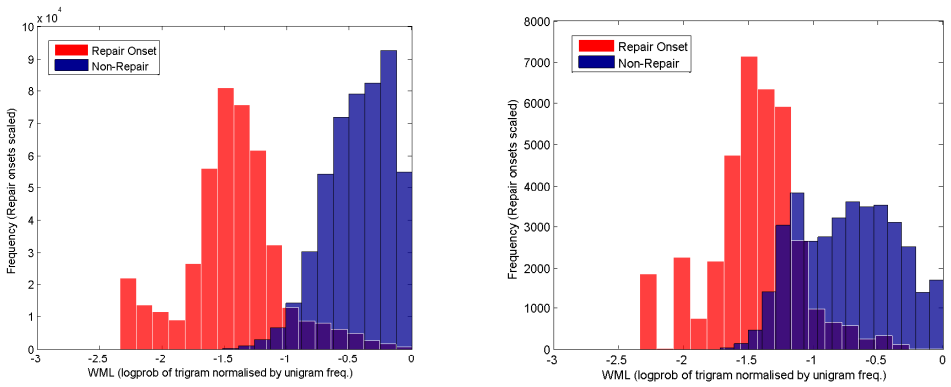


Figure 5.4: WML_{lex} fluency measure density plots for training data (left) and heldout data (right)

The trigram WML^{lex} measure for the repaired utterance "I haven't had any [good + really very good] experience with child care" can be seen in Figure 5.3. The steep drop at the repair onset shows the usefulness of *WML* features for fluency measures.

To compare n-gram measures against other local features, I ranked the features by Information Gain using 10-fold cross validation over the Switchboard heldout data– see Table 5.1. The language model features are far more discriminative than the alignment features, showing the potential of a general information-theoretic approach. Interestingly, the s^{lex} features are not as useful as some of the other information-theoretic features, with WML in both p^{lex} and p^{pos} models being the most discriminative.

Reparandum start detection

In detecting rm_{start} positions given a hypothesised rp_{start}, computing all features from each of the previous six non edit term words, I use the noisy channel intuition that removing the reparandum (from rm_{start} to rp_{start}) increases fluency of the utterance, expressed here as *WMLboost* as described above. On the heldout data, this is shown to be the case, with a mean $WMLboost^{lex}$ of 0.223 (sd=0.267) for reparandum onsets and -0.058 (sd=0.224) for other words in the six-word history. The negative boost for non-reparandum words captures the intuition that backtracking from those points would make the utterance ungrammatical, and conversely the boost afforded by the correct rm_{start} detection helps solve the continuation problem for the listener (and our detector).

Parallelism in the onsets of rp_{start} and rm_{start} can also help solve the continuation problem,

average merit	average rank	attribute
0.139 (+- 0.002)	1 (+- 0.00)	H^{pos}
0.131 (+- 0.001)	2 (+- 0.00)	WML^{pos}
0.126 (+- 0.001)	3.4 (+- 0.66)	WML^{lex}
0.125 (+- 0.003)	4 (+- 1.10)	s^{pos}
0.122 (+- 0.001)	5.9 (+- 0.94)	$w_{i-1} = w_i$
0.122 (+- 0.001)	5.9 (+- 0.70)	BestWMLboost^{lex}
0.122 (+- 0.002)	5.9 (+- 1.22)	InformationGain^{pos}
0.119 (+- 0.001)	7.9 (+- 0.30)	BestWMLboost^{pos}
0.098 (+- 0.002)	9 (+- 0.00)	H^{lex}
0.08 (+- 0.001)	10.4 (+- 0.49)	ΔWML^{pos}
0.08 (+- 0.003)	10.6 (+- 0.49)	ΔH^{pos}
0.072 (+- 0.001)	12 (+- 0.00)	$POS_{i-1} = POS_i$
0.066 (+- 0.003)	13.1 (+- 0.30)	s^{lex}
0.059 (+- 0.000)	14.2 (+- 0.40)	ΔWML^{lex}
0.058 (+- 0.005)	14.7 (+- 0.64)	BestEntropyReduce^{pos}
0.049 (+- 0.001)	16.3 (+- 0.46)	InformationGain^{lex}
0.047 (+- 0.004)	16.7 (+- 0.46)	BestEntropyReduce^{lex}
0.035 (+- 0.004)	18 (+- 0.00)	ΔH^{lex}
0.024 (+- 0.000)	19 (+- 0.00)	$w_{i-2} = w_i$
0.013 (+- 0.000)	20 (+- 0.00)	$POS_{i-2} = POS_i$
0.01 (+- 0.000)	21 (+- 0.00)	$w_{i-3} = w_i$
0.009 (+- 0.000)	22 (+- 0.00)	edit
0.006 (+- 0.000)	23 (+- 0.00)	$POS_{i-3} = POS_i$

Table 5.1: Feature ranker (Information Gain) for rp_{start} detection- 10-fold x-validation on Switchboard heldout data.

and in fact the KL divergence between $\theta^{pos}(w \mid rm_{start}, rm_{start-1})$ and $\theta^{pos}(w \mid rp_{start}, rp_{start-1})$ is the third most useful feature with average merit 0.485 (+- 0.012) in cross-validation. The highest ranked feature is $\Delta WMLboost^{lex}$ (merit=0.516 (+- 0.004)) which here encodes the drop in the *WMLboost* from one backtracked position to the next, and the second ranked feature is the equivalent in the POS model $\Delta WMLboost^{pos}$ (merit=0.505 (+- 0.004). Within the 32 features I use, again information-theoretic ones are higher ranked than the logical features– see Appendix A for the full table.

Repair end detection and structure classification

For rp_{end} detection, which also constitutes assigning the final structure of the repair, using the notion of parallelism, I hypothesise an effect of divergence between θ^{lex} at the reparandum-final word rm_{end} and the repair-final word rp_{end}: for repetition repairs, KL divergence will trivially be 0; for substitutions, it will be higher; for deletes, even higher. Upon inspection of our feature ranking this KL measure ranked 5th out of 23 features (merit= 0.294 (+- 0.006)).

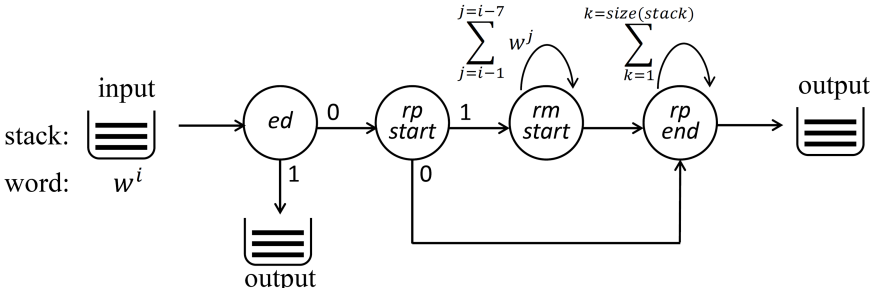


Figure 5.5: STIR's pipeline of classifiers

I introduce another feature encoding parallelism *ReparandumRepairDifference*: the difference in probability between an utterance cleaned of the reparandum and the utterance with its repair phase substituting its reparandum. In the running example from Figure 5.1, this would be as in equation (5.7). In both the POS (merit=0.376 (+- 0.008)) and word (merit=0.364 (+- 0.002)) LMs, this was the most discriminative feature. Again see Appendix A for the full feature ranking table.

$$\begin{aligned} &\textit{ReparandumRepairDifference}(\text{“John [likes + loves]”}) = \\ &\qquad \textit{WML}(\text{“John loves”}) - \textit{WML}(\text{“John likes”}) \end{aligned} \tag{5.7}$$

5.3.3 Classifier pipeline

STIR effects a pipeline of classifiers as in Figure 5.5, where the *ed* classifier only permits non *ed* words to be passed on to rp_{start} classification and for rp_{end} classification of the active repair hypotheses. The active repair hypotheses are maintained in a stack, each consisting of a $\langle rm_{start}, rp_{start}, rp_{end} \rangle$ triple of word positions where the position of rp_{end} may change as more words are consumed. The rp_{start} classifier passes positive repair hypotheses to the rm_{start} classifier, which backwards searches up to 7 words back in the utterance. If a rm_{start} is classified, the output is passed on for rp_{end} classification at the end of the pipeline, and the hypothesis is pushed onto the repair stack, whether an rp_{end} is found or not. The rp_{end} detector may temporarily *cancel* a hypothesis after two words have been consumed beyond the repair onset, which does not remove the hypothesis indefinitely but subdues its effect in its output before searching for more suitable rp_{end} points– this could cause output jitter.

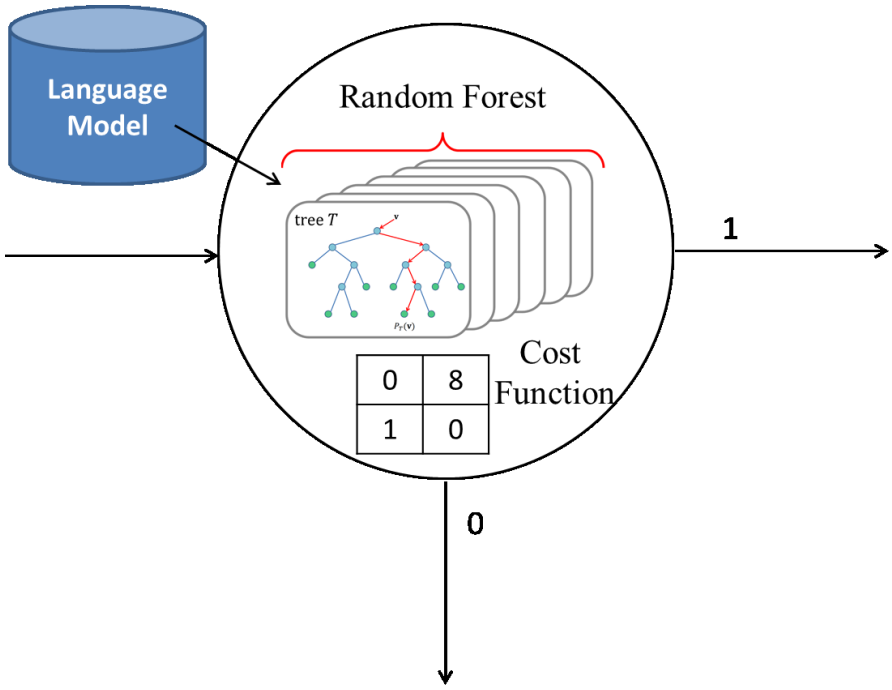


Figure 5.6: An individual STIR classifier

Repair hypotheses are are popped off the stack when the string is 7 words beyond its rp_{start} position. Putting limits on the stack's storage space is a way of controlling for processing overhead and complexity as will be discussed below. Embedded repairs whose rm_{start} coincide with another's rp_{start} are easily dealt with as they are added to the stack as separate hypotheses.[7]

Classifiers Classifiers are implemented using Random Forests (Breiman, 2001) and I use different error functions for each stage using MetaCost (Domingos, 1999), effecting the set-up schematically shown in Figure 5.6 for each one. In early investigation Random Forests, which are a set of decision trees trained simultaneously and in testing they 'vote' on the classification with the majority classification winning, gave much better performance than single decision tree classifiers. The flexibility afforded by implementing adjustable error functions in a pipelined incremental processor allows control of the trade-off of immediate accuracy, run-time, stability and final accuracy of the sequence classification.

Processing complexity The pipeline avoids exhaustively searching all repair hypotheses. If the search is limited to within the $\langle rm_{start}, rp_{start} \rangle$ possibilities, the number of possible repairs grows

[7] I constrain the problem not to include embedded deletes which may share their rp_{start} word with another repair – these are in practice very rare – see Chapter 4.

approximately in the triangular number series– i.e. $\frac{n(n+1)}{2}$, a nested loop over previous words as n gets incremented, which in terms of a complexity class is a quadratic $O(n^2)$. If more than one $\langle rm_{start}, rp_{start} \rangle$ hypothesis is permitted per word, the complexity goes up to $O(n^3)$, however, as will be shown in the tests that I describe below, STIR is able to achieve good detection results without permitting this extra search space. Under the assumption that reparandum onset detection is only triggered after repair onset detection, and repair extent detection is dependent on positive reparandum onset detection, a pipeline with accurate components will allow STIR to limit its processing to a small subset of this search space.

5.4 Experimental set-up

STIR is trained on the Switchboard training data described above, and tested on the standard Switchboard test data (PTB III files 4[0-1]*) with partial words and punctuation removed from all files for fair comparison to other systems. In order to avoid over-fitting of classifiers to the basic language models, I use a cross-fold training approach: the corpus is divided into 10 folds and language models trained on 9 folds are used to obtain feature values for the 10th fold, repeating for all 10. The Random Forest classifiers are then trained as standard on the resulting feature-annotated corpus. This cross-fold method resulted in better feature utility for n-grams and better F-score results for detection in all components in the order of 5-6%.[8]

Training the classifiers Each Random Forest classifier was limited to 20 trees of maximum depth 4 nodes, putting a ceiling on decoding time. In making the classifiers cost-sensitive, MetaCost re-samples the data in accordance with the cost-functions: I found using 10 iterations over a re-sample of 25% of the training data gave the most effective trade-off between training time and accuracy. As Domingos (1999) demonstrated, there are only relatively small accuracy gains when using more than this, but with the cost of training time increasing in the order of the re-sample size. I only use one cost setting for *ed* as changing this did not have a noticeable effect on results, however I use 8 different cost-functions in rp_{start} with differing costs for false negatives of the form below, where R is a repair onset and F is a fluent onset:

[8] A similar approach was taken for Switchboard data in Zwarts and Johnson (2011) for training a re-ranker of repair analyses.

$$\begin{array}{c} \\ R^{gold} \\ F^{gold} \end{array} \begin{array}{c} \begin{array}{cc} R^{hyp} & F^{hyp} \end{array} \\ \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} \end{array}$$

I adopt a similar technique in rm_{start} using 5 different cost functions and in rp_{end} using 8 different settings, which when combined gives a total of 320 different cost function configurations. I hypothesise that higher recall permitted in the pipeline's first components would result in better overall accuracy as these hypotheses become refined, though at the cost of the stability of the hypotheses of the sequence and extra downstream processing in pruning false positives.

I also experiment with the number of repair hypotheses that can be added to the stack per word, experimenting with limits of 1-best, 2- and 3-best hypotheses. I expect that allowing 2 or more hypotheses to be explored per rp_{start} should allow greater final accuracy, but at the expense of greater decoding and training complexity (theoretically this goes up from quadratic to cubic as described above), and possible incremental instability in its output.

In addition to testing accuracy in the standard way, I wish to explore the incremental performance versus final accuracy trade-off that STIR can achieve, so I now describe the evaluation metrics I employ that measure this.

5.4.1 Incremental evaluation metrics

Following Baumann et al. (2011) I divide the evaluation metrics into *similarity metrics* (measures of equality with or similarity to a gold standard), *timing metrics* (measures of the timing of relevant phenomena detected from the gold standard) and *diachronic metrics* (evolution of incremental hypotheses over time).

Similarity metrics For direct comparison to previous approaches I use the standard measure of overall accuracy, the F-score over reparandum words, which I abbreviate $\mathbf{F_{rm}}$ (see 5.8):

$$\begin{aligned} \text{precision} &= \frac{rm^{correct}}{rm^{hyp}} \\ \text{recall} &= \frac{rm^{correct}}{rm^{gold}} \\ \text{F}_{rm} &= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \end{aligned} \tag{5.8}$$

I am also interested in repair structural classification given the different functions possible in repair shown in the last chapter, therefore I also measure F-score over *all* repair components

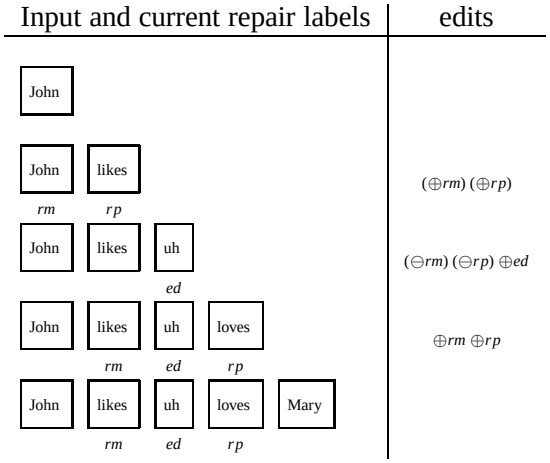


Figure 5.7: Edit Overhead- 4 unnecessary edits

(*rm* words, *ed* words as interregna and rp words), a metric I abbreviate $\mathbf{F_s}$. This is not measured in standard repair detection on Switchboard. To investigate incremental accuracy I evaluate the *delayed accuracy* (**DA**) introduced by Zwarts et al. (2010), as described in Section 3.1.3 against the utterance-final gold standard disfluency annotations of reparandum words, and use the mean of the 6 word F-scores.

Timing and resource metrics Again for comparative purposes I use Zwarts et al's *time-to-detection* metrics, that is the two average distances (in numbers of words) consumed before first detection of gold standard repairs, one from rm_{start}, $\mathbf{TD_{rm}}$ and one from rp_{start}, $\mathbf{TD_{rp}}$. In STIR's 1-best stack setting, before evaluation I know a priori TD_{rp} will be 1 token, and TD_{rm} will be 1 more than the average length of $rm_{start} - rp_{start}$ repair spans correctly detected. However when I introduce a beam where multiple rm_{start}s are possible per rp_{start} with the most likely hypothesis committed as the current output, the latency may begin to increase: the initially most probable hypothesis may not be the correct one. In addition to output timing metrics, I account for intrinsic processing complexity with the metric *processing overhead* (**PO**), which is the number of classifications made by all components per word of input.

Diachronic metrics To measure stability of repair hypotheses over time I use Baumann et al. (2011)'s *edit overhead* (**EO**) metric. EO measures the proportion of edits (add, revoke, substitute) applied to a processor's output structure that are unnecessary. STIR's output is the repair label sequence shown in Figure 5.1, however rather than evaluating its EO against the current gold

$$\begin{array}{c} \\ rp^{gold}_{start} \\ F^{gold} \end{array}\begin{array}{c} \begin{array}{cc} rp^{hyp}_{start} & F^{hyp} \end{array} \\ \begin{pmatrix} 0 & 64 \\ 1 & 0 \end{pmatrix} \end{array} \quad \begin{array}{c} \\ rm^{gold}_{start} \\ F^{gold} \end{array}\begin{array}{c} \begin{array}{cc} rm^{hyp}_{start} & F^{hyp} \end{array} \\ \begin{pmatrix} 0 & 8 \\ 1 & 0 \end{pmatrix} \end{array} \quad \begin{array}{c} \\ rp^{gold}_{end} \\ F^{gold} \end{array}\begin{array}{c} \begin{array}{cc} rp^{hyp}_{end} & F^{hyp} \end{array} \\ \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} \end{array} \quad \text{Stack depth = 3}$$

$$\begin{array}{c} \\ rp^{gold}_{start} \\ F^{gold} \end{array}\begin{array}{c} \begin{array}{cc} rp^{hyp}_{start} & F^{hyp} \end{array} \\ \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} \end{array} \quad \begin{array}{c} \\ rm^{gold}_{start} \\ F^{gold} \end{array}\begin{array}{c} \begin{array}{cc} rm^{hyp}_{start} & F^{hyp} \end{array} \\ \begin{pmatrix} 0 & 16 \\ 1 & 0 \end{pmatrix} \end{array} \quad \begin{array}{c} \\ rp^{gold}_{end} \\ F^{gold} \end{array}\begin{array}{c} \begin{array}{cc} rp^{hyp}_{end} & F^{hyp} \end{array} \\ \begin{pmatrix} 0 & 8 \\ 1 & 0 \end{pmatrix} \end{array} \quad \text{Stack depth = 1}$$

Figure 5.8: The cost function settings for the MetaCost classifiers for each component, for the best F_{rm} setting (top row) and best total score (TS) setting (bottom row)

	F_{rm}	F_s	DA	EO	PO
Best Final *rm* F-score (F_{rm})	**0.781**	0.736	0.702	4.043	1.733
Best Final repair structure F-score (F_s)	0.772	**0.737**	0.707	4.535	1.660
Best Delayed Accuracy of *rm* (DA)	0.767	0.721	**0.718**	1.483	1.689
Best (lowest) Edit Overhead (EO)	0.718	0.674	0.675	**0.864**	1.230
Best (lowest) Processing Overhead (PO)	0.716	0.671	0.673	0.875	**1.229**
Best Total Score (mean % of best scores) (TS)	*0.754*	*0.708*	*0.711*	*0.931*	*1.255*

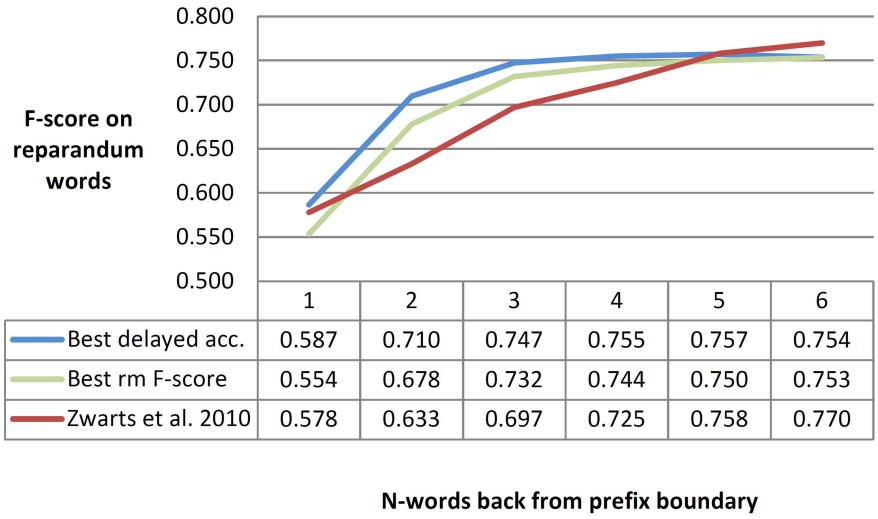
Table 5.2: Comparison of the best performing system settings using different measures

standard labels, I use a new mark-up I term the *incremental repair gold standard*: this does not penalise lack of detection of a reparandum word *rm* as a bad edit until the corresponding rp_{start} of that *rm* has been consumed. While F_{rm}, F_s and DA evaluate against what Baumann et al. (2011) call the *current gold standard*, the incremental gold standard reflects the repair processing approach I set out in 5.2. An example of a repaired utterance with an EO of 44% ($\frac{4}{9}$) can be seen in Figure 5.7: of the 9 edits (7 repair annotations and 2 correct fluent words), 4 are unnecessary (bracketed). Note the final $\oplus$*rm* is not counted as a bad edit for the reasons just given. .

5.5 Switchboard Results and Discussion

I evaluate on the Switchboard test data; Table 5.2 shows results of the best performing settings for each of the metrics described above, together with the setting achieving the highest total score (TS)– the average % achieved of the best performing system's result in each metric.[9] The settings found to achieve the highest $F_{\mathbf{rm}}$ (the metric standardly used in disfluency detection), and that found to achieve the highest TS for each stage in the pipeline are shown in Figure 5.8.

[9]We do not include time-to-detection scores in TS as it did not vary enough between settings to be significant, however there was a difference in this measure between the 1-best stack, 2-best and 3-best stack conditions – see below.



	1	2	3	4	5	6
Best delayed acc.	0.587	0.710	0.747	0.755	0.757	0.754
Best rm F-score	0.554	0.678	0.732	0.744	0.750	0.753
Zwarts et al. 2010	0.578	0.633	0.697	0.725	0.758	0.770

Figure 5.9: Delayed Accuracy Curves

The experiments showed that different system settings perform better in different metrics, and no individual setting achieved the best result in all of them. The best utterance-final F_{rm} reaches 0.781, marginally though not significantly exceeding Zwarts et al. (2010)'s measure and STIR achieves 0.737 on the previously unevaluated F_s. The setting with the best DA improves on Zwarts et al. (2010)'s result significantly in terms of mean values (0.718 vs. 0.694), and also in terms of the steepness of the curves as Figure 5.9 shows. The fastest average time to detection in a setting is 1 word for TD_{rp} and 2.6 words for TD_{rm}, these scores remaining fairly invariant across all settings, improving dramatically on the noisy channel model's 4.6 words and 7.5 words.

Incrementality versus accuracy trade-off I aimed to investigate how well a system could do in terms of achieving both good final accuracy and incremental performance, and while the best F_{rm} setting had a large PO and relatively slow DA increase, STIR can find a good trade-off setting: the highest TS scoring setting achieves an F_{rm} of 0.754 whilst also exhibiting a very good DA (0.708) – over 98% of the best recorded score – and low PO and EO rates – over 96% of the best recorded scores. See the bottom row of Table 5.2. As can be seen in Figure 5.8, the cost functions for these winning settings are different in nature. The best non-incremental F_{rm} measure setting requires high recall for the rest of the pipeline to work on, using the highest cost, 64, for false negative rp_{start} words and the highest stack depth of 3 (similar to a wider beam); but the best overall TS scoring system uses a less permissive setting to increase incremental performance.

	F_{rm}	F_s	DA	EO	PO	TD_{rp}	TD_{rm}
1-best rm_{start}	0.745	0.707	0.699	3.780	1.650	1.0	2.6
2-best rm_{start}	0.758	0.721	0.701	4.319	1.665	1.1	2.7
3-best rm_{start}	0.758	0.721	0.701	4.341	1.666	1.1	2.7

Table 5.3: Comparison of performance of systems with different stack capacities

I make a preliminary investigation into the effect of increasing the stack capacity by comparing stacks with 1-best rm_{start} hypotheses per rp_{start} and 2-best and 3-best stacks. The average differences between the three conditions for the 3 most permissive rp_{start} settings is shown in Table 5.3. I did not test all settings as I assume the stack size increase will not help in the less permissive settings where the hypotheses will not be able to be added.

Moving to the 2-best condition results in gain in overall accuracy in F_{rm} and F_s, but at the cost of EO and also time-to-detection scores TD_{rm} and TD_{rp}. There is no further gain in any accuracy score when moving to the 3-best condition with marginally more PO and EO. The highest F_{rm} scoring system was from within the 3-best condition. The extent to which the stack can be increased and used for useful increase without increasing jitter, latency and complexity will be investigated in future work.

5.6 Adaptation to out-of-domain data: clinical psychology

While the computational linguistics community focusses on the Switchboard disfluency challenge, begun by Charniak and Johnson (2001), the effort to use the models outside of the domain has been rare. This is possibly because gold-standard disfluency annotation in the format shown in (5.1) is rare, and in fact, Switchboard provides the only consistently annotated large corpus available for this purpose. Furthermore, utterance segmentation as carried out by the Switchboard disfluency scheme (Meteer et al., 1995) is also rare.

To investigate the extent to which STIR performs on out-of-domain data, I apply it to an a corpus of psychiatric consultations. As described in Section 2.4, repair rate can help with prediction of schizophrenic patients' adherence to medical treatment, so any reliable way of automating the repair detection will be of practical use.

5.6.1 Clinical Data

The clinical corpus was constructed using a subset of data from a study investigating clinical encounters in psychosis (McCabe et al., 2013), collected between March 2006 and January 2008.[10] The corpus consists of transcripts from 51 outpatient consultations of patients with schizophrenia and their psychiatrist. These transcripts relate to 51 different patients, and 17 different psychiatrists. The consultations varied in length, with the shortest consisting of only 709 words (lasting approximately 5 minutes), and the longest 8526 (lasting nearly an hour). The mean length of consultation was 3500 words.

Each transcript was hand-annotated for repair using the protocol described in (Healey et al., 2005). For the purposes of this study, the data extracted consisted of the transcripts and associated position 1 self-repairs (annotated with reparandum phrase and corresponding repair phase). Filled pauses are not explicitly annotated, but are identifiable as interregna as the unannotated text between the end of the reparanda and its repair. Filled pauses, while consistently transcribed, were found to be inconsistently spelt between transcribers (*aammm*, *er*, *eerrrrmm*, *uhmmm* etc). A find-and-replace operation was therefore applied to the corpus prior to analysis to give these a standardised spelling, i.e. a consistent 'er'. Prior to the analysis, the corpus was also POS-tagged using the Stanford POS tagger (Toutanova et al., 2003). The Stanford tagger is trained on written text, and previous work applying it to spoken dialogue has shown the error rates to be in the order of 10% (Mieskes and Strube, 2006). Here, the concern is not with the POS labels per se, but in the parallelism between POS label sequences as described above– given that errors are likely to be fairly consistent (dependent on transcription spelling or spoken dialogue idiosyncracies) this is sufficient for the purposes of this study.

5.6.2 Adjusting STIR to deal with partial words

I begin by optimizing performance on the Switchboard data with partial words included, which was not the case for the tests described above. I include a penalising factor for WML to be -3.0 (below the lowest recorded value) when a partial word is transcribed in the second position of the trigram, or if it is an unknown word, that it is a prefix of its following word. This was because the corpus study in Chapter 4 suggests that a non-utterance-final partial word presence

[10] This sub-section was largely written by Chris Howes and is from the paper (Howes et al., 2014). It is included here for coherence. Chris also performed the pre-processing steps on the clinical corpus and the correlation results.

predicts a disfluency almost perfectly, that is $p(rp_{start} \mid partial) = 0.935$, for multi-word as well as single partial-word disfluent cut-offs. For incremental detection modelling purposes, I make the realistic assumption that a word w_i can only be interpreted as an abandoned partial word after having encountered the following word w_{i+1}.

A more principled approach to modelling partial words can be taken which I will now explain, but hedge with the fact this has not been tested and the simple approach just described is used in practice: It is possible to train a simple word completion model $p^{complete}(w|w_i)$ which operates on any annotated partial word prefix w_i to provide a distribution over possible complete words that it could have started, and thus also the most likely completion (based on the prefix and unigram co-occurrence). So, as opposed to leaving the partial word as unknown vocabulary item (or with a set probability) it is possible to define a probability distribution of the completion probability of each word in the vocabulary, for which I posit the probability function p^{fluent} that for a partial word w_i, the likelihood of w being its corresponding complete word at the time of interruption given its two word context is as in (5.9).

$$\begin{aligned} p^{fluent}(w \mid w_{i-2}, w_{i-1}, w_i) = \frac{1}{Z} \times p_{kn}(w \mid w_{i-2}, w_{i-1}) \\ \times p^{complete}(w \mid w_i) \end{aligned} \tag{5.9}$$

where Z is a standard normalisation constant to ensure that $\sum_{w \in Vocab} p^{fluent}(w \mid w_{i-2}, w_{i-1}, w_i) = 1$.

The probability $p^{\hat{fluent}}$ of most likely completion of w_i is then as in (5.10).

$$p^{\hat{fluent}}(w \mid w_{i-2}, w_{i-1}, w_i) = \max_w p^{fluent}(w \mid w_{i-2}, w_{i-1}, w_i) \tag{5.10}$$

The intuition here is that when they encounter a partial word hearers attempt to find the most likely fluent word that both maximises its likelihood to be its complete form of the partial word and also of being a continuation of the two preceding words. If “yes I remem-” is encountered, the probability of the completer’s best guess will not be as low as if it was unpredictable, such as after an utterance initial “T-”. Furthermore, the information-theoretic information from these distributions should help predict the interpretation of the repair. I predict repairs with reparandum-final partial words rm_{end} with high entropy over possible completions $\theta^{fluent}(w \mid rm_{end-2}, rm_{end-1}, rm_{end})$ (i.e. the distribution over all words in the vocab of the probability function (5.9)) will be interpreted as deletes rather than substitutions– in deletes the high uncertainty over predicted complete words is interpreted as ‘cancelled’. Partial word reparandum ends interpreted as substitutions should give high p^{fluent} values for w the repair word in cases

such as "yes I [remem- + remember]" and the information gain that *w* causes relative to the distribution $\theta^{fluent}(w \mid rm_{end-2}, rm_{end-1}, rm_{end})$ will be low, where as in delete partial words such as "[T- +] yesterday was nice" the information gain will be much higher relative to the predicted distribution.

This method is yet to be implemented but in initial testing seems a promising approach. Now I will return to STIR's functioning on the clinical data.

5.6.3 Error functions and edit terms in out-of-domain data

I employed the weighted MetaCost error functions described above to balance recall and precision in the desired way for the task. This allows fine-grained control over the rate of onset prediction, which proved to be very useful for the clinical data. For the Switchboard test setting, I optimise the cost functions for MetaCost on the standardly used Switchboard heldout data with partial words included. For the PCC data, while I keep the base classifier the same as for Switchboard, I optimise the weights to balance precision and recall on a heldout set of doctor-patient interaction of ≈20K words. This step was carried out as the weights used for Switchboard yielded much higher precision than recall in rp_{start} detection on a word-by-word level. I then test on a different set of ≈25K words.

For edit term detection on the PCC data, transcribed filled pauses are automatically tagged as edit terms and then edit term detection is performed using a model trained on the Switchboard data– this serves as an approximation to edit term detection; due to the lack of gold standard this was not evaluated quantitatively, but see below for discussion.

5.6.4 Evaluation

I then tested STIR in terms of its precision, recall and F-score for repair onset rp_{start} detection only (rather than reparandum words as above) to make the comparison fair given the otherwise differing annotation conventions. Also, for this test I evaluate in two ways: a *strict* evaluation at the word level, requiring the exact repair point word rp_{start} to be identified; and a *relaxed* evaluation at the turn level, with a rp_{start} hypothesis taken as correct if in the same turn as a gold-standard repair annotation, but with every additional hypothesised rp_{start} over the correct number treated as a false positive (i.e. incrementing the counts of rp_{start}s hypothesised but not rp_{start}s correct).

detection	precision	recall	F-score
rp_{start} position	0.882	0.797	0.837
repairs in turn	0.930	0.793	0.856

Table 5.4: Switchboard test data results

detection	precision	recall	F-score
rp_{start} position	0.527	0.536	0.532
repairs in turn	0.682	0.679	0.680

Table 5.5: Clinical data test results

5.6.5 Results: Clinical data and partial words Switchboard data

The best performing setting's performance for rp_{start} detection for the Switchboard test data including partial words is in Table 5.4 and that for the clinical data shown in Table 5.5.[11] While not the focus of this comparison, it is worth mentioning performance on overall repair structure detection was better with partial word information included compared to performance on partial words excised-data described in Section 5.5, with a best F_{rm} score of 0.783 and best F_s score of 0.751.

Turn-level data As can be seen in Table 5.4, on the Switchboard data (partial words included, so not the same as in the previous results in the chapter) the system identifies both that there is a repair and its exact position in the turn very well (F-scores > 0.8). However, for the PCC data (see Table 5.5), although the system identifies that there are repairs in the turn reasonably well (F-score ≈ 0.7), there is a large drop in performance when looking at the strict position-based metric (F-score ≈ 0.5).

This is likely to be due to differences in both transcription and annotation conventions. In the PCC data, the emphasis for annotators was on identifying the number and type of repairs in the turn. Although there was good agreement between annotators at this level – with levels comparable to the relaxed evaluation performance (Cohen's $\kappa = 0.73$, McCabe et al. (2013)), it is not clear whether the annotators position repair points systematically or agree on positioning. Examination of the transcripts suggests that annotation differences can abound.

Dialogue level data Given the differences in turn-level data, as outlined above, and the different ways in which automatically annotated repair data might be used, the number of identified repairs

[11] The results in Table 5.4 are improvements on Howes et al. (2014) as the full feature set is used here.

	Hand-coded		Automatic		Correlation	
	Mean	(s.d.)	Mean	(s.d.)	r	p
Patient P1 repair	62.51	(44.87)	48.90	(33.29)	0.945	< 0.001
Doctor P1 repair	41.57	(23.25)	41.02	(23.23)	0.906	< 0.001

Table 5.6: Relationship between hand-coded and automatically generated repair measures

over each dialogue were compared.[12]

As can be seen from Table 5.6, there is a very high correlation (> 0.9) between the number of repairs per transcript detected by the automatic incremental classifier and those annotated by hand. At this coarse-grained level, the system provides a useful overview of self-repair, which allows us to make comparisons between speakers who typically use a lot of repair and those who do not, as well as looking for associations with outcomes on a by-patient level as in McCabe et al. (2013). However, as can also be seen, the automatic repair numbers are lower than those for the hand-coded data, and this is especially the case where patients are concerned. This indicates that the system is systematically *not* picking up certain types of repair that the patients are using.

When comparing the hand annotations on the PCC data with STIR's output, we see differences due to several factors of annotation protocol and behaviour and not just due to inherently poor system performance. See examples (5.11)-(5.13) where the hand annotation tags (shown in (a) in each case) differ from STIR's annotations (shown in (b)).

(5.11) **(a) D:** ... and if you tell me that **that**$[RP_{START}]$ that the depressions kicks in ...

(b) D: ... and if you tell me that **that**$[rp_{start}]$ **that**$[rp_{start}]$ the depressions kicks in ...

(5.12) **(a) D:** and so **I**$[RP_{START}]$ mean otherwise I'm not too concerned about your mental health...

(b) D: and so **I**$[ed]$ **mean**$[ed]$ otherwise I'm not too concerned about your mental health...

(5.13) **(a) P:** I don't **I'm**$[RP_{START}]$ not like hearing voices...

(b) P: I don't I'm not like hearing voices...

In (5.11a) the second repeat of 'that' is evaluated as a false positive by STIR, reflecting the embedded repairs often found in Switchboard, while the annotator views this as part of one

[12]Again I acknowledge Chris Howes for the running of these tests and her writing contribution to this subsection.

longer repair. A false negative from STIR can be seen in (5.12b) where an annotator deems this a repair, while according to Switchboard, and STIR, this would be an editing phrase 'I mean'. In (5.13c), another false negative is evaluated as STIR misses the transcribed repair onset from 'I'm not'. These types of self-repair, 'restarts', or utterance-initial deletions, are not marked in Switchboard, as discussed above, treated as two separate utterance units, and so it suffers from lack of training data for these.

5.6.6 Discussion

In summary, STIR can detect self-repair reliably across modalities and domains, but only under a relaxed evaluation metric. However, this is sufficient for the purposes of examining overall rates of repair, as used in some clinical studies (McCabe et al., 2013), and automatic self-repair detection using STIR can therefore be usefully applied to these datasets, removing the need for time-consuming and costly hand annotations.

Using a more strict word-by-word evaluation, differences in annotation schemes and transcription conventions have a marked effect on the system's performance. Switchboard annotation conventions result in a biasing on particular types of repair, namely, mid-utterance repetitions, deletions and substitutions, whereas it is not marked for restarts, which caused it to perform poorly on detecting them in the clinical data. On the clinical side, the fact that editing terms are often marked as the repair onset means a Switchboard-trained detector will not detect these, or if detecting them as interregna will not get the precise position of the repair onset as marked. This has implications for the generalisability of all repair detection systems that rely on strict word-by-word evaluation, such as those used in dialogue systems – the way in which the training data has been annotated and transcribed will affect what types of repair it reliably detects.

The advantage STIR has over other systems practically speaking is its modularity– different phases of the detection cycle may be omitted if there is no data to train on. Results can still be reasonable, and useful for health professionals, even if this is not the same level of mark-up as the Switchboard data.

5.6.7 Towards domain general repair detection

Despite the differences in the type of disfluency annotation available, one can build a system that is practically useful for detection purposes using the set-up as shown in Figure 5.10. As long as there is some heldout data available of the same type as the target corpus, even if not considerable

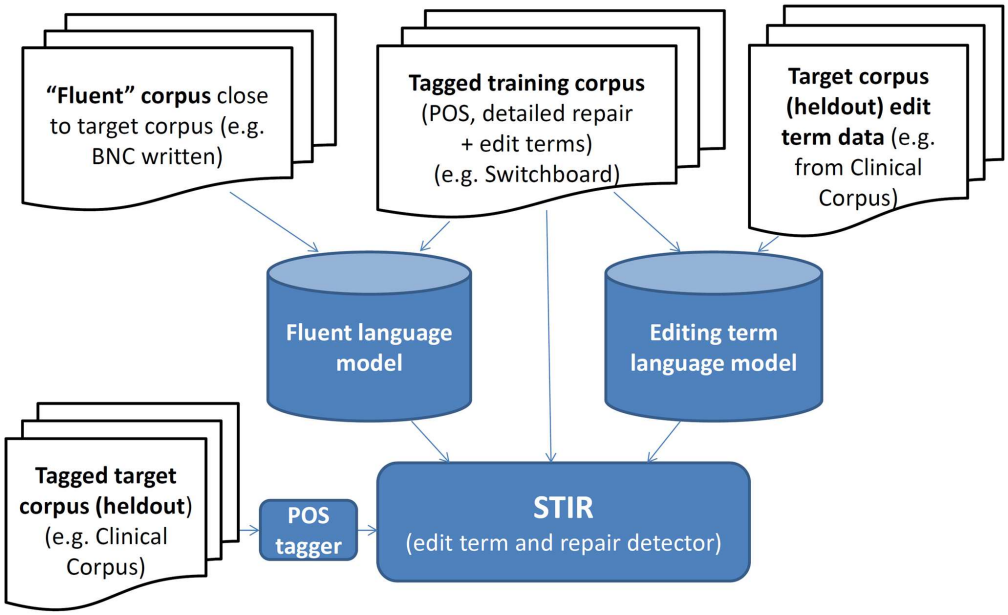


Figure 5.10: STIR training and heldout sources for a new target domain

in size, STIR's error functions can be manually adjusted (or automatically experimented with) to yield the best accuracy results before testing. This technique is effective in terms of giving results with good overall correlations as described above.

The element of Figure 5.10 not present in the version of STIR here is the *"fluent" corpus* which could form additional training data to the fluent language model in STIR. I hypothesize that the appropriate data, even if from written, rather than spoken sources, could boost results on out-of-domain (non-Switchboard) data. Zwarts and Johnson (2011) show how large text-based corpora included in a repair hypotheses re-ranker can improve detection on Switchboard, however I would like to explore the effect of additional resources in improving performance on other data, such as the PCC corpus described here. Other data STIR does not currently use is acoustic information, which has been shown to help disfluency recognition by improving partial word detection (Liu, 2004). Incorporating speech signal information will form part of future work.

5.7 Conclusion

The chapter has presented STIR, a state-of-the-art incremental repair detector that can be used to experiment with incremental performance and accuracy trade-offs. I have shown its efficacy on the Switchboard test data. Its primary features are information-theoretic ones which are accessed in a psycholinguistically plausible time-linear process. The word-by-word incrementality and efficiency for which it was designed allows it to operate with no latency and high incremental accuracy, which will be a useful feature for interactive dialogue systems.

STIR also shows some promise in becoming a domain-general incremental repair detector and therefore of practical use, in this case being used in psychiatric consultation transcripts though it is also currently being employed for other repair annotation tasks. In future work in addition to incorporating acoustic data, I plan to include probabilistic and distributional features from a top-down incremental parser e.g. (Roark et al., 2009), and use STIR's distributional features to classify repair type to make it more robust and explore the information-theoretic paradigm for incremental processing further.

Chapter 6

An Incremental Semantics Driven Model of Self-Repair Processing

---

In this chapter I describe an abstract formal model for incremental self-repair processing as it functions within Natural Language Understanding (NLU) and Natural Language Generation (NLG) in a dialogue model. I also describe methods for implementing this model into the dialogue system DyLan (Purver et al., 2011, Section 3.3). I argue that the framework and implementational methods laid out here address lacunae in previous formal computational models of self-repair described in Chapter 3.[1]

6.1 Self-repair requirements for an incremental dialogue model

For a dialogue model or system to have incremental self-repair processing capability, given the empirical evidence hitherto presented, it should be able to parse and generate repaired utterances like the following at no greater processing cost, and with the same degree of strong incremental interpretation and grain of incremental representation as for fluent utterances, without filtering the effects on fluency (the edit terms and reparanda) out of the input:

(6.1) "But one of [the, + the] two things that I'm really…"

Repeat (sw4356)

(6.2) "John goes to Paris, { uh,} from London"

[1] The work here is an extension of (Hough, 2011) and (Hough and Purver, 2012).

Extension (or forward-looking disfluency) as an editing term (constructed example)

(6.3) "…but my kids are only elementary [grades, + levels] right now" *Substitution (sw4325)*

(6.4) "…the bank was suing them [for, + { uh,}] because they went to get…"
Delete (sw4356)

(6.5) "[the interview was, + {…} it was] alright"
Substitution with continued access to the reparandum (Clark, 1996, p.266)

(6.6) "Peter went [swimming with Susan, + {or rather,} surfing] yesterday"
Substitution requiring ellipsis resolution using the reparandum (constructed example from anonymous SemDial 2012 reviewer)

6.1.1 NLU requirements

Several desiderata for self-repair processing in NLU can be inferred from these examples, given the demands of incrementality argued for in the previous chapters.

The first requirement is that self-repair processing should be strongly incremental. Repairs should be detected and assigned a suitable representation immediately upon the repair onset with minimal latency, as the STIR system described in the previous chapter was capable of on a structural level.

Secondly, and perhaps the core claim of this thesis, NLU should be able to incrementally interpret the type of repair made in terms of its contribution to the *meaning* of the utterance in the dialogue situation; this information should be made available with strongly incremental interpretation (maximal semantic information) within an appropriate framework, and not in any way be filtered out. While it should be capable of dealing with the surface forms of edit terms, repetitions, substitutions and deletes, or more complex subtypes of these as described in Chapter 4, these should processed in terms of their meaning and dialogue function, rather than surface form– this chapter therefore takes a semantics-first approach. For example, NLU must incrementally be able to interpret isolated edit terms as indicating forward-looking trouble for the speaker, and other repairs as backward-looking trouble sources for particular parts of the utterance, in line with Ginzburg et al. (2014)'s proposal for differentiating the interactive meaning of the two types.

Thirdly, NLU is required to keep track of the semantic increments it has derived in reparanda,

e.g. in (6.5),"the interview" needs to remain accessible when parsing "it" for anaphora resolution to function, so it must also keep track of processing context and re-use parts of the semantic and syntactic context built up by utterances appropriately. This is particularly the case in repairs containing ellipsis like (6.6), where, given a suitable context, "with Susan" should be incorporated into the asserted information rather than discarded and can be seen as an element present implicitly when processing the repair phase "surfing".

Fourthly, given the possible parsing ambiguity in this last example, an NLU model must tightly interact with dialogue context, which may be available from other sources of discourse information in the dialogue framework, in order to select the most likely interpretation. So the repair interpretation process, whilst clearly having a close interaction with syntactic processing, should function in an interleaved fashion by querying the 'higher-level' conceptualisation part of the framework. In the same vein, for NLU to interact with NLG, it must also be able to stop parsing at any given point and have an accessible context ready to be used by NLG. While this interchangeability is more of a general constraint for an interactive framework (see Purver et al., 2014), its specific role for modelling self-repair will be explained below.

Finally, the NLU repair mechanisms should incur no greater processing costs than fluent utterances wherever possible. This is not only a practical stipulation for their implementation, but is also consistent with psycholinguistic evidence (Brennan and Schober, 2001, Section 2.4.3).

6.1.2 NLG requirements

The desiderata for an NLG account share the five just described of NLU (but in a generative capacity for the mentioned phenomena), with the additional requirement that its interaction with a conceptualiser may include changing generation inputs at any given point during the interaction, and must deal with such changes with appropriate processing and output behaviour. In the spirit of (Guhe and Habel, 2001; Guhe, 2007), communicative goals may be incrementally constructed by a conceptualiser (or dialogue manager) and passed to the tactical generation processes, so an efficient mechanism must be in place to generate required repairs to reflect the conceptual changes in the most natural way possible. As discussed in Section 3.3.1, Skantze and Hjalmarsson (2010) began to address this, but their system, lacking an incremental semantics and the requirements listed for NLU above, lacked the facility for representing the discourse effects caused by generating self-repairs, nor could their system construct representations that could be worked on by NLU. Buß and Schlangen (2011)'s model represented repair events, though

$$R_1 : \begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_2 \\ l_3 & : & T_3(l_1) \end{bmatrix} \quad R_2 : \begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_{2'} \end{bmatrix} \quad R_3 : []$$

$$S_1 = \begin{bmatrix} l_1 & = & a \\ l_2 & = & b \\ l_3 & = & c \end{bmatrix} \quad S_2 = \begin{bmatrix} l_1 & = & a \\ l_2 & = & b' \end{bmatrix} \quad S_3 = []$$

Figure 6.1: Example TTR record types (top row) and records (bottom)

it did not interface with an incremental grammar that would make interchangeability possible, nor provide an NLU framework that could represent incremental meaning construction from an addressee (the user, in practical terms)– both are required to represent the discourse effects of self-repairs.

The purpose of the NLU and NLG models and the implementational methods presented here is primarily a *tool-for-understanding* cognitive models of dialogue (Schlangen, 2009), and also to build systems to facilitate more natural interaction with human users. Consequently the technical tools used are part of the abstract model that I believe gives the basis for a plausible incremental semantics for self-repair, but these same tools are used for implementational purposes within the algorithms described in Section 6.7 for dialogue systems. One purpose informs the other, and this chapter can be read from either perspective.

6.2 Background: DS-TTR, the IU framework and DyLan

I now re-introduce the technical tools from Section 3.3.3 in more detail, by describing Type Theory with Records (TTR), its combination with Dynamic Syntax in DS-TTR, the IU framework and the `DyLan` NLU module necessary for the account.

6.2.1 TTR

Firstly, I describe the chosen semantic representation framework for the model, Type Theory with Records (TTR, Cooper, 2005, 2012) in more detail.[2]

In TTR, the principal logical form of interest is the *record type* (abbreviated 'RT' largely

[2]I only introduce the elements of TTR relevant to the phenomena discussed below. See Cooper (2012) for a detailed formal description.

from here), consisting of sets of *fields* of the form $[\, l : T \,]$ containing a label l and a type T, representing the central type-theoretic judgement $l : T$, that an object labelled l is of type T. RTs can be witnessed (i.e. judged as inhabited) by *records* of that type, where a record is a semantic object structured isomorphically to a RT, consisting of sets of label-value pairs $[\, l = v \,]$. See Figure 6.1 for examples.

The central type judgement in TTR that a record s is of record type R, i.e. $s : R$, can be made from the component type judgements of individual fields of R, e.g. the one-field record $[\, l = v \,]$ is of record type $[\, l : T' \,]$ just in case type v is of type T'. Whether this is *true* or *false* is determined by a model for a TTR domain (a type system) that has a valuation function $A(T)$ for each type T, which maps each T to a set of objects (type inhabitants) which are disjoint from the set of types and T is a type on a defined partially ordered set of types (type hierarchy). Therefore, $l : T'$ is true iff the object labelled l in the type system is a member of the set $A(T')$; i.e. $l : T'$ iff $l \in A(T')$. If $l : T'$ is true then the judgement in a record $l = v$ is true iff all objects of type v are of type T'; i.e. given $l : T'$ is true, then $l = v$ is true iff $v : T'$ (where $v : T'$ is true iff $A(v) \subseteq A(T')$ is true).

This kind of type theory can be viewed as a set theory with set labels, which includes atomic objects which are not sets themselves but can be set members (type inhabitants), and where the sets are labelled by their given type. Given this equivalence, the syntax of the valuation function $A(T)$ where T is a complex type (i.e. meet (conjunctive) type, join (disjunctive) type or negative type) can be seen as equivalent to and consistent with set-theoretic valuations with the equivalent set-theoretic operators (i.e. set intersection, set union or complement sets) in the type system's semantics. This can be shown for two types T_1 and T_2 in (6.7) where the equivalence of a meet (conjunctive) type $T_1 \wedge T_2$ to set intersection $A(T_1) \cap A(T_2)$ and the equivalence of the join (disjunctive) type $T_1 \vee T_2$ to set union $A(T_1) \cup A(T_2)$ for a given domain of objects $\{a, b, c\}$ is shown.[3]

[3]This set-theoretic characterisation of TTR's semantics is how I understand it, and seems to be the case implicitly in Cooper (2012) as a set-theoretic valuation function $A(T)$ where $T : \mathit{Type}$ is defined as part of any TTR type system. I will not deal with the valuation function in more detail here, nor address the decidability of type judgements in these type systems, though I have found that for the purposes of dialogue systems and models, an implementation of TTR with a simple type system can give decidable judgements.

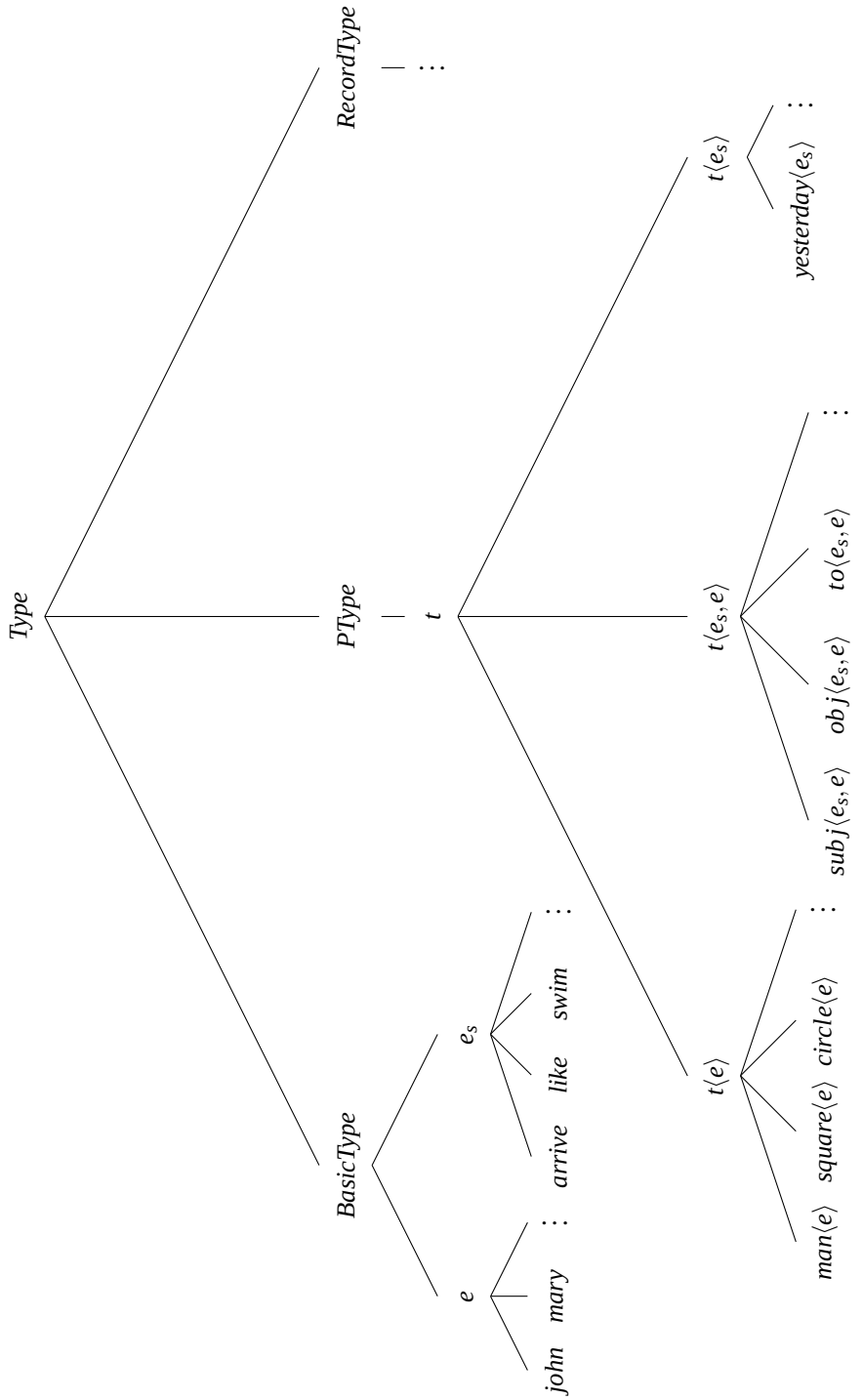


Figure 6.2: Part of the type hierarchy ordered by type inclusion. The $\langle \cdots \rangle$ show the arity (sequence of argument types) of the *PType*s.

$$A(T_1) = \{a,b\} \tag{6.7}$$
$$A(T_2) = \{b,c\}$$
$$A(T_1 \wedge T_2) = A(T_1) \cap A(T_2) = \{b\}$$
$$A(T_1 \vee T_2) = A(T_1) \cup A(T_2) = \{a,b,c\}$$

In addition to judgements that atomic objects are of given types, it is possible to make judgements that a type is of another type inductively from their position in a type order or hierarchy; a simple type hierarchy including most of the types used in this thesis is shown in Figure 6.2. This hierarchy is ordered by the subtype relation as will be explained further in the next chapter. The main types we are concerned with are *BasicType*s, that is types with sets as their extension; *PType*s, which are predicate types with arguments of *BasicType*s or other *PType*s and which are essentially functions from a defined sequence of arguments of the appropriate type (i.e. their arity) to a type; and *RecordType*s, which as exemplified above are sequences of type judgements. I also make use of list types later on.

The single-field RT check just mentioned is generalisable to records and RTs with multiple fields: a record s is of RT R if s includes fields with labels matching those in the fields of R in a one-to-one relation, such that all fields in R are matched, and all label-matched fields in s have a value of the same type (which can be a subtype of that type, see below) as their corresponding field in R– this can be defined as in (6.8). Thus it is possible for the record s to have more fields than RT R and for $s : R$ to still hold, but not vice-versa: $s : R$ cannot hold if R has more fields than the record s.

(6.8) *Record type check*:

For a record s and and record type R, $s : R$ is true iff for every field $\left[\ l\ :\ T\ \right]$ in R there is a field $[\ l\ =\ v\ ]$ in s such that $v : T$.

Fields can have values representing predicate type (*PType*) judgements, such as $l_3 : T_3(l_1)$ in Figure 6.1, and consequently fields can be *dependent* on fields preceding them (i.e. represented graphically higher up) in the RT, e.g. in Figure 6.1, l_1 is bound in the *PType* judgement field l_3, so l_3 depends on l_1.

Subtypes and manifest fields

Given the assumption that the semantic interpretation requirement of incremental NLU and NLG models is to extract or present the maximal information from an utterance as it is processed, a strongly incremental account will require checking whether RTs under construction are consistent with the RTs representing domain concept RTs provided by a conceptualiser (or dialogue manager) incrementally. To do this I make use of the $\sqsubseteq$ ('is a subtype of') relation, which is *subsumptive* in TTR, that is if RT R_1 is a subtype of RT R_2 (i.e. $R_1 \sqsubseteq R_2$) then there are no objects of type R_1 that are not of type R_2, or in the sense of the phrase from Description Logic, R_1 *is subsumed by* R_2.

Operationally, subtype relation checking can be defined for RTs in terms of fields as simply: $R_1 \sqsubseteq R_2$ iff for all fields $[\, l : T_2 \,]$ in R_2, R_1 contains $[\, l : T_1 \,]$ where $T_1 \sqsubseteq T_2$. In Figure 6.1, it will be the case that $R_1 \sqsubseteq R_3$, $R_2 \sqsubseteq R_3$ and $R_1 \sqsubseteq R_2$ iff $T_2 \sqsubseteq T_{2'}$. The transitive nature of this relation (i.e. iff $R_1 \sqsubseteq R_2$ and $R_2 \sqsubseteq R_3$ then $R_1 \sqsubseteq R_3$) can be used effectively for type-theoretic inference as will be described in the next chapter. An operational definition for a subtype check, adapted from (Fernández, 2006, p.96), is given in (6.9). If R_1 has n fields and R_2 has m fields, assuming naively a uniform cost for each type check on the type hierarchy, the complexity of this check can be $O(n \times m)$ in the worst case where every field in one RT needs to be compared against every field in the other.[4] Note that the label-matching conventions for type checking are extremely useful for computability here, as the complexity would be far greater if unconstrained re-labelling was permitted.

(6.9) *Subtype relation check*:

For record types R_1 and R_2, $R_1 \sqsubseteq R_2$ holds just in case for each field $\left[\; l \;:\; T_2 \;\right]$ in R_2 there is a field $\left[\; l \;:\; T_1 \;\right]$ in R_1 such that $T_1 \sqsubseteq T_2$. This relation is reflexive and transitive.

While I do not discuss the full stratified type hierarchy of types for TTR here, I note that for all types, $T_1 \sqsubseteq T_2$ implies that $T_1 : T_2$, but does not imply $T_2 : T_1$ unless $T_2 \sqsubseteq T_1$, a consistency that extends to RT judgements – see Cooper (2012). There are many complexities here which this thesis will not deal with, for instance stratification of different orders of types to avoid Russell's paradox– again see Cooper (2012) for details. I do not believe these complexities affect TTR's suitability for dialogue modelling and the discussion here.

[4] The cost of the subtype check for a field may be more costly if it is dependent (i.e. a *PType*, however this is not important for the discussion here.

I use the notion of *manifest* (singleton) types, e.g. T_a, the type T of which only a is a witness. Here, I represent manifest RT fields such as $[\, l : T_a \,]$ where $T_a \sqsubseteq T$ by using the syntactic sugar $[\, l_{=a} : T \,]$ following Cooper (2012). The subtype relation effectively allows progressive instantiation of fields in a monotonic fashion, as the addition of fields to an RT R, and the manifestation of fields in R, leads to R' where $R' \sqsubseteq R$. This is practically useful for an incremental dialogue system in terms of meeting the strong incremental interpretation and minimization of re-computation requirements (see Section 3.5.3) and for other reasons of self-repair processing as I will explain.

Meet types and merge operations

I also make use of the *meet type* of two or more RTs and an operation to yield an equivalent RT to their meet type. As Cooper (2012) explains, the meet of two RTs results in a type that is no longer an RT, even if the objects it witnesses are records, however an *equivalent* RT to the meet type of two RTs R_1 and R_2 is the yield of a merge operation ($R_1 \barwedge R_2$) (Larsson, 2010). Two types T_1 and T_2 are equivalent iff for any object a in the domain such that iff $a : T_1$ then $a : T_2$ and vice-versa. In the simplest case merge can be characterized as union of fields of two RTs, for example for R_1 and R_2 in (6.10).[5]

$$\begin{gathered} \text{if } R_1 = \begin{bmatrix} l_1 : T_1 \\ l_2 : T_2 \end{bmatrix} \text{ and } R_2 = \begin{bmatrix} l_2 : T_2 \\ l_3 : T_3 \end{bmatrix} \\ R_1 \wedge R_2 \equiv R_1 \barwedge R_2 = \begin{bmatrix} l_1 : T_1 \\ l_2 : T_2 \\ l_3 : T_3 \end{bmatrix} \end{gathered} \tag{6.10}$$

I also introduce *asymmetric merge* ($\boxed{\barwedge}$) as described by Dobnik et al. (2013). Operationally this differs from the standard merge in that given two record types R_1 and R_2, $R_1 \boxed{\barwedge} R_2$ will yield a RT which is the union of all fields with labels not shared by R_1 and R_2 and the asymmetric merge of the remaining fields with the same labels, whereby R_2's type values take priority over R_1's fields, yielding a resulting RT with R_2's fields only in those cases. A simple example is given in (6.11).

[5] This chapter is only concerned with simple examples that avoid label-type clashes between two RTs (i.e. cases where R_1 contains $l_1 : T_1$ and R_2 contains $l_1 : T_2$); in these cases the operations are more complex than union of fields.

$$\text{if } R_1 = \begin{bmatrix} l_1 : T_1 \\ l_2 : T_2 \end{bmatrix} \text{ and } R_2 = \begin{bmatrix} l_2 : T_{2'} \\ l_3 : T_3 \end{bmatrix}$$
$$R_1 \boxed{\wedge} R_2 = \begin{bmatrix} l_1 : T_1 \\ l_2 : T_{2'} \\ l_3 : T_3 \end{bmatrix} \tag{6.11}$$

RT functions and paths

One of the advantages of TTR over purely feature matrix based systems is that it incorporates elements of the λ-calculus, which enables it to use insights from use of the simply-typed λ-calculus work in formal semantics since Montague (1974). In particular, TTR allows functions which map from a domain of a given type to a range of a given type. Here we are interested in functions of type $RecordType \rightarrow RecordType$, which can be represented as a lambda function with a domain and range RT:

$$\lambda r : RecordType . r_1 : RecordType$$

In cases where the range RT may need to reference fields in the domain RT, this can be done through the range RT referencing the *path* to the relevant field values in the domain RT. For example, if a function carries the value of a l_1 labelled field in its domain RT to its range RT, whilst changing the types of the other fields, this will be represented as in the $\begin{bmatrix} l_1 : r.l_1 \end{bmatrix}$ field in (6.12).

$$\lambda r : \begin{bmatrix} l_1 : T_1 \\ l_2 : T_2 \\ l_3 : T_3 \end{bmatrix} . \begin{bmatrix} l_1 : r.l_1 \\ l_2 : T_{2'} \\ l_3 : T_{3'} \end{bmatrix} \tag{6.12}$$

To avoid re-duplicating identical fields in the range and domain, this can also be represented through using the asymmetric merge of the domain RT to the RT in the range of the function, so it is possible to represent the same function in different ways, with asymmetric merge providing a more concise representation, as in (6.13).

$$\lambda r : \begin{bmatrix} l_1 : T_1 \\ l_2 : T_2 \\ l_3 : T_3 \end{bmatrix} . \begin{bmatrix} l_1 : r.l_1 \\ l_2 : r.1_2 \\ l_3 : T_{3'} \end{bmatrix} = \lambda r : \begin{bmatrix} l_1 : T_1 \\ l_2 : T_2 \\ l_3 : T_3 \end{bmatrix} . r \boxed{\wedge} \begin{bmatrix} l_3 : T_{3'} \end{bmatrix} \tag{6.13}$$

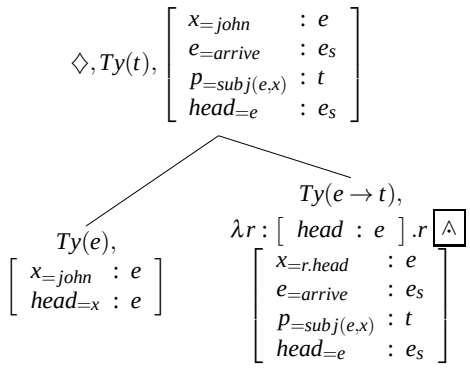


Figure 6.3: Final DS-TTR tree for "John arrives"

This equivalence is also ontologically attractive if one thinks of the RT in the domain as being a type of dialogue situation or context. For cases where the function does not depend on certain fields in a context, these need not be explicitly present in the domain RT, and will remain unchanged in the resulting RT, and the function will only override labels which clash with those in the range RT. Ginzburg (2012)'s KoS framework makes extensive use of this within dialogue state update rules. It is worth noting here that all the operations described can also apply to records in addition to RTs.

Given the highly flexible nature of TTR and its range of operations, with a suitable type hierarchy it is possible to express the fine-grained type judgements which are required for syntax, lexical semantics and dialogue update functions. The next chapter will introduce ways in which records representing dialogue situations can be used in an intuitive way to build a semantic model, while in this chapter I will focus on methods to yield appropriate TTR representations incrementally within a dialogue framework.

6.2.2 DS-TTR

Through TTR records and RTs we have access to fine-grained representations which can be constructed in a monotonic fashion, however in order for these to become available on an incremental basis, a TTR construction mechanism triggered by linguistic input as it is consumed word-by-word is required. For this purpose TTR is combined with Dynamic Syntax (DS, Kempson et al., 2001; Cann et al., 2005, *inter alia*) in the formalism DS-TTR (Purver et al., 2011; Eshghi et al., 2012, 2013) which integrates TTR representations with the inherently incremental DS.

As described in Section 3.3.3, DS(-TTR) is an action-driven formalism. The trees such as Figure 6.3 are constructed monotonically through sequences of tree-building actions consistent with Logic of Finite Trees (LOFT, Blackburn and Meyer-Viol, 1994). The DS lexicon comprises *lexical actions* keyed to words, and also a set of globally applicable *computational actions* (equivalent to general syntactic rules), both of which constitute packages of monotonic update operations on semantic trees, and take the form of IF-THEN-ELSE action-like structures. DS-TTR does not change the LOFT backbone of the DS tree building process, nor does it currently augment the computational actions directly– the computational actions used in the current implementation of DS-TTR used here can be found in Appendix B. However, RT formulae are introduced into the lexical actions; for example the lexical action for the word "John" has the preconditions and update operations in (6.14):

(6.14)

IF	$?Ty(e)$
THEN	`put`$(Ty(e))$
	`put`$(\left[\ x = john : e\ \right])$
ELSE	`abort`

As can be seen in Figure 6.3, the DS node types (rather than the RT formulae at the nodes) are terms in the typed lambda calculus, with mother-daughter node relations corresponding to semantic predicate-argument structure. The pointer object, $\diamondsuit$, indicates the node currently under development. Parsing begins by an initial prediction step on an axiom of a single node with requirement $?Ty(t)$ and then the set of computational actions are Kleene star iterated over to yield a tree set. The first word is parsed when it is consumed by triggering all possible parses in the current tree set, and then the set of computational actions are then again iterated over to yield a new tree set.

DS parsing yields an incrementally specified, partial semantic tree as words are parsed or generated, and following Purver et al. (2011) in this thesis DS-TTR tree nodes are decorated not with simple atomic formulae but with RTs, and corresponding lambda abstracts representing RT λ-functions of type $RT \rightarrow RT$. See Figure 6.3. On functor nodes semantic content decorations are of the form $\lambda r{:}[\ l_1 : T_1\ ].r\ \boxed{\mathbb{A}}\ [\ l_{2=r.l_1} : T_1\ ]$ where $r.l_1$ is a path expression referring to the label l_1 in r – see the functor node with DS type label $Ty(e \rightarrow t)$ of Figure 6.3.

Using TTR's affordance of manifestness of fields as mentioned above, we have a natural representation for underspecification of leaf node content of DS trees, e.g. $[\ x : e\ ]$ is unmanifest whereas $[\ x_{=john} : e\ ]$[6] is manifest and the latter is a subtype of the former.

[6]Note again this is syntactic sugar for $[\ x : e_{john}\ ]$ and the $=$ sign is not the same semantically as that

Following Eshghi et al. (2013), DS-TTR tree nodes include a field *head* in all RTs. Technically, the range of the λ-functions at functor nodes is the asymmetric merge of their domain RT(s) with the RT in their range. This allows the *head* field of argument node RTs in β-reduction operations to be replaced by the *head* field of the function's range RT at the sister functor node in their resulting mother node RT or RT function.

The modifications in DS-TTR also include incorporating the asymmetric merge function into the LINK EVALUATION computational action (see Appendix B) . This action applies between RTs at the root of DS *linked* trees and the RTs at nodes they link from, such as in Figure 6.4's trees for the utterance "John, who smokes, arrives". This way the lower linked tree's semantic content can become incorporated in the spirit of the original presentation in Kempson et al. (2001), but with an elegant variable binding technique. In Figure 6.4, the initial LINK ADJUNCTION computational action copies the RT on the higher matrix tree's $Ty(e)$ node with its RT copied onto the lower linked tree's $Ty(e)$ node. When the LINK EVALUATION action fires after "John, who smokes" has been parsed, the RT compiled at the root $Ty(t)$ node of the lower tree is carried onto the matrix tree in its totality modulo its *head* field, via an asymmetric merge with the $Ty(e)$ node it links from. This allows normal β-reduction functionality (the ELIMINATION action) when "arrives" is parsed and the asymmetric merge preserves the information built up on the linked $Ty(e)$ node. It is worth mentioning these modifications were found to have efficiency benefits for inducing lexical actions from target RTs in the automatic learning of a DS-TTR grammar (Eshghi et al., 2013).

One final deviation from original DS representation, and one that was made in Cann (2011) in some detail for standard (non-TTR) DS, is the assumption of a neo-Davidsonian representation of verbs and adjunctive predicates, where fields correspond to an event term (notated as type e_s) and to each semantic role projected by the event. This allows all available semantic information to be specified incrementally in a strict subtyping relation e.g. providing the $\left[\ p_{=subj(e_s,x)} : t\ \right]$ field when subject but not object has been parsed, or even predictively after initial computational actions create the subject node – see Figure 6.3. While event terms are important, the need for a separate event node on the tree as originally proposed by Cann (2011) was found lacking for the purposes here. However, incremental specification of event information, such as the tense and aspect information contributed by auxiliary verbs in English (Cann, 2011) may still be achieved through using linked trees from functor nodes or the root node, which copy the event

in a record.

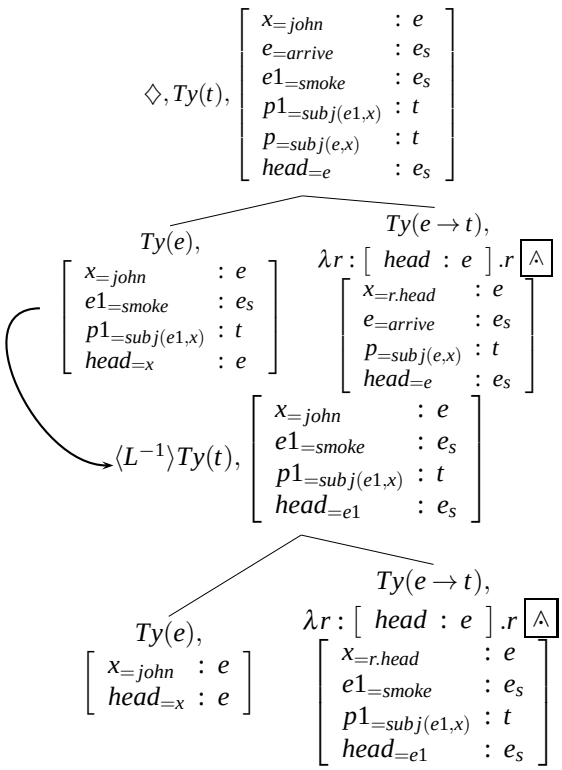


Figure 6.4: Final DS-TTR tree with a linked tree for "John, who smokes, arrives"

term and compile with *PType* fields at their root node that bind the copied term, using standard LINK ADJUNCTION. Temporal adverbs such as 'yesterday' and other adjunctive forms may also predicate on the event term, and are similarly treated as linked trees. This approach reduces the size of the matrix tree as sub-trees encoding the structure of epsilon terms to represent events as in (Cann, 2011) are dispensed with, however progressive construction of the term through allowing incremental predicate binding does not make the framework any less dynamic, nor reduce its coverage.

6.2.3 The IU Framework and DyLan

The Incremental Unit (IU) framework (Schlangen and Skantze, 2009, 2011) is an abstract framework for incremental dialogue systems which can be used for the formal specification of a dialogue model, or as the basis for software design. I now introduce the key elements used in this chapter.

The IU framework can be described as a network of modules, each comprising a *left buffer*

of a graph of input *incremental units* (IUs), a *processor* and a *right buffer* consisting of a graph of output IUs. IUs have a *payload* which determines what type of data they carry, whether it is a word, POS tag or numerical value, or anything else determined by the system designer.

It is the edits to IU graphs, which comprise *add*, *revoke* and *commit* action of IUs in the IU graph of a module's right buffer and the effect of doing so on its downstream modules' left buffers that determines system behaviour. These functions being applied to a given IU IU_n will be notated *add*(IU_n), *revoke*(IU_n) and *commit*(IU_n) while the boolean valuation of IU_n being in these states will be notated by *added*(IU_n), *revoked*(IU_n) and *committed*(IU_n).

Furthermore, IUs can have *same level link* relations between one another if it is desirable that two or more IUs be in some dependency relation to each other within a module buffer, or have *grounded in* relations between IUs in different module buffers. The creation of these links will be notated *SameLevelLink*(IU_2, IU_1) where the direction of the link is from IU_2 to IU_1 and similarly *GroundedInLink*(IU_2, IU_1) makes IU_2 grounded in IU_1. The boolean valuations of IUs being in these relations to each other will be notated *SameLevelLinked*(IU_2, IU_1) and *GroundedIn*(IU_2, IU_1). Same level links standardly go back in processing time, for example word 2 will be same level linked back to word 1, i.e. *SameLevelLinked*$(word_2, word_1)$, and in these cases I will call the path-final IU in the graph the *right-most IU* which has no *SameLevelLink* linking to it. Otherwise *SameLevelLink*s can be indicative of structure or hierarchy, depending on the module concerned: for example in parsing if a tree is constructed by sub-trees, the same level link will go from the larger tree to the subsumed sub-trees that composed it. Grounded in relations on the other hand standardly go from output to the input they were triggered by, for example a process downstream of ASR such as POS-tagging will be grounded in its triggering word graph input IUs, or dialogue act tags may have grounded in links to the parse trees used for their tagging decision. Motivated by fine-grained incrementality and the dependencies just mentioned, buffers are defined as graphs with nodes that represent IUs, allowing for multiple hypotheses to be constructed with time-linear input and their subsequent revision. In this chapter I will use the incremental graph construction process of the IU framework, only slightly deviating from the original proposal in using edges instead of nodes for IUs, to model strongly incremental NLU and NLG, and then show how this can be used in implemented algorithms in Section 6.7.

The `DyLan` NLU module (Purver et al., 2011) combines the grammar and parsing process

of DS-TTR and RT checking in a module of a Java-based implementation of the IU framework, Jindigo (Skantze and Hjalmarsson, 2010). It makes use of Sato (2011)'s characterization of DS parsing as a DAG, and the graphical characterization allows an overarching super-graph which ranges over sequences of the more fine-grained DS parse DAG and which characterizes the module's IU graph. This can be seen clearly by repeating the diagram Figure 3.9 from Chapter 3 in Figure 6.5.

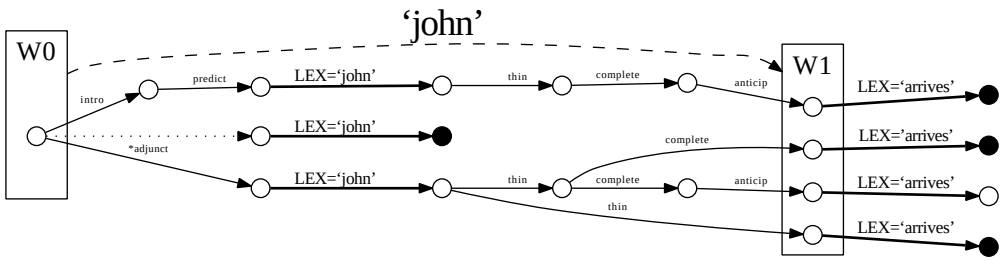


Figure 6.5: DS context as a parse DAG (circular nodes=trees, solid edges=lexical(bold) and computational actions) with an over-arching word DAG (rectangular nodes=tree sets, dotted edges=word hypotheses) with word hypothesis 'john' spanning tree sets W0 and W1.

Despite this strong incremental representation available for each word, before the modifications described below were made, DyLan was capable of compiling RT formulae for complete trees only, which mirrored the lack of strong incrementality in formal DS-TTR at the time. The steps for improving this below are therefore for formal modelling as well as for implementational benefit.

6.3 Extensions to DS-TTR and the IU framework for strong incrementality

DyLan as originally presented (Purver et al., 2011) begins to meet the self-repair and incrementality desiderata described in Section 6.1, but it requires some development. DS's augmentation to DS-TTR, when implemented in an NLU module in the IU framework permits incremental interaction with frame-based dialogue management. However DS-TTR does not meet all of the above requirements for NLU, particularly of strong incremental interpretation on a word-by-word basis and incremental representation of repair events. Furthermore, no generation model was specified originally, nor the potential interaction between NLU, NLG and dialogue management. Below I will describe the steps taken to address these issues.

6.3.1 TTR operations for RT similarity and difference

The first extension to TTR I make is that of similarity and difference functions for two RTs. While the core of TTR's motivation is reasoning with the structural similarity between semantic objects (Cooper, 2012), it lacks functions for directly obtaining a measure of this or the appropriate RT representations. For more specific motivation, as was shown in Chapter 5, computing differences in incremental representations is important for self-repair detection and classification. To address this, I introduce a new operation that calculates the *difference* between two RTs. While difference is not standardly included in type theory (as it pre-supposes a notion of complementation), its inclusion into the version of TTR presented here is not harmful to the rest of the system. Intuitively $R_1 - R_2$ should yield a type representing the maximal amount of information in R_1 that is not in R_2. The difference may be additive, subtractive or both. To account for this I describe the difference as a meet (conjunctive) type of two record types: one comprising the type judgements in R_1 but not in R_2 and the other a *negative* RT representing the type judgements present in R_2 but not in R_1 – see (6.15). Either of these conjuncts may not feature in the difference result depending on the RT arguments the function applies to: e.g. the subtractive type is not present in example (6.16). I define the difference operationally in terms of field comparison as in Algorithm 2. Note that if for any field in a RT R_1 of the form $\left[\ l\ :\ T\ \right]$ there is a corresponding field $\left[\ l\ :\ T'\ \right]$ in RT R_2 and it is the case that $T \sqsubseteq T'$, then in the result of $R_1 - R_2$, the difference in the subtractive conjunct should be $\left[\ l\ :\ T \wedge \neg T'\ \right]$, however this algorithm is a simple version of RT difference which assumes it will never encounter such inter-field subtype differences – see (6.16).

$$\text{if } R_1 = \begin{bmatrix} l_1 : T_1 \\ l_2 : T_2 \\ l_3 : T_3 \end{bmatrix} \text{ and } R_2 = \begin{bmatrix} l_2 : T_2 \\ l_3 : T_4 \end{bmatrix} \text{ and not } T_3 \sqsubseteq T_4 \text{ then}$$
$$R_1 - R_2 = \begin{bmatrix} l_1 : T_1 \\ l_3 : T_3 \end{bmatrix} \wedge \neg \left[\ l_3 : T_4\ \right] \tag{6.15}$$

$$\text{if } R_1 = \begin{bmatrix} l_1 : T_1 \\ l_2 : T_2 \\ l_3 : T_3 \end{bmatrix} \text{ and } R_2 = \begin{bmatrix} l_2 : T_2 \\ l_3 : T_4 \end{bmatrix} \text{ and } T_3 \sqsubseteq T_4 \text{ then}$$
$$R_1 - R_2 = \begin{bmatrix} l_1 : T_1 \\ l_3 : T_3 \end{bmatrix} \tag{6.16}$$

In terms of Sundaresh and Hudak (1991)'s work on projection algebras, we could also say that if R_1 and R_2 are elements of a commutative distributed domain, the difference between R_1 and R_2 is the least element R_3 such that $R_1 \sqsubseteq R_2 \vee R_3$. However, this terminology presupposes

Algorithm 2 Record Type difference operation $R_1 - R_2$

INPUT: R_1 and R_2
Let $add = [] : RecordType$ and $subtract = [] : RecordType$ ▷ Initialise difference conjuncts.
Let $matched = \{\}$ ▷ A list of field names that have been matched.
for $\left[\ l_i\ :\ T_1\ \right] \in R_1$ **do** ▷ Iterate over all fields.
  for $\left[\ l_j\ :\ T_2\ \right] \in R_2$ **do** ▷ Check each field for differences.
    if $l_i = l_j$ **then**
      $matched.append(l_i)$ ▷ Matching field labels.
      if not $T_1 : T_2$ **then** ▷ T_2 does not subsume T_1.
        $add = add \wedge \left[\ l_i\ :\ T_1\ \right]$ ▷ Add to the appropriate difference conjuncts.
        $subtract = subtract \wedge \left[\ l_i\ :\ T_2\ \right]$
  if not $l_i \in matched$ **then** ▷ No match for this label, must be addition.
    $add = add \wedge \left[\ l_i\ :\ T_1\ \right]$
for $\left[\ l_j\ :\ T_2\ \right] \in R_2$ **do** ▷ Another pass to add unmatched fields in R_2 to subtract conjunct.
  if not $l_j \in matched$ **then**
    $subtract = subtract \wedge \left[\ l_j\ :\ T_2\ \right]$
OUTPUT: $add \wedge \neg subtract$ ▷ Return difference conjunction.

a well defined ordering on our domain of RTs and requires a notion of join types (disjunctions) on it: as I currently do not characterize the subtypes of *RecordType* as an ordered set for now, I leave the difference operator in the operational form in Algorithm 2. I will discuss order-theoretic characterizations of RTs in modelling a dialogue domain in the next chapter.

It is also possible to quantify RT difference numerically as well as yield an appropriate representation. For this I use an asymmetric similarity metric between RTs I developed for DS-TTR grammar induction evaluation in (Eshghi et al., 2013). This is done by calculating precision, recall and F-score of one RT to another using a method similar to the semantic graph evaluation function proposed by Allen et al. (2008).

This similarity function from RT R_1 and RT R_2 (i.e. the degree of R_1's similarity to R_2) is computed in the following way: Each field will get a potential score in the range [0,1]. A method $maxMapping(R_1, R_2)$ constructs a mapping from fields in R_1 to those in R_2 to maximise alignment, with fields that map completely scoring a full 1, and partially mapped fields receiving less, depending on the proportion of the R_1 field's representation that subsumes its mapped R_2 field; e.g. a unary predicate field in R_2 such as $\left[\ p_{=there(e)}\ :\ t\ \right]$ could score a maximum of 3: 1 for correct type t, 1 for correct predicate *there* and 1 for the subsumption of its argument e; I use the total to normalise the final score. The potential maximum $maxMapping(R_1, R_2)$ is therefore the number of fields in R_2 (including those in embedded record types). It is then possible to

use the standard F-score (F1) measure to balance the effect of precision and recall for mapping fields correctly, and use this measure for $Similarity(R_1, R_2)$ (the similarity of R_1 to R_2). So, for some RT R and goal RT G, with N_R and N_G fields respectively, $Similarity(R, G)$ is computed as in (6.17).

(6.17) $precision(R, G) = maxMapping(R, G)/N_R$

$recall(R, G) = maxMapping(R, G)/N_G$

$Similarity(R, G) = 2 \times \frac{precision(R,G) \times recall(R,G)}{precision(R,G) + recall(R,G)}$

With these ways of computing similarity and difference at hand, it is possible to allow fine-grained type-theoretic inference for dialogue processing and allows gradient values of similarity which, as discussed in the previous chapters, are important for self-repair interpretation. Before these can be used however, we need a way of strengthening DS-TTR's incrementality, that is to yield RT representations from utterances as they are produced word-by-word. This will address the requirements of modelling the semantics of self-repair.

6.3.2 Strongly incremental construction of record types

I describe the principal modification made to the DS-TTR parsing process here. This is that TTR record types are compiled incrementally on a word-by-word basis after each input word (or a candidate word in generation) is parsed, giving RT representations for *partial trees* in addition to complete ones, as was the state of DS-TTR in Purver et al. (2011). The incremental compilation allows the representations to become available immediately to other modules through accessing the RT compiled at root DS-TTR tree nodes. This modification gives DS-TTR an interface to representations from other sources of contextual information, most naturally dialogue semantics in the style of KoS (Ginzburg, 2012), on a word-by-word level. As I will show, adopting the DS principle that generation is driven by parsing, these alterations *a fortiori* extend to generation, as will be described in Section 6.3.4.

Previously in DS-TTR (Purver et al., 2011), only DS type-complete tree nodes had formulae and consequently only complete trees with no requirements yielded RTs at their root. I introduce underspecified RTs into type incomplete nodes that lack semantic formulae, allowing the relation between these and instantiated RT values to be computed, giving the maximal amount of semantic information available and satisfying the principle of strong incremental interpretation.

The DS-TTR compilation is simply achieved by allowing functional application to apply word-by-word incrementally on partial formulae rather than only permitting it for DS type-complete sub-trees, allowing a RT at the root node to be compiled for any partial tree, which is incrementally further specified as parsing proceeds, preserving DS's monotonicity. It is worth noting that while underspecified RTs are introduced, the DS requirement types are left as requirements, rather being made complete before they have been satisfied by an appropriate lexical or computational action. DS-TTR is still therefore driven by DS parsing dynamics and LOFT. Within a given parse path, each maximal RT of the tree's root node is a subtype of the parser's previous maximal output, making the formula construction monotonic.

More concretely, compilation of a RT for any partial tree is achieved by a simple two-stage algorithm:

1. Decorate all terminal nodes lacking formulae with RTs containing a field of the appropriate maximally general type judgement for the DS node type. Argument nodes are given simple RTs and functor nodes given RT functions with underspecified *PType* fields in their range RT with the appropriate corresponding argument fields. Examples of the mapping from DS node types to TTR formulae are as below:

DS node type	Underspecified TTR formulae
$?Ty(e)$	$\begin{bmatrix} x & : e \\ head_{=x} & : e \end{bmatrix}$
$?Ty(e \rightarrow t)$	$\lambda r : \begin{bmatrix} head & : e \end{bmatrix}.r \boxed{\mathbb{A}} \begin{bmatrix} x_{=r.head} & : e \\ e & : e_s \\ p_{=subj(e,x)} & : t \\ head_{=e} & : e_s \end{bmatrix}$
$?Ty(e \rightarrow (e \rightarrow t))$	$\lambda r : \begin{bmatrix} head & : e \end{bmatrix}.\lambda r1 : head \begin{bmatrix} e & : \end{bmatrix}.r1 \boxed{\mathbb{A}} r \boxed{\mathbb{A}} \begin{bmatrix} x1_{=r1.head} & : e \\ x_{=r.head} & : e \\ e & : e_s \\ p_{=obj(e,x)} & : t \\ p_{=subj(e,x1)} & : t \\ head_{=e} & : e_s \end{bmatrix}$

2. Carry out functional application from the RT functions of the functor nodes to the RTs of their sister argument nodes, compiling a β-reduced RT at their mother node (functioning like the standard DS ELIMINATION action). Continue in a bottom-up fashion until all

nodes are covered. The ordering of these applications for a given tree is achieved through an initial depth-first iterative search for functor nodes with incomplete mother nodes, halting upon compilation of a formula at the root node. This must start in the bottom left-hand corner of the tree. If present, linked trees must be compiled first, from the bottom linked tree if multiple, and results from LINK EVALUATION must be compiled up to the matrix tree. *Unfixed* nodes constructed from STAR ADJUNCTION (see Appendix B) must also be collapsed into a vacant appropriate node first before the matrix tree is compiled.

In terms of advantages for a dialogue system, this compilation process gives us a data structure representing a parse path's maximal semantic content made incrementally available. This is practically and methodologically useful as a RT can be extended with information that may not correspond directly to DS tree structure (not strictly linguistic information), for example contextual dialogue information about the speaker of each word in the utterance as described by Purver et al. (2010), giving the potential for integration of a full context model in the style of KoS (Ginzburg, 2012), and even perceptual statistical data being made available to a dialogue agent on-line (Cooper et al., 2014; Dobnik et al., 2013; Larsson, 2011). I will discuss RT inference in more detail as regards its connection to classifying dialogue situations in the next chapter.

6.3.3 Strongly incremental interpretation in `DyLan`

Now situating the DS-TTR incremental compilation process in the `DyLan` dialogue framework as a whole, it is possible to describe a parse state which combines Purver and Kempson (2004)'s formulation of DS context as explained in Section 3.3.3, a minimal notion of dialogue context in the style of KoS (Ginzburg, 2012) and also Sato (2011)'s insight that the context of DS parsing can be characterized in terms of a DAG with trees for nodes and DS actions for edges. This combination allows a satisfaction of strong incremental interpretation and incremental representation principles, and a notion of dialogue situation that is extendible beyond purely linguistic content and context, but can model dialogue or interaction context.

To do this formally, I propose a RT to represent DS-TTR parsing judgements as that in (6.18). Parsing judgement records contain four fields:

- **tree**: the path-final DS-TTR tree
- **actions**: the lexical and computational actions fired with the last word on this path

- **cont**: the semantic content, in the form of the maximal RT compiled after parsing the last word on this path

- **ctxt**: a simple notion of dialogue context which represents the currently active dialogue move(s) in the form of a list of simple dialogue move objects active after processing the actions. This is not quite the same as the LatestMove field in KoS (Ginzburg, 2012), as there could be multiple active moves at once. For these dialogue move objects I stipulate a *PType* called *Move* of which the following are subtypes, and whose arguments are always a dialogue participant identifier and semantic content in the form of a *RecordType*:
  $Assert\langle DialogueParticipant, RecordType\rangle$
  $Question\langle DialogueParticipant, RecordType\rangle$
  $Revoke\langle DialogueParticipant, RecordType\rangle$ (=Revocation of previously asserted content)
  $FwdProblem\langle DialogueParticipant, RecordType\rangle$ (=Forward-looking problem)

In the ctxt field I introduce dialogue move judgements which rely on the direct building of question fields into the DS-TTR lexicon in the style of Gargett et al. (2009); Cann (2011). In interpreting fluent utterances, simplistic dialogue act classification consists of a type judgement on compiled record type R where the current speaker is *Speaker*in ParseIU S_n of the form: IF $R \sqsubseteq \left[\ q : question\ \right]$ THEN S_n.ctxt $= [Question(Speaker, \mathrm{cont}.q)]$ ELSE S_n.ctxt $= [Assert(Speaker, \mathrm{cont})]$. The way the other two *Move* types are used will be discussed later in discussing their construction during self-repair events. As the ctxt and action fields are list types, here I define list types as behaving as lists in functional programming languages are often defined to do, with just four operations: $append(list, a)$ returns the same list $list$ with a appended the end of it, $prepend(list, a)$ returns the list $list$ with element a added at the beginning, $head(list)$ returns the first element of $list$ and $tail(list)$ returns the rest of the list from the second element onwards. If a list is given the value [] this means it is set to be empty. If lists are within record types, the TTR $\boxed{\mathbb{A}}$ operation for a given field of list type operates in the standard way, completely over-writing the value of the field in its first argument with that of the second argument.

It is simple to cast records of these interpretation judgements as IUs in an IU framework based dialogue model. I will call the type of these judgements *ParseIU* types of the RT in (6.18).

$$(6.18)\quad \mathit{ParseIU} = \begin{bmatrix} \text{tree} & : & \mathit{DSTTRTree} \\ \text{actions} & : & \mathit{list}(\mathit{DSTTRAction}) \\ \text{cont} & : & \mathit{RecordType} \\ \text{ctxt} & : & \mathit{list}(\mathit{Move}) \end{bmatrix}$$

An example record of type *ParseIU* after the speaker *User* has said "John" utterance-initially, given a lexical entry for 'John', John'^{lex} has fired after initial INTRODUCTION and PREDICTION computational actions have run will be characterized as in (6.19).

$$(6.19)\quad \begin{bmatrix} \text{tree} & = & ?Ty(t), \begin{bmatrix} x_{=john} & : & e \\ e & : & e_s \\ p_{=subj(e,x)} & : & t \\ head_{=e} & : & e_s \end{bmatrix} \\ & & \quad \langle\rangle, ?Ty(e \to t) \\ & & Ty(e), \quad \lambda r : \begin{bmatrix} head & : & e \end{bmatrix}.r \boxed{\mathbb{A}} \\ & & \begin{bmatrix} x_{=john} & : & e \\ head_{=x} & : & e \end{bmatrix} \quad \begin{bmatrix} x_{=r.head} & : & e \\ e & : & e_s \\ p_{=subj(e,x)} & : & t \\ head_{=e} & : & e_s \end{bmatrix} \\ \text{actions} & = & [\mathit{introduction}, \mathit{prediction}, \mathit{John}'^{lex}, \mathit{thinning}, \mathit{completion}, \mathit{anticipation}] \\ \text{cont} & = & \begin{bmatrix} x_{=John} & : & e \\ e & : & e_s \\ p_{=subj(e,x)} & : & t \end{bmatrix} \\ \text{ctxt} & = & [\mathit{Assert}(\mathit{User}, \text{cont})] \end{bmatrix}$$

As each ParseIU will have a *same level link* to a predecessor in the DS-TTR parse graph and also a *grounded in* link to the word input that triggered it, the strong incremental interpretation and incremental representation required for self-repair processing is available throughout the parse and generation state ParseIU graph.

Having made this IU formulation of increments of the parse state, it is possible to characterize DyLan's NLU process as three linked IU graphs, where each graph is a DAG whose edges are IUs with a *same level link* to their predecessor, and the links between the graphs are *grounded in* links which go from the output IUs to the input IUs which triggered them, as shown in Figure 6.6. The three graphs concerned are:

- *input:* a time-linear word graph posted by the ASR module, consisting of word hypothesis edge IUs W_n
- *processing:* the internal DS parsing DAG, which adds ParseIUs S_n *grounded in* their triggering word hypothesis edge IUs

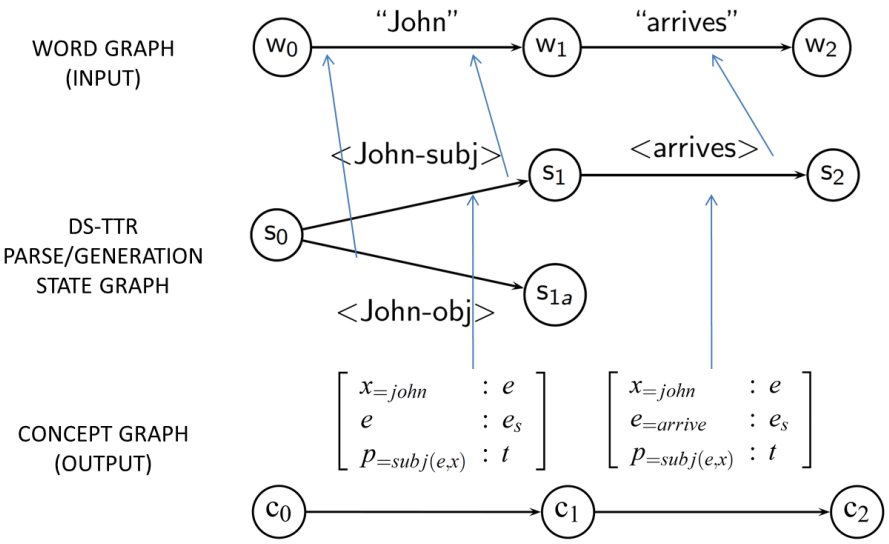


Figure 6.6: Strongly incremental interpretation in `Dylan`. *Grounded in* links go from bottom to top. *Same level links* for IUs (edges) are indicated by the predecessor relation in their IU DAG.

- *output:* a concept graph consisting of domain concept edge IUs, which are record types, C_n, *grounded in* the corresponding ParseIUs

The final state of the graphs and their inter-connectedness after parsing "John arrives" can be seen in Figure 6.6 with the value of ParseIU edge's cont field RT displayed under it. Grounded in links go from output to input IUs, which is shown in a bottom-up direction in this diagram. Same level links are not drawn as they are implicit in the ordering on the DAG, where each edge is grounded in its predecessor edge, with an (invisible) trivial empty IU being posited as the source (first) of each graph (Schlangen and Skantze, 2011). An incremental derivation can be seen in Figure 6.6. Note the grounded in links allow incremental representation (Milward, 1991) to become possible– the precise semantic contribution of each word is derivable from the graph as the DS-TTR actions are grounded in words, and furthermore the type-theoretic inference steps in the conceptualiser or dialogue manager (i.e. judgements of the current dialogue state) are grounded in the DS-TTR action sequences. Not only will this allow the operations of

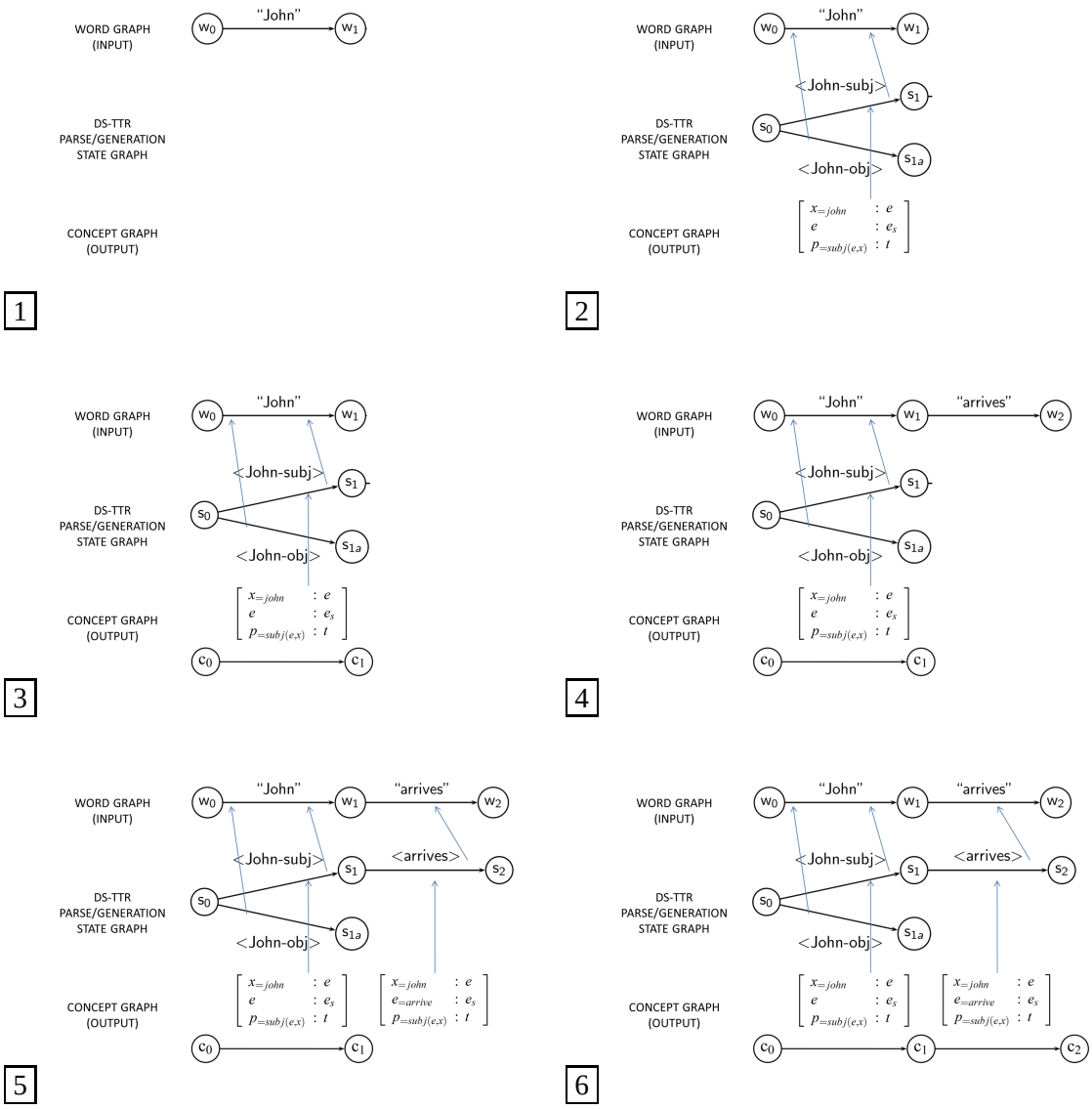


Figure 6.7: Incremental interpretation: Transitions in DyLan during parsing of 'John arrives'

adding, revoking and committing IUs in the IU framework to operate with the desired effects on all processing levels, but this will help meet some of the desiderata for both abstract and implemented models of self-repair processing set out above, as will be explained.

6.3.4 Strongly incremental semantics driven generation

The NLG model for DyLan described here builds on Purver and Kempson (2004)'s model of DS generation, with several alterations that will be explained below. The first principal modification is the use of TTR record types for *goal concepts*.

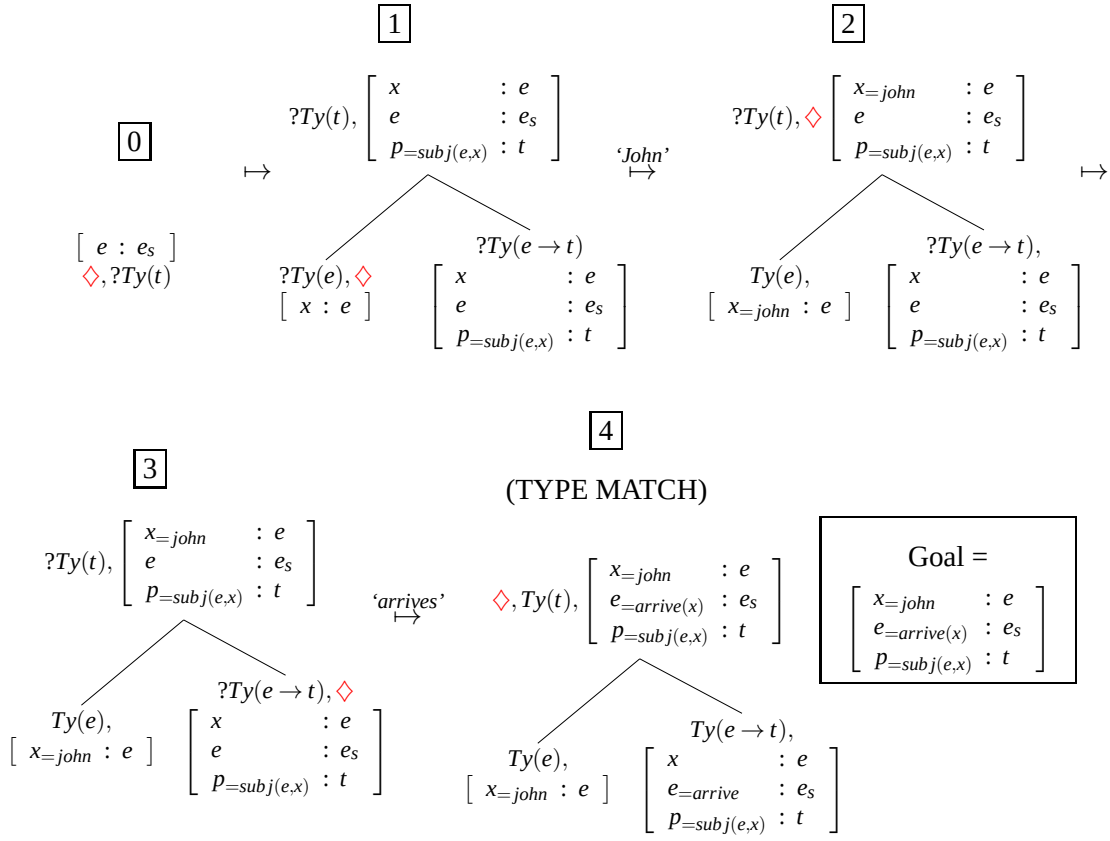


Figure 6.8: Successful generation path in DSTTR

From goal trees to goal concepts

Following the DS-TTR extension of DS parsing explained above, the modification to the original account of DS generation in Purver and Kempson (2004) is made in that in place of a goal tree, a goal concept represented by a RT is used to guide the generation process instead. Currently, fully specified RT goal concepts may look like (6.20) where the dialogue participant *Speaker*'s generation goal is to communicate an assertion that they will go to Paris.

$$(6.20)\quad \begin{bmatrix} \text{cont} & : & \begin{bmatrix} e_{=go} & : & es \\ p2_{=future(e)} & : & t \\ x1_{=Paris} & : & e \\ x_{=Speaker} & : & e \\ p1_{=to(e,x1)} & : & t \\ p_{=subj(e,x)} & : & t \end{bmatrix} \\ \text{ctxt} & : & [Assert(Speaker, \text{cont})] \end{bmatrix}$$

I replace the subsumption check for trees under construction with a *semantic filter* stage, which is characterized as a series of RT subtype relation checks (see definition (6.9) above)

between the goal concept and the RT judgement of the cont and ctxt fields of ParseIUs in the parse state graph– for the subsumption check to be passed by a candidate RT C against the goal concept GC, it must be the case that $GC \sqsubseteq C$.

As discussed for NLU, in generating *Assert* and *Question* moves, the ctxt field's list members for a ParseIU are calculated by a subtype relation check on its cont field RT built up by parsing. In generation the subsumption check extends to the ctxt field to prevent the generation of an assertion form where a question is desired by the conceptualiser and vice-versa.

An example of a successful generation path is shown in Figure 6.8 where the incremental generation of "John arrives" succeeds as the appropriate lexical actions for 'John' and 'arrives', interspersed with sequences of applicable computational actions (e.g.transitions $\boxed{0} \mapsto \boxed{1}$ and $\boxed{2} \mapsto \boxed{3}$), can be applied sequentially, passing the stage-by-stage subtype relation check, until arriving at a tree that *type matches* in $\boxed{4}$.[7]

Pre-verbal lexicalisation through type checking

Another efficiency advantage made possible by using a goal concept RT is that subtype checking can reduce the computational complexity of word-by-word lexicalisation used in the original DS generation model (Purver and Kempson, 2004). This is done through pre-verbal lexical action selection, removing the need to iterate through the entire lexicon after each word is generated. For this purpose I introduce the idea of generating a set of lexical actions *SubLex* (a sub-lexicon) which can be populated when a goal concept *GC* is input to the generator. This is done by a simple iterative selection process on the lexicon using subtype checking as in (6.21).

(6.21) *SubLexicalise(GC)* returns *SubLex*

> For each lexical action L_i in the lexicon, add to *SubLex* if *GC* is a subtype of the TTR record type (or range of the TTR record type function) added by L_i.

Depending on a system designer's choice of how many fields a DS-TTR lexical action's TTR formulae has, the size of *SubLex* will vary. For instance if lexical actions for verbs lack a field for tense information, several candidates may be selected in *SubLex* which are all valid supertypes of the goal concept (e.g. "likes", "like", "liked"), and less appropriate candidates may be filtered out

[7] Technically, the functor node TTR formulae in Figure 6.8 should be functions of type $\lambda r : RecordType.r \boxed{\wedge} r1 : RecordType$, as can be seen on the $Ty(e \rightarrow t)$ nodes in Figures 6.3 and 6.4, however to avoid clutter here I only represent their range record types. Also the ctxt part of the goal is omitted to avoid clutter: here this is an *Assertion*.

at a later stage. In this sense, the more lexicalised the grammar, the smaller *SubLex* will be, and consequently the smaller the search space for the following generation. It is also worth noting that semantically underspecified lexical entries, such as those for 'do' auxiliaries used in verb phrase ellipsis, may be selected here by default, as the values in their fields inherit values from inter- and intra- utterance context (Kempson et al., 2011), which is beneficial as where possible, anaphoric and elliptical forms can be used. The CONTEXT SUBSTITUTION and REGENERATION rules (Kempson et al., 2011) make the search back along the DS parsing DAG for values in ellipsis possible.

In DyLan, the generation module shares the same ParseIU context DAG as the interpreter, making it an interleaved dialogue system which meets requirements of inter-changeability. In the generation module, the architecture is the inverse of interpretation given the input of RT goal concepts, so we are concerned with three linked graphs:

- *input:* the goal concept graph has goal concept IU edges (RTs) between vertices GC_n posted by the dialogue manager or conceptualiser
- *processing:* the DS parsing graph of ParseIUs (shared with the interpreter module's graph) is incrementally constructed word-by-word by parsing the lexical actions in the sublexicon and subtype checking the result against the current goal concept, *add*ing the best (closest matching) ParseIU S_n
- *output:* the word graph's edges W_n are *add*ed to the output buffer incrementally during word-by-word generation, *grounded in* their corresponding DS parsing graph ParseIU, which form part of a valid generation path subsuming the goal concept (as in Figure 6.8)

6.3.5 Adding repair links and repaired status to IU networks

The final extension I make to DyLan is an augmentation of the set of graphical link relations by positing *repair links*, and also the notion of an IU being in a *repaired* state as opposed to a *committed* or *revoked* state. Repair links have a different semantics to same level links or grounded in links in that they are directed links from the repair phase to the reparandum phase of a repair. If an IU is repaired (i.e. is the receiver of a repair link), then it is given *repaired* status; this function will be notated as *RepairLink*(IU_2,IU_1) where the repair link direction is from IU_2 to IU_1. Similarly to the other links, the boolean evaluation that IU_2 repair links to IU_1 will be notated *RepairLinked*(IU_2,IU_1). *Repaired* status can be also be given by assignment without repair linking; this function will be notated *Repair*(IU_n). Repaired IUs cannot be a direct trigger

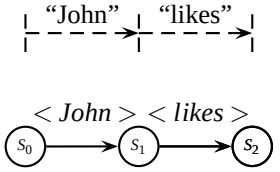
for output, but they are not revoked. This gives the requisite downgrading without removal from context (Ginzburg, 2012). Furthermore, the effect of repaired IUs within their own graph is crucial for constructing the meaning of self-repairs as I will now explain.

6.4 Interpreting self-repairs incrementally

With the extensions just described, incremental self-repair modelling falls out quite naturally from DyLan when viewed as an abstract framework, and also in terms of implementation.

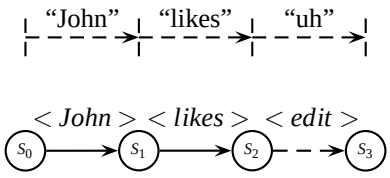
In repair processing, edit terms should be detected immediately, and a reparandum should be detected as soon as the repair onset is detected, as was the case in the STIR system in the last chapter. Furthermore the appropriate representation for repairs should be made available in as unified a way as possible to normal fluent utterance processing just described, here in terms of the DS-TTR parsing and generation and the effects on the context of a ParseIU graph described above. I will describe how this is achieved below, and these methods can be seen schematically in Figure 6.9 which diagrams the incremental construction of the ParseIU graph as triggered by the input utterance "John likes, uh, loves Mary" spoken by the speaker *User*. The word input graph is the top graph on the left-hand side, and below that is its corresponding ParseIU graph being constructed as the words are consumed and the appropriate record representing the right-most ParseIU's cont and ctxt fields is shown on the right.

Figure 6.9 shows a substitution repair and the desired incremental semantics for each phase of repair processing. As soon as "uh" is consumed at T1, a $FwdProblem(User, \text{cont})$ dialogue act is added to the ctxt list of moves in the right-most ParseIU in the graph. This means the information that the user has a forward looking problem continuing after the semantic content in the cont field has been constructed will immediately become available for other processes in the dialogue model. There is no semantic change to the cont field as the edit term is a communicative act rather than one tied to specific lexical meaning, and the previous *Assertion* dialogue act is also preserved as there is no evidence this should be revoked at this point. When the repair onset "loves" is processed, the lack of ability to parse from the right-most ParseIU's tree causes a repair to be inferred at T2. At T3, this step solves the continuation problem (Levelt, 1989) in repair interpretation– that of finding the extent of the reparandum and how far the parser should backwards search in order to yield a felicitous parse. The graph structure of the parse state allows for an easy backward search process over the sequences of DS-TTR actions keyed by each word.



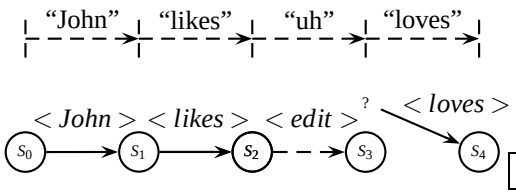
T0

$$\begin{bmatrix} \text{cont} = \begin{bmatrix} x1 & : e \\ x_{=John} & : e \\ e_{=likes} & : e_s \\ p1_{=obj(e,x1)} & : t \\ p_{=subj(e,x)} & : t \end{bmatrix} \\ \text{ctxt} = [Assert(User, \text{cont})] \end{bmatrix}$$



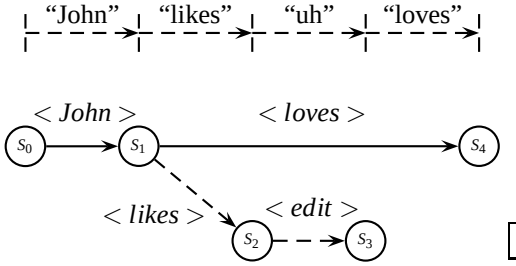
T1

$$\begin{bmatrix} \text{cont} = \begin{bmatrix} x1 & : e \\ x_{=John} & : e \\ e_{=likes} & : e_s \\ p1_{=obj(e,x1)} & : t \\ p_{=subj(e,x)} & : t \end{bmatrix} \\ \text{ctxt} = \begin{array}{l}[Assert(User, \text{cont}), \\ FwdProblem(User, \text{cont})] \end{array} \end{bmatrix}$$



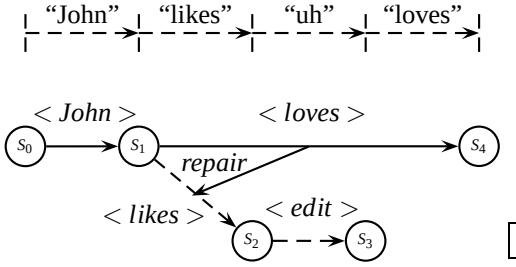
T2

$$\begin{bmatrix} \text{cont} = \begin{bmatrix} x1 & : e \\ x_{=John} & : e \\ e_{=likes} & : e_s \\ p1_{=obj(e,x1)} & : t \\ p_{=subj(e,x)} & : t \end{bmatrix} \\ \text{ctxt} = \begin{array}{l}[Assert(User, \text{cont}), \\ FwdProblem(User, \text{cont})] \end{array} \end{bmatrix}$$



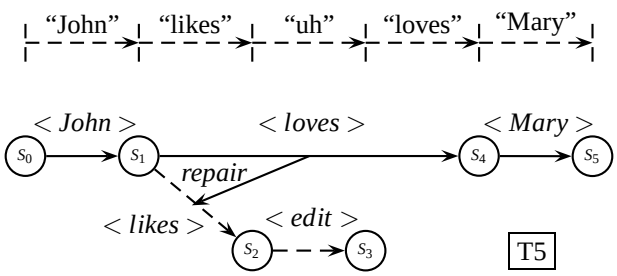
T3

$$\begin{bmatrix} \text{cont} = \begin{bmatrix} x_{=John} & : e \\ e & : e_s \\ p_{=subj(e,x)} & : t \end{bmatrix} \\ \text{ctxt} = \begin{array}{l}[Assert(User, \text{cont}), \\ Revoke(User, \begin{bmatrix} x1 & : e \\ e_{=likes} & : e_s \\ p1_{=obj(e,x1)} & : t \end{bmatrix})] \end{array} \end{bmatrix}$$



T4

$$\begin{bmatrix} \text{cont} = \begin{bmatrix} x1 & : e \\ x_{=John} & : e \\ e_{=loves} & : e_s \\ p_{=obj(e,x1)} & : t \\ p_{=subj(e,x)} & : t \end{bmatrix} \\ \text{ctxt} = \begin{array}{l}[Assert(User, \text{cont}), \\ Revoke(User, [\ e_{=likes} : e_s\ ] \\ \wedge\neg[\ e_{=loves} : e_s\ ])] \end{array} \end{bmatrix}$$



T5

$$\begin{bmatrix} \text{cont} = \begin{bmatrix} x1_{=Mary} & : e \\ x_{=John} & : e \\ e_{=loves} & : e_s \\ p_{=obj(e,x1)} & : t \\ p_{=subj(e,x)} & : t \end{bmatrix} \\ \text{ctxt} = \begin{array}{l}[Assert(User, \text{cont}), \\ Revoke(User, [\ e_{=likes} : e_s\ ] \\ \wedge\neg[\ e_{=loves} : e_s\ ])] \end{array} \end{bmatrix}$$

Figure 6.9: Strongly incremental interpretation of repair in DyLan. *Repaired* ParseIUs are dashed.

Once this is found, the speaker's commitment to the reparandum's semantic contribution can be taken back with the addition of the *Revoke* act in the ctxt field, which revokes the difference between the cont fields of ParseIU edges $S_1 \rightarrow S_2$ and $S_1 \rightarrow S_4$. Finally, at T4, the precise difference between the repair phase and the reparandum phase is computed which updates the current *diff* argument of the $Revoke(User, diff)$ dialogue move in the ctxt field.

This was a sketch of the processing steps and incremental semantic construction for a typical substitution repair, now I detail the mechanisms the model uses to deal with the different types of repair that have been observed empirically in this thesis.

6.4.1 Edit terms and interregna

I follow Ginzburg et al. (2014)'s intuition that disfluencies with the surface form of repeated words, filled pauses or discourse markers are time-filling devices signalling communicative trouble. The recognition of an edit term is done by a two-fold process: firstly, recognition of a parsing disfluency from a lack of parse from the normal lexicon, and secondly, a check that the word has the phonetic form of an edit term. The last step is made possible by adding to DyLan's resources a sub-lexicon for English edit terms $EditTerms$, which are those found in the corpus study in Chapter 4. As for an edit term's semantic update effects, as Figure 6.9 shows at T1 they result in a forwards-looking problem dialogue act $FwdProblem$ being appended to the context list of dialogue moves (ctxt field of the right-most ParseIU). The edit term does not change any semantic or syntactic content outside of this, so a ParseIU is created that is a copy of the right-most ParseIU but TTR asymmetric merged such that the dialogue act information of the problem signal $FwdProblem(Speaker, \text{cont})$ is the most recent move, and also with an empty list for the actions field, due to the fact no new DS actions have been run. No semantic information built up in the right-most cont field is revoked, however, if a repair onset follows, revocation of constructed semantic content may be indicated in the following ParseIU as will be explained below. Edit term IUs simply follow consecutively in the ParseIU graph from the right-most IU, and are *same level link*ed to the previously right-most IU. They will be given *repaired* status but without a repair link, as they do not have reparanda.

6.4.2 Repeats

Repeat repairs, the most common type of repair structure, have a similar dialogue context effect to edit terms– their recognition triggers a $FwdProblem$ move to be interpreted and they do not

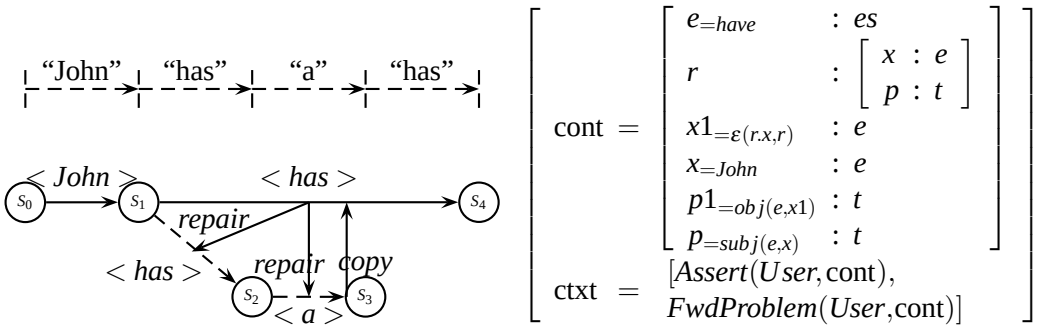


Figure 6.10: Repeat repair onset using ParseIU copying for "John [has a + has ..."

change the current semantic content built up in the utterance. As with edit terms, the initial cause of the recognition of their onset is a failure in the DS-TTR parsing process to construct a tree from the right-most ParseIU's tree and the consequent processing again makes use of a copying of the right-most ParseIU (the reparandum end IU), again with the asymmetric merge of the appended dialogue act information of the problem signal $FwdProblem(Speaker, \text{cont})$. However their recognition and subsequent action copying and integration into the ParseIU graph uses a different process, due to the fact they may last several words and are not treated as syntactically atomic like edit terms: a string of edit terms triggers concatenated copied $FwdProblem$-containing IUs without requiring comparison to a reparandum as no repair has been processed, while the situation is different for repeats. The ParseIU triggered by a repeat repair's onset is integrated as the successor of the ParseIU preceding the start of the reparandum, effecting the same backwards search as for the substitution repairs as in Figure 6.9. The difference in processing terms to substitution or delete repairs is that no parsing needs to take place– a word identity check of the current word with preceding words in the word graph, if positive, triggers the ParseIU keyed by the word before the repair onset to be copied (including its actions field) and then same level linked to the ParseIU before the reparandum onset. See Figure 6.10 for the state of the model after the partial utterance "John [has a + has ...", with the right-most ParseIU's cont and ctxt fields. Note the cont field has not revoked the information caused by parsing "a", the ε binder for the restricting embedded RT *R*– it is only if the following word conflicts with this information that this will be over-written. The default strategy is to assume this is the beginning of a repeat.

Also, *repair links* are added going from the repair onset word to each of the IUs in the repeat's reparandum. By definition, repair links make the receiving IU *repaired*. Should the

repeat be multiple words long, the word matching and IU copying process can continue word-by-word, and the repair link of the predecessor edge allows for a pre-parsing step to check for an unfinished repair, and if a repeat, the copying of the ParseIU removes the need to parse the word afresh. The model predicts the finding people can be faster when processing repeated repair words than fluent ones (Brennan and Schober, 2001), a finding that should also predict a speaker's faster production of them. Lexical retrieval and re-parsing is not required, as the copying process takes care of this.

6.4.3 Substitutions

Detection and interpretation of substitutions follows the incremental process as shown in Figure 6.9 as described above. The semantic representation obtained for substitutions is motivated by the extra information shown to be used by dialogue participants in repair processing (Brennan and Schober, 2001; Ferreira et al., 2004), making these repairs more informative than simply the right-most repair's 'cleaned' semantic content equivalent to a fluent utterance. As Ginzburg et al. (2014) argue, it is clear implicatures are obtained from revoking the reparandum and replacing it. Example (6.22) repeated from (4.8) in Chapter 4 demonstrates this.

(6.22) "So, a lot of our clients are [oil companies, + big oil companies]"
(sw4330)

The extra information derived in addition to the content of the cleaned utterance "So, a lot of our clients are big oil companies" is that the fact that the speaker is talking about big oil companies as opposed to those that are not big, i.e. any commitment to the fact they are not big is revoked overtly. I achieve this type of inference through use of the RT difference operation (see Algorithm 2 and (6.15) and (6.16)) between the reparandum end's IU's cont field and that of the repair onset's cont field. If the substitution is longer than one word then this difference calculation is recalculated with each new repair word's constructed ParseIU. Without this comparison strategy, the extra inferences would not become available. Operationally, the difference between the right-most ParseIU before the repair onset (i.e. the reparandum end) S_{n-1} and the repair onset ParseIU S_n is calculated then integrated as revoked information in S_n in two simple steps:

1. $diff = (S_{n-1}.\text{cont} \boxed{\wedge} S_n.\text{cont}) - S_n.\text{cont}$
2. $S_n = S_n \wedge [\ \text{ctxt} = [Revoke(Speaker, diff)]\ ]$

The reason for the asymmetric merge in step 1 will become clearer when discussing elliptical repairs, but the motivation here is to recycle as much of the content built up as possible, consequently not revoking constructed semantic information unless a repair's content over-writes it.

Another strength of this repair strategy is that the contextual information available in the reparandum phase is not removed, just downgraded to *repaired* status. I use this status to stipulate a modification to DS's mechanisms for ellipsis and anaphora resolution: the context they use in this model is not just the linear path back to the root of the DS Parse DAG but one which includes any repaired ParseIUs. This strategy therefore allows the parsing of example (6.5) "the interview was.. it was alright", with the correct anaphora resolution of "it". The ParseIU edges with the DS-TTR tree and cont field containing the semantics for "the interview" are accessible and DS anaphora mechanisms using context may run as normal. In other words, any committed preceding edge on the word hypothesis graph can be accessed (i.e. any word or partial word heard in the speaker's speech stream) as can its corresponding *grounded in* ParseIU.

Another aspect of the model I propose is that the parsing of self-repairs can be triggered not only by syntactic disfluency but also by *pragmatic infelicity*. For example, if given a unique object selection task where the user is only permitted to pick one object the user were to say "I pick the yellow square and the blue square", while this may be parsable in the DS grammar without repair backtracking, the repair onset mechanism will still work in the same way because in this micro-domain there is no available concept that represents the user selecting both the yellow and blue squares simultaneously in one turn. This is done via a step of checking whether the cont field's RT subsumes (is a supertype of) any domain concepts D specified in the conceptualiser.

6.4.4 Deletes and restarts

For deletes and restarts, the model is in fact not any different to that for substitutions, except in parsing deletes, one would expect different types of *diff* values in the $Revoke(Speaker, diff)$ repair moves. This is yet to be tested in detail, but one would expect a difference in the type similarity between the reparandum and repair phases. One would expect larger RTs in the conjuncts of the RT difference operation (Algorithm 2) that calculate *diff* in deletes. Substitutions are likely to re-use structures built up already in terms of DS tree building, while deletes are likely to have stronger over-writing effects. The *Similarity* function (6.17) could be also used to give a real value to the difference between reparandum and repair content, motivated by the fuzziness

between the interpretation of mid-utterance deletes and substitutions shown in Chapter 4.

6.4.5 Defining self-repair for NLU

To unify the processing for the different types of repair discussed above, it is possible to define a self-repair onset function for NLU in our model, as in Algorithm 3, here simplified without including the appropriate repair links, grounded in links and repaired statuses that are assigned during the process. It takes the state of the current word graph $W_0 \dots W_n$, the state of the current ParseIU graph $S_0 \dots S_{n-1}$ and the current speaker *Speaker* as its arguments. The function used in this algorithm for parsing, which is important for the system as a whole, is that of *BestParse*, which selects the best ParseIU to be constructed given the last word input. This gives DyLan a best-first quality in its parsing operation, and it relies not just on syntactic felicity but also on semantic and pragmatic felicity. The best parse for a given word W_n is yielded by selecting the ParseIU constructed by it which has the combination of the most complete tree, i.e. that with the largest proportion of nodes which are type complete, a measure notated here as $\mathit{Completeness}(S_n^j.\text{tree})$, and whose top node (and cont field) has the highest similarity score (see (6.17)) to an RT R_x in the domain concepts D, notated here $\mathit{Similarity}(S_n^j.\text{cont}, R_x)$. The definition is given in (6.23).

(NLU)Select ParseIU S_n^j triggered by word W_n according to:

$$\arg\max_{S_n^j} \mathit{BestParse}(S_n^j \mid W_n) = \arg\max_{S_n^j, R_x \in D} \mathit{Completeness}(S_n^j.\text{tree}) \times \mathit{Similarity}(S_n^j.\text{cont}, R_x) \quad (6.23)$$

Note how in Algorithm 3 if an edit term is encountered there is no need to backtrack, otherwise repeats, substitutions and deletes are processed by the backwards search consistent with a local model for self-repair backtracking found in corpora (Shriberg and Stolcke, 1998, Chapter 4). This function seems to give us the appropriate repair semantics for processing repair onsets. More detail of implementation of this function and mid-repair processing within the whole NLU algorithm is given in Section 6.7.

6.5 Generating self-repairs incrementally

Due to the reversible quality of the DyLan framework, all the types of repair that can be interpreted can be generated and the effects on the shared ParseIU context graph are very similar,

Algorithm 3 Self-repair onset function for NLU in DyLan

function SELF-REPAIR ONSET NLU($W_0 \dots W_n$; $S_0 \dots S_{n-1}$; *Speaker*)
  if from parsing word W_n an edge S_n cannot be constructed (cannot be parsed)
  or there is no $S_n : R_x$ for $R_x \in D$ (no concept in the domain) **then**
    if $W_n \in EditTerms$ **then** ▷ Edit term. Copy edge but with no actions.
      $add(S_n)$ where $S_n = S_{n-1}$ $\boxed{\wedge}$ $\left[\begin{array}{ll} \text{actions} & = [] \\ \text{ctxt} & = append(S_{n-1}.ctxt, FwdProblem(Speaker, \text{cont})) \end{array}\right]$
      $SameLevelLink(S_n, S_{n-1})$; **Return** ▷ Order in normal left-right way.
    for backwards search $i = n-1 \rightarrow 0$ **do** ▷ Not edit term, then backwards search.
      if W_n matches W_i **then** ▷ Repeat. Copy edge.
        $add(S_n)$ where $S_n = S_i$ $\boxed{\wedge}$ [ctxt $= append(S_i.ctxt, FwdProblem(Speaker, \text{cont}))$]
        $SameLevelLink(S_n, S_{i-1})$; **Return** ▷ Order appropriately.
      else try parse W_n from S_{i-1} ▷ Not a repeat or edit term.
        if successful parse **then** ▷ Substitution or delete.
          $add(S_n)$ where S_n = arg $\max_{S_n^j} BestParse(S_n^j | W_n)$ ▷ See (6.23).
          $diff = (S_n.\text{cont}$ $\boxed{\wedge}$ $S_{n-1}.\text{cont}) - S_n.\text{cont}$ ▷ RT diff. to reparandum end IU.
          $S_n = S_n$ $\boxed{\wedge}$ [ctxt $= append(S_n.ctxt, Revoke(Speaker, diff))$] ▷ Revoke act.
          $SameLevelLink(S_n, S_{i-1})$; **Return** ▷ Order appropriately.
        else Continue ▷ Keep backtracking if no parse.
end function

in keeping with the generators as parsers approach in DS (Purver and Kempson, 2004). What follows is a tactical NLG account of repair generation, but a more strategic account of why a conceptualiser decides to generate repairs for communicative reasons is given in the next chapter.

In interaction, the generator's goal concept may be revised shortly after, or even during, the generation process, so trouble may be encountered during realisation of an on-going utterance. A repair onset function should operate if there is no possible DAG extension from the right-most ParseIU after the semantic filtering stage of generation (i.e. removal of ParseIU paths that do not subsume the goal concept), but this will happen in different ways depending on the type and timing of that change.

6.5.1 Edit terms, interregna and covert repairs

The generation of edit terms is practically useful for timing purposes and has user experience benefit in dialogue systems (Skantze and Hjalmarsson, 2010; Buschmeier et al., 2012). This is the preferred strategy for stalling due to incomplete formulation or conceptualisation, which reflects the prevalence of non-repair edit terms over interregna as shown in Chapter 4. Given the way the model operates in NLG mode by sending goal concept IUs to the DS-TTR generator,

there may be cases where the integration of the new material may be costly, for instance having to initialise a full lexical search– i.e. generate a new sub-lexicon as in (6.21). In these cases where a timing delay to formulation may take place, an edit term is selected randomly from a distribution in the *EditTerms* resource and added to the output word graph. Given, in theory, vocalisation (or synthesis) should be able to happen asynchronously in a separate downstream process as generation continues (an approach implemented by Buschmeier et al. (2012)), the vocalisation of the edit term can span the time taken to complete the lexical search and generation of the next word. Another cause for edit term generation is if the pointer is in a DS-incomplete tree after exhausting its generation possibilities given the goal concept and no further words can be parsed, for example "You go to {uh}…". In these cases the inference from the generator is that there is a more specified goal concept to come, and in these cases, rather than simply pausing, an edit term can be generated while the rest of the concept is constructed.

Either of these two edit term scenarios can be completed successfully without backtracking before generating the next word as in (6.2) "I go to Paris {uh} from London", which I call an *extension*, or they can serve the purpose of interregna for repairs. In (6.2) the DS parser would treat the additional information after the edit term as monotonic growth of the matrix tree through LINK ADJUNCTION, resulting in subtype extension of the root RT. Thus, a change in goal concept during generation will not always put demands on the system to backtrack.

The semantic update effect of generating an edit term is identical to parsing, with a *FwdProblem* generated in the ParseIU created, so the generator has a record of this even though it was not part of the original goal concept but emergent from the necessity to generate a problem signal.

In addition to overt problem signalling, *covert repairs* can be made where the goal concept has changed, but there is sufficient time to do the appropriate lexicalisation before generating the next word without the need of an edit term. These will not update the ParseIU graph with a *FwdProblem* or *Revoke* act containing ParseIU, as this change in generation path is not in the public information state.

6.5.2 Repeats

Repeats are similarly interpreted as forward-looking difficulty as described above, and in generation, these may also be used for time-buying strategies. The time-buying strategy using repeats is consistent with Shriberg and Stolcke (1998)'s empirical model of repeat repairs: a backwards search mechanism which will favour short repeats. However this is conditioned on different

parameters than simply length of the backwards search and length of the utterance so far.

To find out where natural retrace points for repeats lie, I generated repeats stochastically (not using DS, just by string copying) from a corpus of a user interacting via text chat with a simple chatbot that describes London buildings (Hough, 2009), according to Shriberg and Stolcke (1998)'s length-based model– see Section 2.3.5. From the resulting utterances, there were more natural and less natural surface forms that emerged:

More human-like repairs:

(6.24) "The Royal Naval College is situated [in, + in] Greenwich"

(6.25) "Shakespeare's Globe Theatre [is, + is] situated [on, + on] the south bank"

(6.26) "St Paul's Cathedral [was, + was] designed by Christopher Wren"

Less human-like repairs

(6.27) "Ask me some questions or click on the [objects, + objects] "

(6.28) "Yes of course I [can think because computers have, + can think because computers have] intelligence"

(6.29) "If no-one's [there, I, + there, I] may as well talk about the [buildings, + buildings] myself..."

I induced a qualitative regularity from these, which was that the more natural repeats were those which were either interrupted mid-constituent and traced back from the beginning of the constituent (6.25), and the less natural ones were either those whose interruption was at a constituent boundary and traced back to a mid-constituent position (6.27) or those which crossed over a coordination phrase boundary such as (6.28). This seems consistent with the observations in Chapter 4 and Healey et al. (2011)'s experimental findings.

Due to these observations I define a generation tactic only to repeat when the DS pointer is in a type-incomplete DS sub-tree and the backtracking only extends to the creation of the highest incomplete node above the pointer. This way complete sub-tree boundaries cannot be crossed. This approach is consistent with the principle of least communicative joint effort (Clark, 1996), as solving the continuation problem for the hearer is also made easier with this approach as it minimises search. The difference for the NLU task of interpreting the repeat is that it must keep track of which words have been matched, made possible by repair links, whereas in NLG the

match can be made before the words are realised because repair links are added from the repair phase to the reparandum before they are committed to the output word graph.

6.5.3 Substitutions

It is only a semantics or syntax mismatch, where the revised goal concept RT does not subsume a permissible extension of a DS tree in the right-most ParseIU in the DAG, where revoking repair action will occur and substitution and delete surface forms will be generated. Given the reversibility of DyLan, one could take the time serial process shown in Figure 6.9 to be one for generation but taking the right-hand record as the generation goal. However, one could also argue for asymmetry to the NLU account. A *Revoke* act is not required to be part of the goal concept (i.e. the strategic generation or conceptualisation) but it may emerge in the tactical part of the generation which this chapter focusses on: the generator may find the only way to successfully construct the goal concept in the cont field and an *Assert* move binding the cont field in the ctxt field is through repair. There is just the side-effect of backwards search causing revocation. Given this, the subsumption relation check over both cont and ctxt fields of a candidate ParseIU to the goal concept can be relaxed in that it should not apply to *Revoke* and $FwdProblem$ acts, and that in the general case, in order to aim for the minimal amount of repair, the selection of non-repairs can be stipulated to be preferred by using the *Similarity* function (6.17) between the goal concept *GC* and a candidate ParseIU edge for generation *C*, i.e. $Similarity(GC, C)$ and preferring candidates with higher similarity scores. This way, in general goal concepts can be assertion or question content, and the added heuristic during generation is to be as fluent as possible, however the relaxation of subsumption of the overall goal concept does not mean repaired IUs are removed, as repair may be the only method for achieving the goal. Of course, the intention to revoke explicitly can also be part of the goal concept, and in which case appropriate *Revoke*-containing IUs will be selected before those that do not contain this, but in general this need not be the case. It is worth mentioning that downstream vocalisation (synthesis) may require the *Revoke* information to alter the prosody of its utterance according to repair structure: given the empirical evidence that speakers do alter repair onset prosody compared with fluent words (Brennan and Schober, 2001), it is clearly important to preserve the *Revoke* status on an IU if we want to achieve natural sounding output from the dialogue system.

Generation of a substitution repair can be shown in Figure 6.11 where the changing goal concepts on the right-hand side are just *Assert* moves, however *Revoke* moves are required for the

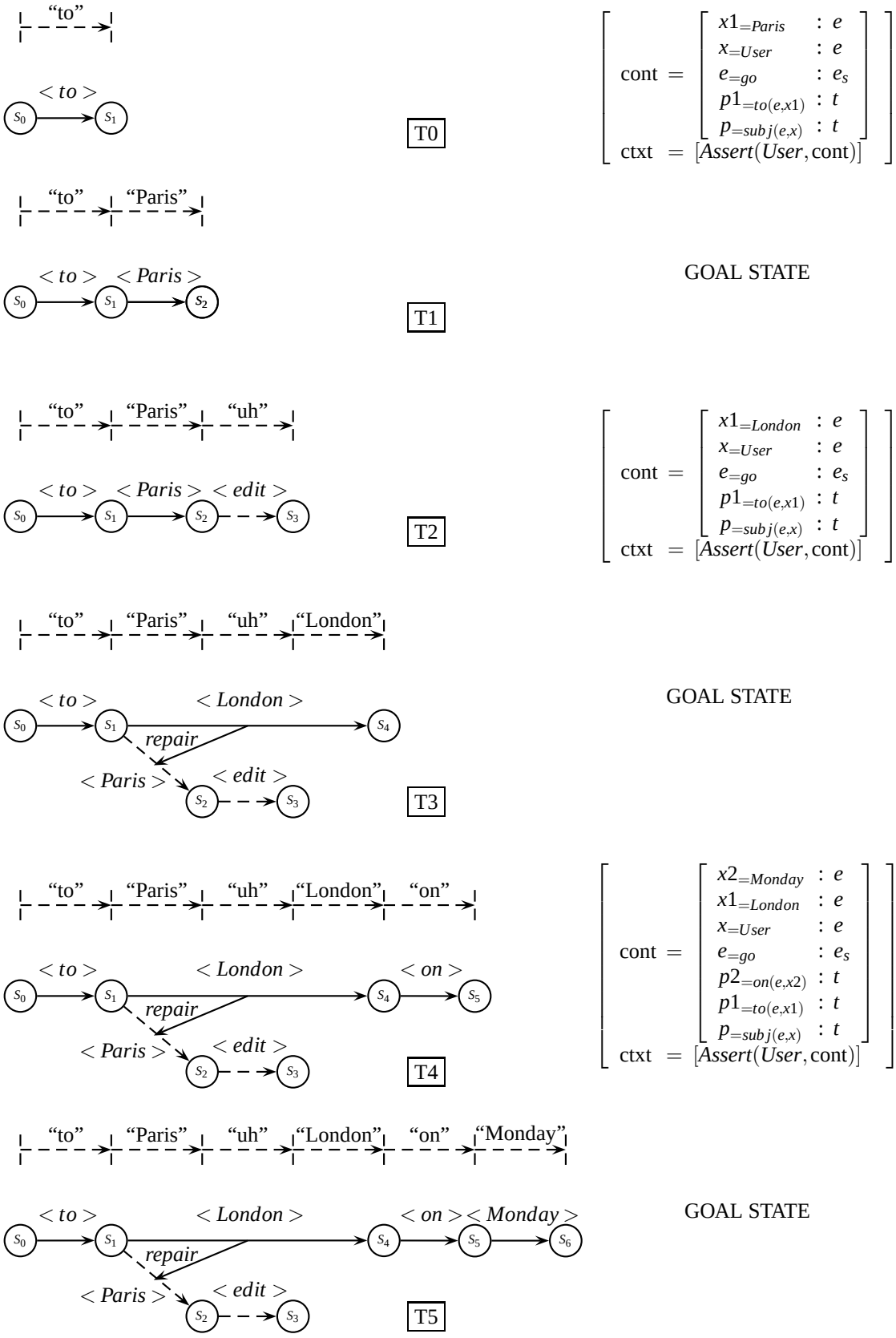


Figure 6.11: Strongly incremental generation of repair in DyLan. Goal concept changes cause substitution repair at T2 and covert repair at T4. *Repaired* ParseIUs are dashed.

appropriate asserted content to be realised, and these would present in the ParseIUs in the shared parsing and generation ParseIU graph (not shown here). The utterance "to Paris, uh, London on Monday" is generated here. The initial goal concept is achieved at T1 and the goal concept changes at T2 such that sub-lexicalisation has to be run again, and given the lack of possible parse available from the right-most ParseIU, an edit signal is added whilst the backwards search takes place. When the backwards search and parse is successful at T3 the goal is met (despite the right-most ParseIU in the ParseIU graph, not shown, now containing a *Revoke* act). At T4 there is another change in goal concept, but because no backwards search has to take place, the extension can be covert without having to generate an error signal, finally the generator meets the new goal at T5.

6.5.4 Deletes and restarts

As was the case for NLU, I define the NLG model not to generate delete and restart surface forms differently from substitutions. The $\mathit{Revoke}(\mathit{Speaker}, \mathit{diff})$ moves caused in the ParseIU graph should again have different sorts of *diff* values to substitution repairs, as more semantic material should have been revoked, however this needs to be tested empirically. One possibility for further development of this is to define the decision to restart from the beginning of the ParseIU tactically, that is to say, the decision is made to restart if certain criteria are met, otherwise defaulting to the backwards search technique. This tactic to restart may be driven by too drastic a difference between the current right-most ParseIU's cont field and the new goal concept to warrant an attempt to revoke the constructed semantic content through substitution, again consistent with a principle of making life easier for both the generator and the parser. This is yet to be integrated into the proposed algorithm, but could be promising to explore.

6.5.5 Defining self-repair for NLG

It is now possible to define a function for repair onset initiation (including covert repair initiation) given changing goal concepts for the `DyLan` generator as follows. Given a new goal concept GC_n has been added to the concept graph and *SubLex* is the set of words created by the sublexicalisation check (6.21) from GC_{n-1}, the function applies as in Algorithm 4, again here simplified without the appropriate repair links, grounded in links and repaired status assignment. As with NLU, we need a notion of best-first search in generation for selecting the best word and ParseIU possible, and this can be done, given the contigency of NLG on NLU in `DyLan`, through

Algorithm 4 Self-repair onset function for NLG in DyLan

function SELF-REPAIR ONSET NLG($GC_0 \dots GC_n$; $S_0 \dots S_{n-1}$; *SubLex*; *Speaker*)
  if from parsing all words $W_l \in$ *SubLex* an edge S_n cannot be constructed (cannot be parsed)
  or there is no edge S_n such that $GC_n \sqsubseteq S_n$(no subsumption of goal concept) **then**
    SubLex = *SubLexicalise*(GC_n) ▷ Repopulate the sublexicon.
    try Parse $\sum_l W_l \in$ *SubLex* from S_{n-1} ▷ Check if an extension or covert repair possible.
    if Parse of W_l is successful **then** ▷ Extensible.
      if address($\diamondsuit$) in $S_{n-1}.tree$ contains ?*Ty* **then** ▷ Inside an incomplete tree? Repeat.
        add(S_n) where $S_n = S_{n-1} \curlywedge [\ \text{ctxt} = append(S_{n-1}.ctxt, FwdProblem(Speaker, \text{cont}))\ ]$ ▷ Copy edge.
        add(W_n) where $W_n = W_{n-1}$ ▷ Repeat; one word default.
        SameLevelLink(S_n, S_{n-2}); *SameLevelLink*(W_n, W_{n-1}) ▷ Order appropriately.
        $n = n+1$ ▷ Increment as added an IU.
      Where S_n, W_n = arg max$_{S_n^j, W_n^k}$ *BestGenerate*($S_n^j, W_n^k | GC_n$): ▷ See (6.30).
      add(S_n) to ParseGraph; *add*(W_n) to WordGraph
      SameLevelLink(S_n, S_{n-1}); *SameLevelLink*(W_n, W_{n-1}) ▷ Order left-right.
      Return
    for backwards search $i = n-1 \rightarrow 0$ **do** ▷ No extension possible, need to backtrack.
      if $i < n-1$ **then** ▷ If longer than one word backtrack generate random edit term.
        add(S_n) to ParseGraph where $S_n = S_{n-1}$ ▷ Just copy, do not re-parse.
        $S_n = S_n \curlywedge \left[\begin{smallmatrix} \text{actions} & = & [] \\ \text{ctxt} & = & append(S_n.ctxt, FwdProblem(Speaker, \text{cont})) \end{smallmatrix}\right]$ ▷ Fwd act. No actions added.
        add(W_n) to WordGraph where $W_n = random(EditTerms)$
        SameLevelLink(S_n, S_{n-1}); *SameLevelLink*(W_n, W_{n-1}) ▷ Order left-right.
        $n = n+1$ ▷ Increment as added an IU.
      try Generate $\sum_l W_l \in$ *SubLex* from S_{i-1} ▷ Substitution or delete. Generate.
      if successful generation of W_n,S_n from S_{i-1} **then**
        $diff = (S_{n-1}.\text{cont} \boxed{\curlywedge} S_n.\text{cont}) - S_n.\text{cont}$ ▷ RT diff. to reparandum end IU.
        $S_n = S_n \curlywedge [\ \text{ctxt} = append(S_x.ctxt, Revoke(Speaker, diff))\ ]$; **Return** ▷ Revoke act.
      else Continue ▷ Keep backtracking if no word generated.
end function

selecting the word that both maximises its *BestParse* score given the ParseIU context and also its closeness to the goal concept GC_n. The definition is given in (6.30).

$$(NLG)\text{Select WordIU } W_n^k \text{ and ParseIU } S_n^j \text{ triggered by goal concept } GC_n \text{ according to:}$$

$$\arg\max_{S_n^j, W_n^k} BestGenerate(S_n^j, W_n^k | GC_n) = \arg\max_{S_n^j, W_n^k} BestParse(S_n^j \mid W_n^k) \times Similarity(GC_n, S_n^j) \tag{6.30}$$

6.6 Ellipsis in repair processing

While the model described covers repeats, substitutions, deletes, extensions and covert repairs, it also has the ability to deal with repairs involving ellipsis, such as the following, from (6.6) above:

(6.31) "Peter went [swimming with Susan, + {or rather,} surfing] ..."

I assume that at the end of this part of the utterance the semantics for the right-most ParseIU, rather than being as in (6.32), will be as in (6.33).

(6.32)
$$\left[\begin{array}{lll} \text{cont} & = & \left[\begin{array}{ll} e_{=surf} & : e_s \\ x_{=Peter} & : e \\ p_{=subj(e,x)} & : t \end{array}\right] \\ \text{ctxt} & = & [Revoke(Speaker, \left[\begin{array}{ll} e_{=swim} & : e_s \\ x1_{=Susan} & : e \\ p1_{=with(e,x1)} & : t \end{array}\right])] \end{array}\right]$$

(6.33)
$$\left[\begin{array}{lll} \text{cont} & = & \left[\begin{array}{ll} e_{=surf} & : e_s \\ x1_{=Susan} & : e \\ x_{=Peter} & : e \\ p1_{=with(e,x1)} & : t \\ p_{=subj(e,x)} & : t \end{array}\right] \\ \text{ctxt} & = & [Revoke(Speaker, [\ e_{=swim} : e_s\ ])] \end{array}\right]$$

This is to say the information about the event being with Susan is not revoked. This is of course defeasible if conflicting information were to follow in the next word.

This example is different to the previous ones in another respect: it is not clearly syntactically disfluent. In many grammars, including DS, this could be parsed as a fluent non-repaired coordination with "or rather" as the coordinator: if the coordinator was simply "or" this would not necessarily be a repair, rather an enrichment to the effect that the speaker did not know whether it was a swimming or surfing event, and DS would in fact use ellipsis resolution mechanisms as will be explained below. However the 'rather' here lexically makes the 'or rather' an editing phrase instead of a coordinator. The DS ellipsis mechanisms can be used to get the correct semantics for the repair phase 'surfing' as it would for ellipsis in a fluent utterance as will be explained, however what is actually giving the reparandum its repaired status is the lexical content of the editing phrase, in addition perhaps to prosodic factors indicating contrast, which are beyond the scope of this thesis.[8]

Despite this hedge about the cause of the repair backtracking in the linguistic context for this example, the model set out above still allows (6.33) to be constructed directly at the repair onset. The difference between reparandum and repair is computed as the difference between the asymmetric merge of the current right-most (reparandum end) ParseIU's cont field with the new information from re-parsing, to the new information– see the two-step process in Section 6.4.3.

[8]Thanks to Nick Asher for an intensive discussion on this example in my viva.

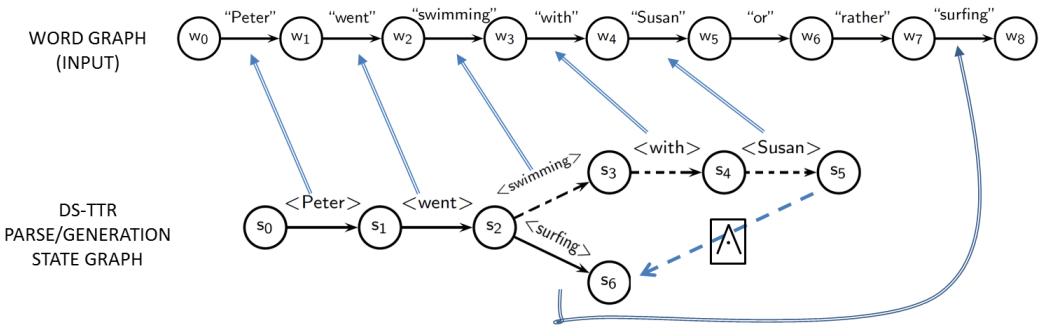


Figure 6.12: Incremental interpretation of elliptical self-repair in DyLan using TTR *asymmetric merge*

This means only information in the reparandum is over-written if it clashes with the new information. This operation is shown schematically in Figure 6.12.

So during the interpretation of the self-repair in Figure 6.12 the asymmetric merge takes place between the cont field at edge $s_4 \rightarrow s_5$, the reparandum end, and that at edge $s_2 \rightarrow s_6$, the repair onset. This way maximal semantic content can be recovered from context where possible, and the information about the information that the event happens 'with Susan' is not lost if that does not conflict with our main predicate, which it does not in this case.

This is consistent with Ferreira et al. (2004)'s observation of recycling of semantic argument structure from the reparandum phase, even if it is not appropriate with respect to the meaning of the repair. There may be other more pragmatic factors at work which would favour a complete over-writing effect here, but this must be influenced from a more complex model of the situation– I discuss how situation-level information might interface with elliptical repairs in the next chapter.

It is also worth mentioning another way of achieving the same incremental semantics here through using DS ellipsis mechanisms. In this case this can be achieved by reusing preceding action edges in the spirit of the recent DS account of verb phrase ellipsis (VPE) (Kempson et al., 2011, forthcoming). Schematically, the repair mechanism would work in a similar way to the resolution of the VPE in "[Peter went] swimming [with Susan] and $_{<Ai-Aj>}$ surfing $_{<Ak-Al>}$", where $_{<Ai-Aj>}$ is the sequence of action edges used in the construction of the DS-TTR parse DAG triggered by the words "Peter went" and $_{<Ak-Al>}$ is the sequence for "with Susan", both of which are re-used either side of the actions triggered by "surfing", however in the repair case only the

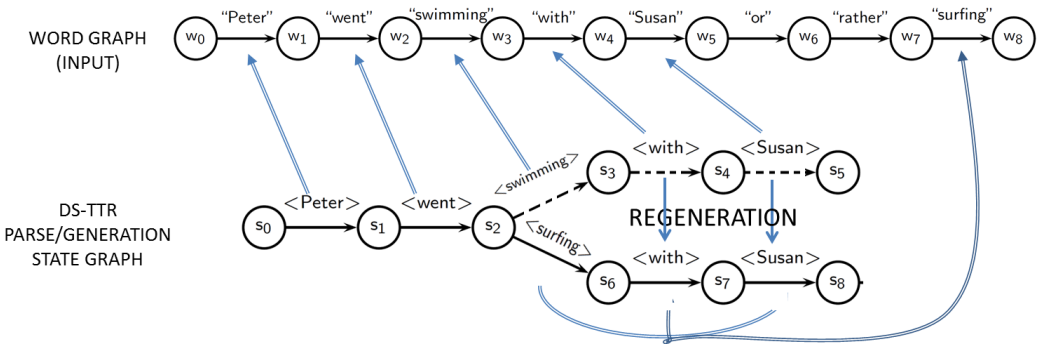


Figure 6.13: Incremental interpretation of self-repair in DyLan using DS re-running of actions

latter sequence needs to be run (see Figure 6.13). The REGENERATION (Kempson et al., 2011, forthcoming) rule operating over DS context makes this possible here, enabling the parser (and generator) to take a sequence of actions from context and re-run them, provided that they were triggered by the same type-requirement as is imposed on the node currently under development in the right-most tree in the parse path.

The difference between the VPE and repair mechanisms is the commitment status of the edges in the DS parse paths: in VPE the parse edges yielded from both construction paths would be added to the ParseIU graph context, whilst in repair, although the reparandum's path remains accessible to the parsing and generation modules, only the repair phase's parse path would remain *added*, while the reparandum would be have *repaired* status. A DS driven account is yet to be fully fledged out for these repairs, however the potential for extending the self-repair model provided by the DS characterization of context is clear.

The advantage of taking a DS-first approach is the possibility of extending other DS ellipsis resolution mechanisms to their repair analogues. Here I present some more complex types of ellipsis from Kempson et al. (forthcoming) and transform them into repair equivalents to illustrate this:

(6.34) *(VPE anaphora)* "I asked the woman who had been hand-gliding how long she had been doing so, and she asked me how long I had."

(Repair equivalent) "[I asked the woman who had been hand-gliding how long she had been doing so, + { no sorry, } she asked me how long I had.]"

(6.35) *(Gapping)* "John is being interviewed for the Linguistics position today, Harry

tomorrow."

(Repair equivalent) "[John is being interviewed for the Linguistics position today, + { no sorry, } Harry, tomorrow.]"

(6.36) *(Left-dislocated bare argument ellipsis)* "Sue, John upset. Mary too."

(Repair equivalent) "[Sue, John upset, + { I mean }, Mary]"

DS ellipsis accounts are still developing and using them for ellipsis resolution within repairs seems a natural step. The DS account could give a nice parallel between the mechanisms used in normal ellipsis and the recoverability from context required in repair quite elegantly. However, the disadvantage of using a DS driven strategy for repair ellipsis is one of efficiency: the asymmetric merge difference can be re-computed for each incoming substitution repair word with uniform cost, whilst if actions are re-run and stored in context, should conflicting information be parsed in following words, the re-run actions would need to be removed from the parse path and the previous ParseIU re-computed. For this reason, while the accounts of repair ellipsis are not fully fledged out for the range of phenomena just illustrated, a simple and efficient solution seems to use the asymmetric merge approach (as in Figure 6.12), which can be used in all substitution and delete repair processing.

Generating repairs with ellipsis (and generating elliptical structures in general) in a strictly incremental way requires slightly more than the interpretation apparatus described here plus a reversal of the input and output, as normal DS arguments would insist. There needs to be a *strategic* reason why an elliptical form would be chosen over a fully pronounced one. One may use two established pragmatics frameworks to begin to begin to describe such a story, Gricean pragmatics (Grice, 1975) and Relevance Theory (Sperber and Wilson, 1986). A computational characterization of conversational relevance will be described in the next chapter, in which I argue probabilistic and information theoretic characterization of dialogue situations play a large role in repair and ellipsis.

6.7 Implementation: Redefining NLU and NLG algorithms with self-repair processing

Given the strategies described above, I now redefine incremental interpretation and generation algorithms to incorporate self-repair processing. Below algorithms are given for the whole NLU

and NLG processes, not just repair onsets, and the full apparatus of repair links between IUs and repaired status of IUs can be used for interpreting multi-word repairs. The motivation for repair links and statuses becomes clearer in the context of a dialogue system, particularly in light of changing ASR word hypotheses. When a word hypothesis is revoked whose grounded in ParseIUs have repair links stemming from them, these links will also be removed, as will the effect on the ParseIUs they repair link to as they will lose their *repaired* status.

The full algorithms for NLU and NLG as they incrementally receive input are given in Algorithms 5 and 6 in pseudocode.

6.7.1 Prototype NLG implementation

A prototype generation module has been implemented in Java according to Algorithm 6 as a module in the incremental dialogue framework Jindigo (Skantze and Hjalmarsson, 2010) to demonstrate its semantically-driven word-by-word generation process and self-repair capacity. This has been tested on a small corpus of goal concepts as described below. For generating self-repairs, `DyLan` is capable of generating substitutions and deletes such as "I go to Paris, uh, London" with goal concepts being artificially changed during generation. It yields the appropriate RT semantics as described above.

6.7.2 Computational efficiency

With regards to the computational complexity of the model, there are several factors that are relevant: the size of the lexicon $|\ lex\ |$, the branching factor $\$b$ of the number of computational actions applied from nodes in the parse DAG, the cost of subtype relation checking between two records $\$subtype$, and the cost of compiling semantic formulae for partial trees $\$partial$. The previous DS model in the worst case for generating utterances n words long would have a runtime complexity in the order of $n \times \$b \times |\ lex\ | \times \$partial \times \$subtype$ where $\$partial$ and $\$subtype$ become more expensive as the trees grow, whereas the new model's runtime is dependent on the initial pre-lexicalisation cost and the subsequent parsing and testing using a much smaller lexicon- this is $(|\ lex\ | \times \$subtype(1)) + (n \times \$b \times |\ SubLex\ | \times \$partial \times \$subtype)$ where $\$subtype(1)$ is a small value and constant[9] and $|\ SubLex\ |$ is significantly smaller than $|\ lex\ |$.

Initial results are encouraging in the implementation for generation– for goals deriving three word utterances (e.g. 'I like Paris') and six word utterances (e.g. 'I go to Paris from London'), in

[9]This is due to the small number of fields for checking in lexical actions.

Algorithm 5 NLU algorithm for DyLan with self-repair ability pseudocode

INPUT: (update from left buffer) WordIU $W_n \in$ WordGraph
(module internal) All ParseIUs $S_0 \dots S_{n-1} \in$ ParseGraph
OUTPUT: (update to right buffer) All edits $\in$ ParseGraph

(1) Best-first syntactic parse:
if W_n is utterance initial **then** ▷ Normal initial parse.
From DS axiom apply all possible computational actions CA*
Apply all lexical actions ΣLex keyed by word W_n
else if $S_{n-1} \sqsubseteq [\ \text{ctxt} : [FwdProblem(Speaker, \text{cont})]\ ]$ **then** ▷ Check for incomplete Fwd act.
if $RepairLinked(S_{n-1}, S_l \dots S_k)$
and W_n.matches($W_x \in W_l \dots W_k$) **then** ▷ Incomplete repeat repair, W_n is a repeat of W_x. Copy S_x. Do not re-parse.
$add(S_n^j)$ where $S_n^j = S_x \boxed{\wedge} [\ \text{ctxt} = append(S_x.ctxt, FwdProblem(Speaker, \text{cont}))\ ]$
$RepairLink(S_n^j, S_k)$ ▷ Repair links to reparandum-final word.
else if $W_n \in EditTerms$ **then** $add(S_n^j)$ where $S_n^j = S_{n-1}$; $Repair(S_n^j)$ ▷ Incomplete edit term. Copy predecessor.
$SameLevelLink(S_n^j, S_{n-1})$; $GroundedInLink(S_n^j, W_n)$; **Return** ▷ Order IUs.
else Apply all lexical actions ΣLex keyed by word W_n to S_{n-1}.tree ▷ Normal parse.
if no actions ΣLex can be parsed **then GoTo (3)** ▷ Syntax driven repair.
(i) Apply all possible computational actions CA* ▷ Normal DS-TTR parse adjust.
(ii) $add(S_n^j)$ where $S_n^j = \arg\max_{S_n^j} BestParse(S_n^j | W_n)$ ▷ Normal best ParseIU. See (6.23).
$SameLevelLink(S_n^j, S_{n-1})$; $GroundedInLink(S_n^j, W_n)$ ▷ Normal left-right ordering.

(2) Semantic type check:
if $S_n^j.\text{cont} \sqsubseteq [\ q : question\ ]$ **then** ▷ Add simple dialogue act classification.
$S_n^j = S_n^j \wedge [\ \text{ctxt} = [Question(Speaker, cont.q)]\ ]$
else $S_n^j = S_n^j \wedge [\ \text{ctxt} = [Assert(Speaker, cont)]\ ]$
Where $test = \begin{bmatrix} \text{cont} : S_n^j.\text{cont} \\ \text{ctxt} : S_n^j.\text{ctxt} \end{bmatrix}$, Select R_x where:
$\arg\max_{R_x \in D} Similarity(test, R_x)$ and $R_x : test$ ▷ Select maximally matched R_x in domain concepts D.
if no successful match R_x found **then**
$revoke(S_n^j)$; $j = j+1$; **GoTo (1) (ii)** ▷ Recurse from here until successful parse or DAG exhausted.
if all edges $\sum_j S_n^j$ are revoked **then GoTo (3)** ▷ Semantics driven repair.
if $RepairLinked(S_{n-1}, S_l \dots S_k)$ **then** ▷ Possibly incomplete substitution or delete repair.
$diff = (S_k.\text{cont} \boxed{\wedge} S_n^j.\text{cont}) - S_n^j\text{cont}$ ▷ RT difference to reparandum end IU S_k.
if not $Revoke(Speaker, diff) \in S_{n-1}$.ctxt **then** ▷ New revoke information.
$S_n^j = S_n^j \boxed{\wedge} [\ \text{ctxt} = append(S_n^j.ctxt, Revoke(Speaker, diff))\ ]$ ▷ Update revoke act.
$RepairLink(S_n^j, S_k)$ ▷ Add repair link to reparandum end IU.
Return

(3) Repair Onset:
if $W_n \in EditTerms$ **then** ▷ Edit term. Copy edge but with no actions. Fwd act.
$add(S_n)$ where $S_n = S_{n-1} \boxed{\wedge} \begin{bmatrix} \text{actions} = [] \\ \text{ctxt} = append(S_{n-1}.ctxt, FwdProblem(Speaker, \text{cont})) \end{bmatrix}$
$GroundedInLink(S_n^j, W_n)$; $SameLevelLink(S_n, S_{n-1})$; $Repair(S_n)$; **Return** ▷ Order in normal left-right way.
for $i = n-1 \to 0$ **do** ▷ Not edit term? Backtrack along the ParseIU graph.
if W_n.matches(W_i) **then** ▷ Start of repeat.
$add(S_n^j)$ where $S_n^j = S_i \boxed{\wedge} [\ \text{ctxt} = append(S_i.ctxt, FwdProblem(Speaker, \text{cont}))\ ]$ ▷ Just copy, do not re-parse. Fwd act.
$GroundedInLink(S_n^j, W_n)$; $SameLevelLink(S_n^j, S_{i-1})$ ▷ Link to previous edge before repeat onset.
$RepairLink(S_n^j, S_x)$ for all edges $S_x \in S_i \dots S_{n-1}$; **Return** ▷ Repair link to all reparandum.
else GoTo (1) for S_{i-1} ▷ Substitution or delete. Re-parse from this point.
if successful parse S_n^j from S_{i-1} **then**
$diff = (S_{n-1}.\text{cont} \boxed{\wedge} S_n^j.\text{cont}) - S_n^j.\text{cont}$ ▷ RT diff. to reparandum end IU.
$S_n^j = S_n^j \boxed{\wedge} [\ \text{ctxt} = append(S_n^j.ctxt, Revoke(Speaker, diff))\ ]$ ▷ Revoke act.
else Continue ▷ Keep backtracking if no parse.
$GroundedInLink(S_n^j, W_n)$; $SameLevelLink(S_n^j, S_{i-1})$ ▷ Link to previous edge before repair onset.
$RepairLink(S_n^j, S_x)$ for all edges $S_x \in S_i \dots S_{n-1}$; **Return** ▷ Repair link to all reparandum.

Algorithm 6 NLG algorithm for DyLan with self-repair ability pseudocode

INPUT: (update from left buffer) Goal ConceptIU $GC_n \in$ ConceptGraph
(module internal) All ParseIUs $S_0 \dots S_{n-1} \in$ ParseGraph
OUTPUT: (update to right buffer) All edits $\in$ ParseGraph; All edits $\in$ WordGraph

(1) Pre-verbal lexicalisation: **if** *SubLex* is empty **then** $SubLex = SubLexicalise(GC_n)$ ▷ See definition (6.21)

(2) Best-first syntactic parse:
if S_n is utterance initial **then** ▷ Normal initial parse in generation mode.
  From DS axiom apply all possible computational actions CA*
  Try to parse all lexical actions $\Sigma Lex \in SubLex$
else Try to parse all lexical actions $\Sigma Lex \in SubLex$ from S_{n-1}.tree ▷ Normal parse in generation mode.
if no actions ΣLex can be parsed **then GoTo (5)** ▷ Syntax driven repair.
for path-final nodes $d \in$ ParseDAG **do**
  Apply all possible computational actions CA* ▷ Normal DS-TTR parse adjust.
  $add(S_n^j)$ where $S_n^j = \arg\max_{S_n^j} BestParse(S_n^j|W_n)$ ▷ Normal best ParseIU.
  $SameLevelLink(S_n^j,S_{n-1})$; $GroundedInLink(S_n^j,GC_n)$ ▷ Normal linking for generation.

(3) Semantic type check and filter:
for path-final ParseIUs $\sum_j S_n^j \in$ ParseGraph **do**
  if S_n^j.cont $\sqsubseteq$ [q : *question*] **then** ▷ Add simple dialogue act classification.
    $S_n^j = S_n^j \wedge$ [ctxt = $[Question(Speaker, cont.q)]$]
  else $S_n^j = S_n^j \wedge$ [ctxt = $[Assert(Speaker, cont)]$]
  Subsumption check $GC_n.cont \sqsubseteq S_n^j$.cont ▷ Semantic content (cont field) only here.
  if Subsumption check fails **then**
    $revoke(S_n^j)$
if all edges $\sum_j S_n^j$ are revoked **then GoTo (5)** ▷ Semantics driven repair.

(4) Generate best word:
Where $S_n^j, W_n^j = \arg\max_{S_n^j, W_n^k} BestGenerate(S_n^j, W_n^k|GC_n)$: ▷ Normal best generation - see (6.30).
if $RepairLinked(S_{n-1},S_x \dots S_k)$ **then** ▷ Possibly incomplete substitution or delete repair.
  $diff = (S_{n-1}.\text{cont} \boxed{\wedge} S_n^j.\text{cont}) - S_n^j\text{cont}$ ▷ RT difference to reparandum end IU.
  if not $Revoke(Speaker, diff) \in S_{n-1}$.ctxt **then** ▷ New revoke information.
    $S_n^j = S_n^j \boxed{\wedge}$ [ctxt = $append(S_n^j.ctxt, Revoke(Speaker, diff))$] ▷ Update revoke act.
    $RepairLink(S_n^j,S_k)$ ▷ Repair link to reparandum-final ParseIU.
$add(W_n^k)$ to WordGraph; $SameLevelLink(W_n^k,W_{n-1})$; ▷ Add word for synthesis.
$GroundedInLink(W_n^j,S_n^j)$; **Return** ▷ Opposite to NLU, word grounded in ParseIU.

(5) Repair Onset:
$SubLex = \{\}$; **try do (1)-(2)** ▷ Repopulate the sublexicon. Check if extension or covert repair possible.
if Parse in *SubLex* is successful **then** ▷ Extension is possible.
  if address($\diamondsuit$) in $S_{n-1}.tree$ contains ?Ty **then** ▷ Inside an incomplete tree? Repeat.
    $add(S_n)$ where $S_n = S_{n-1} \wedge$ [ctxt = $append(S_{n-1}.ctxt, FwdProblem(Speaker, \text{cont}))$] ▷ Copy edge.
    $add(W_n)$ where $W_n = W_{n-1}$; $GroundedInLink(W_n,S_n - 1)$ ▷ Repeat; one word default.
    $SameLevelLink(S_n,S_{n-2})$; $SameLevelLink(W_n,W_{n-1})$;$RepairLink(S_n,S_{n-1})$ ▷ Order appropriately.
    $n = n+1$; **GoTo (2)** for S_n ▷ Increment as added an IU and return to normal generation.
for $i = n-1 \rightarrow 0$ **do** ▷ Backtrack along the ParseGraph.
  if $i < n-1$ **then** ▷ If longer than one word backtrack generate edit term.
    $add(S_n^j)$ to ParseGraph where $S_n^j = S_{n-1}$ ▷ Just copy, do not re-parse.
    $S_n^j = S_n^j \wedge \left[\begin{array}{ll} \text{actions} & = [] \\ \text{ctxt} & = append(S_n^j.ctxt, FwdProblem(Speaker, \text{cont})) \end{array}\right]$ ▷ Fwd act. No actions added.
    $add(W_n)$ to WordGraph where $W_n = random(EditTerms)$ ▷ Generate random edit term.
    $SameLevelLink(S_n^j,S_{n-1})$; $GroundedInLink(W_n,S_n^j)$; $Repair(S_n^j)$
    $n = n+1$ ▷ Added new IUs so increment.
  GoTo (1) for S_{i-1} ▷ Substitution or delete. Re-generate from this point.
  if successful generation of W_n, S_n^j from S_{i-1} **then**
    $diff = (S_{n-1}.\text{cont} \boxed{\wedge} S_n^j.\text{cont}) - S_n^j.\text{cont}$ ▷ RT diff. to reparandum end IU.
    $S_n^j = S_n^j \boxed{\wedge}$ [ctxt = $append(S_n^j.ctxt, Revoke(Speaker, diff))$] ▷ Revoke act.
    $RepairLink(S_n^j, S_x)$ for all edges $S_x \in S_i \dots S_{n-1}$; **Return** ▷ Repair link to whole reparandum.
  else Continue ▷ Keep backtracking if no word generated.

both cases the first type matched generation path is found ≈14 times faster with a lexicon of 210 words when using the pre-verbal lexicalisation method compared to the traditional DS model baseline. In terms of scalability, for bigger lexicon sizes $|\ lex\ |$, the increase in runtime for longer utterances will be a factor of $n\times\ |\ lex\ |$ in the case of traditional DS, whereas the factor will be much smaller for the new model as $|\ SubLex\ |$ is constrained by the grammar not to grow in the same order of magnitude as the size of the overall lexicon.

In generating repaired utterances, due to the storing of parse states in the ParseIU network, the extra runtime complexity caused by backwards searching is negligible– one would suspect that the complexity may not be any worse than for ambiguous fluent parsing with multiple possible DAG extensions, though this is yet to be fully evaluated. Development of this capability into NLU and the integration of these into a working dialogue system is on-going.

The more interesting and major complexity factor in DS parsing and generation that should be considered is the branching factor $b, as constraining this will reduce the blow-up in generation time for longer utterances. It becomes clear that the DS generation DAG may be constrained by using a number of different strategies as described by Sato (2011).

Although the best parse in both modes is currently characterized as choosing the sequence that jointly constructs the most complete tree and also builds the RT structurally closest to a domain concept (or the goal concept in generation), using real-valued parsing probabilities induced over a corpus would make for a more robust system.

6.8 Discussion

The NLU and NLG models proposed here meet the criteria for incrementality set out in Section 3.5.3 and give coverage to the types of repairs observed in corpora from Chapter 4 and in the literature (Chapter 2). They also begin to unify several existing dialogue approaches in an interesting way.

In terms of interaction, as per (Clark, 1996), the mechanisms have been designed such that the overall cost is minimized for the interaction for speaker and hearer of the self-repair. Furthermore, the information of the repair is available both to the generator and interpreter of the repair, in line with the idea of a public dialogue record or game board (Ginzburg, 2012). I have shown how it is possible to get a plausible dialogue semantics for repair in the spirit of Ginzburg et al. (2014)'s account but through well-defined automatic incremental processes.

In terms of the processing models described, the backwards search protocol in both modes is consistent with Shriberg and Stolcke (1998) and Chapter 4's empirical observations that the probability of retracing N words back in an utterance is more likely than retracing from N+1 words back, making the repair as local as possible.

In generation, in contrast to Skantze and Hjalmarsson (2010)'s string-based `SpeechPlan` comparison approach, there is no need to regenerate a fully-formed string from a revised goal concept and compare it with the string generated thus far to characterize the repair. Instead, repair here is driven by attempting to extend existing parse paths to construct the new target record type, backtracking through the parse state in an attempt to find suitable departure points for restarting generation, *retaining* the semantic representation already built up in the generation process.

6.8.1 Relationship to other work

The models here can be seen as part of a bigger plan to incorporate a notion of dialogue context (such as in KoS (Ginzburg, 2012)) within strongly incremental parsing and generation as afforded by DS-TTR and the IU-framework. Here I only incorporate a limited set of dialogue moves which interact incrementally with the context built up by the incremental semantic construction of DS-TTR, but this could be extended to a bigger set, possibly using more thorough-going models of discourse updates such as proposed in Asher and Lascarides (2003) to specify an ontology of contextual updates for dialogue. Furthermore, the context needs to incorporate the idea of dialogue grounding (Clark, 1996) which has been modelled extensively in KoS. This may require enriching the simple context model here with FACTS, Questions Under Discussion (QUD) and PENDING fields (see Section 3.3.2) to model the information states of dialogue participants more thoroughly, while adhering to the strict incrementality and interaction with the grammar I have proposed here.

The closest effort to the integration of incremental semantic construction and incremental dialogue state update in the literature is the PTT model (Poesio and Traum, 1997) which was used to model compound contributions by Poesio and Rieser (2010). PTT takes a grammar-based approach which incorporates syntactic, semantic and pragmatic information via a lexicalised TAG grammar paired with the PTT model for incremental dialogue interpretation. A full account of the incremental interpretation process is provided which incorporates lexical, syntactic and semantic information, meeting the criteria of incremental interpretation and representation de-

scribed in this thesis. Beyond this, Poesio and Rieser (2010) provide a detailed account of how a PTT dialogue model might generate a collaborative completion (a completion of a user's utterance) using inferential processes and the recognition of plans in a micro-domain: by matching the partial representation at a speaker transition point against a repository of known plans in the relevant domain, an agent can determine the components of these plans which have not yet been made explicit, and then use this to complete the plan for the current speaker.

While PTT shares desirable incremental qualities in terms of the recording *what* information is built up incrementally, it lacks a model that records *how* the information was constructed. The *how* part of the dialogue context, which is essential for a model of self-repair, is achieved above through incremental construction of time-linear graphical models of context and the modelling of the dependencies between them, where the structure of these graphs is preserved, at least utterance-internally for the exact increment each word contributes to the context. Treating actions and contextual updates as first-class citizens of the dialogue model not only allows self-repair processing to be modelled formally, but also allows it to be incorporated into robust dialogue systems.

6.8.2 Future work

In terms of future development, once fully implemented, a considerable challenge is the real-time evaluation of the modules in terms of interaction. Despite the costliness of the approach, there has been an encouraging move towards using interaction with human users to evaluate dialogue systems, particularly the criteria used by Skantze and Hjalmarsson (2010) of *human-likeness*, *politeness*, *efficiency*, *intelligence*, *speed of response*, *quality of feedback* and *clarity of turn taking*. To isolate the effects of the `DyLan` modules, these would have to be interchanged with a competing module such as the original Jindigo generator (Skantze and Hjalmarsson, 2010) in the same domain to have any between-systems comparison. To isolate the model's usefulness along interactive measures, a carefully controlled experimental set-up would also have to be devised.

In terms of processing, incremental repair works in a unified way across parsing and generation in a best-first manner according to DS tree completeness and similarity of record types with those in the domain concepts. However this is only a first approximation to best-first search with a reaction to low grammaticality and pragmatic infelicity– different search algorithms using different heuristics can now be experimented with.

Furthermore, this chapter has presented no model of the world for the conceptualiser as it

simply matches types to those in its domain concepts, and so this is fairly representational or syntactic in nature. The next chapter deals with how dialogue agents make use of situational context in self-repair processing within a probabilistic semantic framework.

While I cannot comment much on TTR's suitability for addressing the *logical form equivalence problem* (Shieber, 1993, Section 3.2), as this is beyond the scope of this thesis, it is worth mentioning that in the above explanation of DS-TTR generation above I hope it becomes evident that RT goal concepts are suitable inputs to (tactical) generation, at least in a practical sense for small domains. Due to the flexibility of TTR and the affordance it gives of designing type systems for a given domain, it is possible to constrain the concept types that are permitted as inputs, whilst having the ability to encode varying types of semantic information in the same object (record or RT). It is still not clear that First Order Logic is as well suited for inputs to generation, despite its better studied logical properties, as discussed in (Shieber, 1993, Section 3.2). Further work needs to be done here to investigate other suitable candidates.[10]

6.9 Conclusion

This chapter has presented an incremental semantics account of self-repair processing within the `DyLan` framework using the IU framework and DS-TTR parsing and generation techniques. Several non-trivial extensions to these tools were required before this was possible, particularly strong incrementality for DS-TTR parsing. I have described how plausible representations can be interpreted and generated on a word-by-word basis for the self-repair types observed in real data. The next chapter will introduce a situation-based semantics for dialogue and the position of self-repair processing within this.

[10]Thanks to my examiners Nick Asher and Wilfried Meyer-Viol for a lengthy discussion on this.

Chapter 7

Going Probabilistic: Modelling Conceptual Inference in Self-Repair Processing

This chapter[1] addresses the issue of incorporating probabilistic inference into the incremental dialogue framework presented in the previous chapter, presenting a method for modelling self-repair processing in a psycholinguistically plausible way. The model incorporates recent work on probabilistic TTR (Cooper et al., 2014), probabilistic incremental DS-TTR learning (Eshghi et al., 2013) and order-theoretic models of probability and information theory (Knuth, 2005) into a formal system that reflects the psycholinguistic results of Brennan and Schober (2001)'s experiments and deals with elliptical repair examples from Ginzburg et al. (2014), Levelt (1989) and Chapter 4 in a simple instruction-based reference identification domain.

7.1 Incorporation of probabilistic information

While the STIR system in Chapter 5 provides a reasonably robust statistically-driven automatic incremental self-repair detector, the dialogue framework presented in the previous chapter, while capable of the appropriate processing, is unlikely to scale up to using a large grammar and sizeable domain if limited to boolean parseability and semantic judgements. The most obvious pathway to improving robustness is to generalize the boolean values to probability values. This step should also allow a dialogue model to entertain distributions over judgements of a situation rather than just allow categorical ones and also allow Bayesian inference, an uncontroversial view for

[1] Much of the work here is drawn from Hough and Purver (2014a) and Hough and Purver (2014b).

modern cognitive models (see Chater et al., 2006, for overview).

One conceptually simple solution to integrating probabilistic information would be to incorporate STIR as a pre-parsing step that operates on DyLan's word hypothesis graph, assigning a real-valued probability to each repair hypothesis it makes, with the parseability of the word sequences cleaned of reparanda being used as a further classification step to accept or reject hypotheses. This is an integration step that may not be very interesting, as the probability of repair would be fully contingent on STIR's output, and therefore probabilistic inference is not used except for ranking repair hypotheses based on n-gram language model measures. Furthermore, the approach does not lend itself to inter-changeability between interpretation and generation, one of the desiderata of incrementality set out in Section 3.5.3.

In practical terms, to bring the coverage of DyLan's DS-TTR parser and generator up to that of the n-gram models trained on Switchboard would require large-scale grammar induction. While this would be more straightforward for parsers trained on PTB-style phrase-structure tree or dependency graph treebanks for which Switchboard resources are available (Rasooli and Tetreault, 2014; Honnibal and Johnson, 2014) making this step for a semantically-oriented framework like DS-TTR parsing is beyond the scope of this thesis.[2] Furthermore, it is not clear that the semantic representations for all types of self-repair in Switchboard could be obtained without large-scale manual annotation, which in itself is problematic– see Chapter 4.

Furthermore, in terms of a cognitive model, a simple pipeline approach fails to provide a communicative explanation as to how hearers use information from self-repairs in an interactive setting nor one for how and why speakers produce self-repairs. Consequently, instead of attempting wide coverage, this chapter focuses on modelling a micro-domain where the model is designed to explain psycholinguistic results. As with the last chapter, the integrated self-repair mechanisms operate within DyLan's NLU and NLG algorithms, however here they incorporate probabilistic and information-theoretic information. Furthermore, as Chapter 6 gives no account of the higher levels of inference in terms of model-theoretic semantics or pragmatic or discourse-level processing, this is addressed by proposing a rudimentary reasoning system within a simple dialogue manager that tightly interacts with NLG and NLU. The reasoning system on the interpretation side is not only capable of making inference incrementally word-by-word from fluent utterances but is also able to function with the same degree of incremental interpretation in

[2] Though see Eshghi et al. (2013) for methods for DS-TTR grammar induction.

self-repaired utterances, making use of the material available in reparanda for fast inference. In generation, it makes use of an information-theoretic relevance measure hitherto not presented in computational linguistics, to guide content selection in a word-by-word fashion. I show how the model can be used to model both roles of instructor and instructee in a reference identification task similar to those described in Brennan and Schober (2001) and Levelt (1989).

7.2 Background on modelling referential communication

There has been significant work on simple referential communication games in psycholinguistics, computational and formal modelling. These reference games are usually posed as a human-human situation where an instruction-giver and instructee have access to the same visual scene with simple objects, and the instructor produces an utterance to make the instructee select the object(s) he or she refers to. From the production perspective, Levelt (1989)'s seminal work modelled speaker strategies for producing referring expressions (REs) in such a simple object naming game. He showed how speakers use informationally redundant features of the target object, or *over-specification*, violating Grice's Maxim of Quantity that speakers should say no more than is necessary to convey their communicative intention (Grice, 1975), a result that has been supported in subsequent accounts.

In the NLG community, referring expression generation (REG) has been widely studied (see (Krahmer and Van Deemter, 2012) for a comprehensive survey). The incremental algorithm (IA) (Dale and Reiter, 1995), the most well-known REG algorithm, is an iterative feature selection procedure that operates on the properties of objects in the domain. The IA computes the distractor set of referents that each property used in a RE could cause to be inferred and from this gives a value to properties based on ability to determine the referent, however it does this in a non-greedy manner, as it iterates over the properties in order of their general discriminatory power (salience), not just their ability to determine the referent uniquely, and adds them if they have any discriminatory power, stopping when the RE determines the referent unambiguously. This was designed to be consistent with overspecification phenomena in that more salient properties will be selected if they have any discrimination ability, even if the final RE generated is not optimally brief. More recently Frank and Goodman (2012) investigate salience empirically in a Bayesian model of REG based on information-theoretic surprisal in terms of how much REs reduce uncertainty about their intended referent, a measure which they claim correlates strongly

to human judgements of which RE best describes a given target (carried out in a multiple choice study rather than allowing open answers).

The element of reference identification tasks discussed here, in line with the motivation of this thesis, is incremental processing. The work closest to the model presented here is the incremental REG models described by Guhe (2007) and Fernández (2013) and the model of incremental reference resolution in NLU proposed by Kennington and Schlangen (2014). Guhe (2007)'s approach to REG is to model a fine-grained incremental conceptualiser which passes increments to an incremental syntactic formulator as discussed in Section 3.2.2. Fernández (2013), on the other hand, takes a more purely linguistic, syntax-level perspective. Fernández sketches a novel solution to modelling overspecification, arguing the phenomenon may be caused more by the affordances of incremental left-right word-by-word information processing in different languages, rather than salience of properties as proposed by Dale and Reiter (1995)'s IA– it is worth noting here that the IA deals with incrementality in property selection of the avoidance of re-computation type, rather than word-by-word surface-level incrementality. She argues properties seemingly redundant when considering the RE as a whole unit may in fact be important when considering their incremental word-by-word contribution to reference resolution, that is, their *incremental informativity*. The paper gives cross-linguistic evidence from Spanish speakers based on Rubio-Fernández (2011)'s experimental results, arguing that speakers less frequently over-generate in languages and situations in which redundant post-nominal adjectives do not add any incremental information, and where doing so is syntactically felicitous. She exemplifies a domain where the only red lamp in a scene is the referent, and it is possible to individuate it from its object type (i.e. the property that it is a lamp) alone, where describing its colour is redundant, however still partially discriminative as there is another red object in the domain. For an English speaker "the red lamp" would be a typical overspecified description, whereas in Spanish "la lámpara roja" ('the red lamp') would be less common, and "la lámpara" would be a more common RE. This cross-linguistic difference is due to the fact the post-nominal "roja" does not add any reference information incrementally after "la lámpara", which on its own has sufficient discriminatory power, while the English "red", although not uniquely determining the referent, narrows the reference set and so is incrementally informative. Fernández uses this example to sketch a REG system that uses a variant of Dale and Reiter (1995)'s IA for content selection interleaved with a TAG-based grammar formulator that is strictly left-right incremental in its

tree construction. She emphasizes the importance of tightly coupling the REG procedure with a reference resolution/NLU component but does not give details of how this could be done.

On the interpretation side of reference, reference resolution, Kennington and Schlangen (2014)'s incremental NLU system models the role of the hearer or instructee. The system continuously incrementally outputs a distribution of possible referents conditioning on the logical values of properties of the objects in the scene and the words used to refer to those properties spoken by the instructor. The model forms part of *situated* dialogue processing, as it continuously updates its referent distributions based on perceptual data, and not necessarily just linguistic data. The conditional probabilities are calculated using a generative model (of the speaker) and implemented using Markov Logic networks. Kennington et al. (2014)'s development of the model uses incremental semantic representations built up word-by-word by an incremental rMRS (robust Minimal Recursion Semantics) parser (Peldszus et al., 2012) as part of the property set it conditions on, boosting results from using simple n-gram models. No model of generation is given.

7.3 Towards a dialogue-oriented incremental account

Motivated by incremental approaches such as those just described, this chapter presents an incremental dialogue-motivated account of reference identification which models the speaker in terms of incremental NLG and dialogue management and the hearer in terms of incremental NLU and dialogue management in reference identification games.

More specifically, the model aims to reflect the evidence from Brennan and Schober (2001)'s experiments that people reason at an incredibly time-critical level from linguistic information and the evidence that self-repair can speed up semantic processing (or at least reference identification) in such games. An incorrect RE being partly vocalized and then repaired in the instructions in conjunction with a filled pause interregnum (e.g. "the [yell-, + { uh, } purple] square") yields quicker response times to select the correct object from the onset of the target ("purple") than in the case of the fluent instructions ("the purple square"), with no significant effect on accuracy – see Section 2.4.3 for details. Furthermore, the account should model non-local repair processing of instructions such as "From yellow down to brown – no – that's red." (Levelt, 1989, via Ginzburg et al. (2014)) or here using a syntactically simpler but illustrative example "the [yellow square, +{ no }, purple]". An account of this last example will require a dialogue-level

account of ellipsis, in addition to the apparatus for elliptical repair processing described in the last chapter. More generally, the model should incorporate the interaction between incremental linguistic processing and higher-level inference as Fernández (2013)'s paper attempted to do.

Furthermore, the reasoning system should be useful for interpreting and generating *other*-repair, such as clarification requests, with minimal modification, which fulfils one of the desiderata of self-repair processing that Ginzburg et al. (2014) propose. This is achieved by modelling dialogue processing in terms of incremental type judgements of the dialogue state by the participants, each of which is characterized as an answer to a question about the type-judgement of the current state– it is therefore possible to clarify anything that has been asserted on a word-by-word basis in terms of the type judgements built up in the discourse. The nature of the set of assertions and the set of questions will be explained below, and computationally tractable methods for generating these sets for different situations is explained.

Illustrative examples will be addressed in Section 7.6. First I will set out the framework in which it is possible to model such processing, which extends that of Chapter 6 to include probability judgements.

7.4 Background on technical tools

7.4.1 Probabilistic TTR

While classical type theory has been the predominant mathematical framework in natural language semantics for many years (Montague, 1974, inter alia), it is only recently that probabilistic type theory has been discussed for this purpose. Similarly, type-theoretic representations have been used within dialogue models (Ginzburg, 2012); and probabilistic modelling is common in dialogue systems (Williams and Young, 2007, inter alia), but combinations of the two remain scarce. In this chapter this connection is made, taking Cooper et al. (2014)'s probabilistic TTR as the principal point of departure for modelling incremental inference in dialogue as described above.

At the time of writing there had been no methods for practical integration of probabilistic type-theoretic inference into a dialogue system; here I discuss computationally efficient methods for implementation. I argue for their efficacy in simple referential communication domains, but simultaneously suggest the methods could be extended to larger domains and additionally used for real-time learning in future work.

Given that TTR, introduced in the previous chapter, has a highly flexible rich type system, variants have been considered with type judgements corresponding to real-number-valued perceptual data used in conjunction with linguistic context, such as those representing visual information (Larsson, 2011; Dobnik et al., 2013), demonstrating its potential for situated, embodied and multi-modal dialogue systems. The possibility of integration of perceptron learning (Larsson, 2011) and Naive Bayes learning (Cooper et al., 2014) into TTR show how linguistic processing and probabilistic conceptual inference can be treated in a uniform way within the same formal system.

Probabilistic TTR as described by Cooper et al. (2014) replaces the categorical $s : T$ judgement, the judgement that it is *true* or *false* that an object s is of type T, with the real number valued $p(s : T) = v$ where $v \in [0,1]$.[3] The authors show how standard probability theoretic and Bayesian equations can be applied to type judgements and how an agent might learn from experience in a simple classification game. In their example, the agent is presented with instances of a situation with associated type judgements and it learns with each round by updating its set of probabilistic type judgements to best predict the type of object in focus – in this case updating the probability judgement that something is an apple given its observed colour and shape, i.e. $p(s : T_{apple} \mid s : T_{Shp}, s : T_{Col})$ where $Shp \in \{shp1, shp2\}$ and $Col \in \{col1, col2\}$. From a cognitive modelling perspective, these judgements can be viewed as learning concepts from probabilistic perceptual information, and if framed as a language acquisition scenario these concepts could be associated with words. I use similar methods in the toy reference domain below, but show how complex prior type judgements could be constructed efficiently, and how conditional probabilistic judgements can be made incrementally without exhaustive iteration through individual type classifiers, as was the mechanism implicit in Cooper et al. (2014) and Kennington and Schlangen (2014)'s models.

Before technically introducing probabilistic TTR, I do not re-iterate the elements of TTR introduced in the last chapter but prime the reader that the account below will make use of the definitions for the record type check (6.8), subtype relation check (6.9) and merge operation (6.10). Here I also define a dual of the merge operation, not found in the TTR literature, which is

[3] Several people I have discussed this with are not convinced a type judgement can be probabilistic. I remain agnostic to the plausibility of a non-conditional judgement such as this one being real-valued, however I do think real-valued *conditional* probability judgements are realistic. I thank David Schlangen and Arash Eshghi for discussions on this. The view I set out below can be cashed out purely in terms of conditional type judgements, however the conditional judgement may at times be notationally suppressed where appropriate and in a consistent manner– these cases will be noted where they crop up.

necessary for the analysis below: the common minimal supertype operator $\veebar$. While technically the common minimal supertype of R_1 and R_2 is the *join type* $R_1 \vee R_2$, here, for reasons that will become apparent below in the discussion on type lattices, we are also interested in isolating the minimally common supertype of two RTs R_1 and R_2 which is still a (non-disjunctive) RT, which, when there are no clashing type judgements, amounts to field intersection as below in (7.1). Note the minimally common supertype RT of multiple RTs is generally *not* equivalent to their join type as will be explained.[4]

$$\text{if } R_1 = \left[\begin{array}{l} l_1 : T_1 \\ l_2 : T_2 \end{array}\right] \text{ and } R_2 = \left[\begin{array}{l} l_2 : T_2 \\ l_3 : T_3 \end{array}\right] \tag{7.1}$$
$$R_1 \veebar R_2 = \left[\begin{array}{l} l_2 : T_2 \end{array}\right]$$

For exposition of probabilistic TTR, I repeat some of Cooper et al.'s calculations and show some equivalences not described by the authors. I demonstrate efficient order-theoretic and graphical methods for generating and incrementally retrieving probabilities in Section 7.5.

Cooper et al. (2014), under the assumption that type judgements can be real-valued, define conditional probability of an object being of type R_2 given it is of type R_1 as in (7.2).[5] This is the most important judgement in probabilistic TTR, due to the framework's motivation: an agent can judge a situation s is of a given situation type, given the evidence that it is of other situation types. In this way an agent is positioned as a classifier of situations given the evidence available to it. Here I assume s can be a record, not just a basic type, and so R_1 and R_2 can be record types.

$$p(s : R_2 \mid s : R_1) = \frac{p(s : R_1 \wedge s : R_2)}{p(s : R_1)} \tag{7.2}$$

Given classical probability theoretic equivalences, they define the probability of a situation being of a meet (conjunctive) and join (disjunctive) types of two basic types or RTs in terms of the standard *product* and *sum* rules in probability theory:

$$\begin{aligned} p(s : R_1 \wedge R_2) &= p(s : R_1)p(s : R_2 \mid s : R_1) \\ p(s : R_1 \vee R_2) &= p(s : R_1) + p(s : R_2) - p(s : R_1 \wedge R_2) \end{aligned} \tag{7.3}$$

[4]Here the examples avoid label-type clashes between two RTs (i.e. where R_1 contains $l_1 : T1$ and R_2 contains $l_1 : T2$); in these cases the operations are more complex than field intersection, but the precise definition of how to deal with these cases is not necessary here.

[5]This equation is not included directly in Cooper et al. (2014) as it is suppressed due to space constraints, but was included in the original unpublished manuscript and is consistent with the other equations.

It is practically useful, as I will describe below, that the join probability can be computed in terms of the meet. Given the classical probability theoretic definitions for the meet and the join type they show it is possible to sustain the below:

$$\begin{aligned} p(s:R_1 \wedge R_2) &\leq p(s:R_1) \qquad p(s:R_1 \wedge R_2) \leq p(s:R_2) \\ p(s:R_1) &\leq p(s:R_1 \vee R_2) \qquad p(s:R_2) \leq p(s:R_1 \vee R_2) \end{aligned} \tag{7.4}$$

Also, there are equivalences between meets, joins and subtypes in terms of type judgements as described above, in that assuming if $R_1 \sqsubseteq R_2$ then $p(s:R_2 \mid s:R_1) = 1$, we have:

$$\begin{aligned} &\text{if } R_1 \sqsubseteq R_2 \\ p(s:R_1 \wedge R_2) &= p(s:R_1) \\ p(s:R_1 \vee R_2) &= p(s:R_2) \\ p(s:R_1) &\leq p(s:R_2) \end{aligned} \tag{7.5}$$

I return to an explanation for these classical probability equations holding within probabilistic TTR in Section 7.5. I make a remark here that the meet type probability is the same as the probability of the merge type's probability in (7.6), due to the *extensional* equivalence of a (non record type) meet type $R_1 \wedge R_2$ and the resulting record type from the operation $R_1 \mathbin{\mathbin{\wedge}\!\!\!\!\wedge} R_2$: that is to say no object can be judged to be of type $R_1 \wedge R_2$ without being of type $R_1 \mathbin{\mathbin{\wedge}\!\!\!\!\wedge} R_2$ and vice-versa, despite them being intensionally distinct types. For now, assume all the $\wedge$ conjunctions in the above equations can be replaced by $\mathbin{\mathbin{\wedge}\!\!\!\!\wedge}$ and the equations will still hold. The same is not true of the relationship between the join $\vee$ type and the $\mathbin{\mathbin{\vee}\!\!\!\!\vee}$ operation as I will explain.

$$p(s:R_1 \wedge s:R_2) = p(s:R_1 \wedge R_2) = p(s:R_1 \mathbin{\mathbin{\wedge}\!\!\!\!\wedge} R_2) \tag{7.6}$$

7.4.2 Record Type lattices

To support efficient inference in `DyLan`, I represent dialogue domain concepts as partially ordered sets (*posets*) of RT judgements. This follows insights used in inducing DS-TTR actions from target RTs (Eshghi et al., 2013), however here this idea is fleshed out in a formal way to provide an interface to a reasoning system. A poset has several advantages over an unordered list of un-decomposed record types: the possibility of incremental type checking; increased speed of

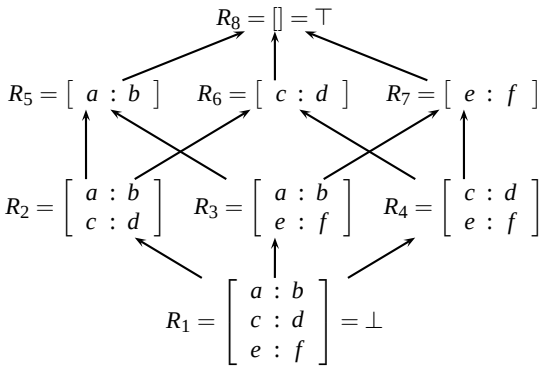


Figure 7.1: Record Type lattice ordered by the subtype relation

type checking, particularly for pairs of or multiple type judgements; immediate use of type judgements to guide system decisions; inference from negation; efficient construction of a question-under-discussion (QUD) structure that includes real number relevance values; and the inclusion of learning within a domain. I leave the final challenge for future work, but discuss the others here.

It is possible to construct a poset of type judgements for any single RT by decomposing it into its constituent supertype judgements in a *record type lattice*. As per set-theoretic lattices, this can be visualised as a Hasse diagram such as Figure 7.1, however here the ordering arrows show $\sqsubseteq$ ('is a subtype of') relations from descendant to ancestor nodes, rather than the normal set inclusion relation.

To characterize a RT lattice L ordered by $\sqsubseteq$, I adapt Knuth (2005)'s description of lattices in line with standard order theory. A RT lattice is a partially ordered set of RTs closed under the *meet* and *join* operations, whereby all pairs of elements have a unique element that is their meet and a unique one that is their join. This is to say, for a pair of RT elements R_x and R_y, their lower bound is the set of all $R_z \in L$ such that $R_z \sqsubseteq R_x$ and $R_z \sqsubseteq R_y$, and in the event that a unique greatest lower bound exists in L between two elements R_x and R_y, this is the meet. The meet is in fact extensionally equivalent to the meet type $R_x \wedge R_y$ in TTR, and, given Remark (7.6), is also extensionally equivalent to the RT resulting from $R_x \barwedge R_y$. Dually, if the unique least upper bound exists for R_x and R_y this is their join in L and in TTR terms is the result of $R_x \barvee R_y$ but not necessarily extensionally equivalent to the join type $R_x \vee R_y$– this is due to the fact that the result of $R_x \barvee R_y$ may be extensionally equivalent to the minimal common supertype of other pairs of RTs in L (and consequently may be the type of different objects or records which are not of type R_x or R_y), so $R_x \barvee R_y$ can be a more general type than the disjunctive type $R_x \vee R_y$. The decision not to include disjunctive and conjunctive types directly on L, only using RTs and operations that

yield new RTs, is motivated by limiting the size (and therefore complexity) of the lattice, and also by keeping consistency in the type hierarchy: the limitation of the lattice to types that are of record type, rather than of disjunctive or conjunctive (record) type means this is a record type lattice. So, while the extensionally equivalent RTs for meet types are included in L, elements representing join types are not, and the join and meet operations under which the lattice is closed are $\veebar$ and $\barwedge$.

I now introduce other relevant terminology. One element *covers* another if it is a direct successor to it in the subtype hierarchy. L has a greatest element ($\top$) and least element ($\bot$), with the *atoms* being the elements that cover $\bot$; in Figure 7.1 if R_1 is viewed as $\bot$, the atoms are R_2, R_3 and R_4. *Join-irreducible* elements are those which cannot be expressed as the join of two other elements– in the case of Figure 7.1 the only join-irreducibles are the atoms and $\bot$, but that need not be the case in other distributed lattices as will be shown.

Given the definitions for the meet and join operations as $\veebar$ and $\barwedge$, a RT lattice L ordered by the subtype relation obeys the following rules for any three elements x, y and z in L:

$$x \veebar x = x;\ x \barwedge x = x \qquad \text{(L1. Idempotency)}$$

$$x \veebar y = y \veebar x;\ x \barwedge y = y \barwedge x \qquad \text{(L2. Commutativity)}$$

$$x \veebar (y \veebar z) = (x \veebar y) \veebar z;\ x \barwedge (y \barwedge z) = (x \barwedge y) \barwedge z \qquad \text{(L3. Associativity)}$$

$$x \veebar (x \barwedge y) = x \barwedge (x \veebar y) = x \qquad \text{(L4. Absorption)}$$

RT lattices ordered by the subtype relation are distributive lattices as they obey the two distributivity relations:

$$x \barwedge (y \veebar z) = (x \barwedge y) \veebar (x \barwedge z) \qquad \text{(D1. Distributivity of } \barwedge \text{ over } \veebar\text{)}$$

$$x \veebar (y \barwedge z) = (x \veebar y) \barwedge (x \veebar z) \qquad \text{(D2. Distributivity of } \veebar \text{ over } \barwedge\text{)}$$

Finally, a RT element R_x has a *complement* if there is a unique element $\neg R_x$ such that $R_x \veebar \neg R_x = \top$ and $R_x \barwedge \neg R_x = \bot$. The lattice in Figure 7.1 is *complemented* as this holds for every element, however as I will discuss RT lattices in general are distributive but not complemented.

Graphically, the join of two elements can be found by following the connecting edges upward until they first converge on a single RT, e.g. $R_2 \veebar R_4 = R_7$ in Figure 7.1, and the meet can be found

by following the lines downward until they connect to give the result of their merge operation, e.g. $R_2 \curlywedge R_4 = R_1$.

If we consider R_1 to be a domain concept in a dialogue system, it is graphically easy to see how its RT lattice L can be used for incremental inference in terms of a downward search from the initial underspecified $\top$. As incrementally specified RTs become available from the NLU module they are matched to those in L to determine how far down towards the final domain concept R_1 our current state allows us to be. In terms of linguistic processing, different sequences of words or utterances lead to different paths. Of course, any practical dialogue system must entertain more than one possible domain concept as an outcome, so the lattice construction methods of Eshghi et al. (2013) must be extended to deal with this. L must therefore contain multiple possible final concepts, constituting its atoms, each with several possible dialogue move sequences, which correspond to possible downward paths. The example domain I use here does indeed have multiple possible final outcomes, and I explain a general lattice construction method for any dialogue situation with the possibility of disjoint final states in Section 7.5.

7.4.3 Lattices for probabilistic and information-theoretic inference

To explain the incorporation of probabilities into RT lattices, it is necessary to draw on Knuth (2005)'s work on generalizing a Boolean algebra to the probability calculus through the use of real-valued inclusion measures on lattices. Knuth shows how a Boolean algebra of logical statements can be expressed as a *distributed complemented lattice* of propositions ordered by the implication ($\rightarrow$) relation, a lattice he calls the *assertion lattice* (see the left-hand lattice in Figure 7.2). Given the assertion lattice is distributed as it obeys the distributivity laws (as in D1 and D2, rules above, replacing $\curlywedge$ with $\wedge$ and $\curlyvee$ with $\vee$) and also complemented, this means the Boolean operators $\wedge$ and $\vee$ and $\neg$ happily coincide with the order-theoretic relations of meet, join and complement. Furthermore, the implication relation $\rightarrow$ which orders the lattice is the inverse of inclusion.

Dual to the assertion lattice is the *question lattice* (see the right-hand lattice of Figure 7.2), the lattice of *down-sets* (all the elements beneath a given element and itself) of the assertion lattice's elements. The question lattice includes all possible unions of down-sets of the assertion lattice's elements and the join-irreducible elements of the question lattice form a lattice isomorphic to the assertion lattice. This lattice is ordered by the subset inclusion relation $\subseteq$, with its meet being set intersection $\cap$ and its join set union $\cup$, and is distributive but not generally complemented.

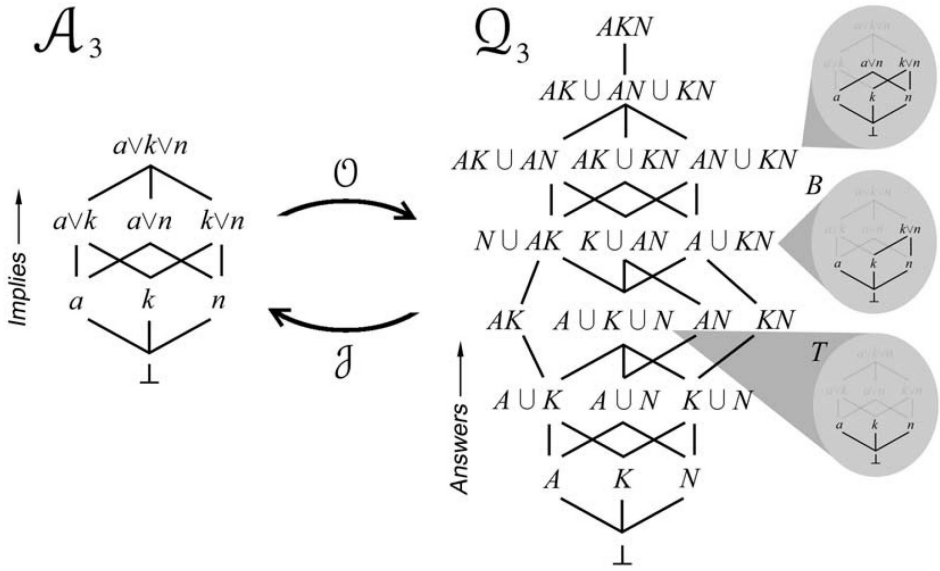


Figure 7.2: An assertion lattice of propositions A_3 and its dual, the question lattice Q_3 formed by the ordered down-sets of the elements of A_3 with some example down-sets of possible answers in the grey ovals, from Knuth (2005)

By way of example, a question such as $A \cup K \vee N$ ($\approx$ 'Is it the case that a or that $k \vee n$?'), as illustrated in the grey circle labelled B in Figure 7.2, has as its possible answers the union of the down-set of a and the down-set of $k \vee n$ on the assertion lattice.

Knuth assumes the classical semantic view of characterizing questions (semantic objects, not necessarily equivalent to linguistic interrogatives) as the set of their possible answers (Hamblin, 1973), a view which has been consequently revised in various popular accounts of questions, most notably by Groenendijk and Stokhof (1984). I will not extend Knuth's characterization much here, as the focus is more on incorporating probability and information-theoretic notions into semantics rather than providing a detailed account of questions.

Knuth's *Inquiry Calculus* extends Boolean algebra to the probability calculus by calculating probabilities in terms of the inclusion function $Z(x,y)$ for distributive lattices, here used to calculate the degree to which statement x includes statement y in the the assertion lattice:

$$p(x \mid y) = Z(x,y) = \begin{cases} 1 & \text{if } y \leq x \\ 0 & \text{if } x \wedge y = \perp \\ p & \text{otherwise, where } 0 \leq z \leq 1 \end{cases} \tag{7.7}$$

To make the generalization from Boolean algebra to probability theory clear, one can characterize the ordering relation on the assertion lattice as logical implication, making it possible to calculate the *degree* to which one assertion implies the other. Here the degree to which y implies x can be real valued as well as Boolean:

$$p(x \mid y) = Z(x,y) = \begin{cases} 1 & \text{if } y \to x \\ 0 & \text{if } x \wedge y = \bot \\ p & \text{otherwise, where } 0 \leq z \leq 1 \end{cases} \tag{7.8}$$

If (7.8) is viewed as a lazy evaluation function, if the first two cases do not apply, the third case p can be calculated by normal probability theoretic rules, that is the sum rule, product rule and Bayes theorem. All these calculations use the meet probability, which can be found by finding the meet on the lattice and reading off the numerical value of that element. The numerical values of meets and joins are all derivable from values initially assigned axiomatically to the join-irreducible elements (which in the case of Knuth's Boolean examples are the atoms) of the assertion lattice– all other probabilities can be calculated in terms of these by the standard probability equations. More detail on this will follow when explaining probability in RT lattices.

The question lattice requires consistency with the assertion lattice from which it was generated, however Knuth uses the degree of inclusion measure Z on the question lattice for a different reason other than logical implication, namely an information-theoretic characterization of *relevance*. That is the relevance of one question Q to another I, in other words the degree to which question I is answered by posing question Q, characterized analogously to the degree of implication on the assertion lattice, but with a different ordering relation (inclusion) and characterzation of the meet (intersection)– see (7.9).

$$d(I \mid Q) = Z(I,Q) = \begin{cases} c & \text{if } Q \subseteq I \ (Q \text{ answers } I) \\ 0 & \text{if } I \cap Q = \bot \ (Q \text{ and } I \text{ are exclusive}) \\ d & \text{otherwise, where } 0 \leq d \leq c \\ & \quad \text{where } c \text{ is the maximal relevance} \end{cases} \tag{7.9}$$

Knuth chooses the abbreviating letters here for illustrative purposes: I stands for 'issue'

and Q for 'query', as the most important relevance measure is the degree to which the query Q answers the issue I. In the case of the question lattice in Figure 7.2, the *central issue* is the question $K \cup A \cup N$ (labelled T) given the situation is posed as finding out which of the three atomic statements in the assertion lattice is true. The central issue is the least *real* question, that is the least question which can always be answered by a true statement– as I will show later for dialogue inference the real questions are the ones to be concerned with here. The most ambiguous real question (i.e. that with the most answers) is at the top of the question lattice and the most informative real question (i.e. that with the fewest answers) is the central issue $K \cup A \cup N$ at the bottom of the real sub-lattice of questions.

The first two cases of the relevance measure (7.9) make intuitive sense– if a question Q is included in I, as it is lower in the ordering, then it will certainly answer I, as it contains a subset of I's possible answers, and therefore guaranteed to answer it, or is maximally relevant to it. For the second case, where the possible answers to I and Q only have the absurdity $\bot$ in common, then Q is completely irrelevant to I. The third case, where the other two conditions do not apply, calculating relevance as the gradient degree of inclusion becomes more complicated, however this can be achieved using the same rules of inference as used for the assertion lattice, those of Knuth's Inquiry Calculus. Also, as was the case for the assertion lattice, the required inclusion values can be calculated by using the values of the join-irreducible elements, which are assigned based on the probability-weighted surprisal of the element in the assertion lattice which generated their answer set. So for any join-irreducible question X, its prior relevance $d(X|\top)$ is the probability-weighted surprisal of the prior probability of the greatest element x in the set of answers X on the assertion lattice, as in (7.10).

$$d(X|\top) = -p(x|\top) log_2 p(x|\top) \tag{7.10}$$

So by way of example from Figure 7.2, the prior relevance of question K, a join-irreducible element of the question lattice which has the simple answer set $\{k, \bot\}$ of elements on the assertion lattice is simply $-p(k|\top) log_2 p(k|\top)$. Given this assumption, the prior relevance of a partition question, which is a join of join-irreducible questions (an example being the central issue $K \cup A \cup N$ in Figure 7.2), by application of the sum rule is in fact the Shannon entropy of the probability distribution of the top elements of each of the disjunctive answer sets. For other

complex questions Knuth (2005) shows how the relevance of one complex question to another can actually be computed in terms of the ratio of degrees to which each question includes the $\top$ question– see (Knuth, 2005, p. 23, calculation (44)). So, for the complex questions B and T highlighted in Figure 7.2, the relevance of question B, which is $A \cup KN$, to the central issue T, which is $K \cup A \cup N$, can be calculated as in (7.11).[6]

$$d(T|B) = c\frac{d(B|\top)}{d(T|\top)} \tag{7.11}$$

where c is a chosen arbitrary normalisation constant for relevance, which can simply be 1 or the maximal relevance (Knuth, 2006). Given the information-theoretic characterization just described for partition questions, in this case this is in fact simply a ratio of the two questions' Shannon entropy values. Intuitively this is to say the relevance of Q to resolving I is the ratio between the amount of information expected to be gained by asking Q and the amount of information expected to be gained by asking I, given I is the central issue (least real question) in the lattice.

7.4.4 Probabilistic DS-TTR parsing judgements

One final technical expansion to Chapter 6 needs to be introduced here– that of assigning probability values to sequences of DS-TTR actions. This is due to a move towards more robust parsing and different parsing techniques. As ambiguity is rife in NLU, DS-TTR should not clean its hands of assigning likelihood to different parses. An example of this for senses of the English word 'is': one example action for the word derived from grammar induction in Eshghi et al. (2013) is shown in Figure (7.3).

Probabilistic DS-TTR parsing judgements can be incorporated into the ParseIU types proposed in the last chapter, consistent with the idea of probabilistic Austinian propositions proposed by Cooper et al. (2014). To achieve this the ParseIU RTs will have an extra field *prob* as the sum of the probabilities of the lexical actions run on the path normalised by the number

[6]While this is the formulation in Knuth (2005), there are conflicting accounts of this bi-valuation for complex questions, for example in Knuth (2006) equations (22) and (23). The formulation here does not obey the calculus described for lattices in general, in this case not being consistent with the Bayes' rule analogue, however entropy ratio is an intuitive way of considering relevance and the results it gives for natural language inference below seem intuitively correct. Greater questions than the central issue will be less or equally relevant to it rather than the reverse, so I suspect a clearer statement of relevance for complex questions needs to be made to resolve the contradictions in the literature.

'is': 0.07291666

IF	$?Ty(e \rightarrow t)$
THEN	$\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_0\rangle)$
	$\texttt{put}(?Ty(e))$
	$\texttt{go}(\langle\uparrow_0\rangle)$
	$\texttt{make}(\langle\downarrow_1\rangle);\texttt{go}(\langle\downarrow_1\rangle)$
	$\texttt{put}(Ty(e \rightarrow (e \rightarrow t)))$
	$\texttt{put}(Fo($
	$\lambda r1 : \left[\begin{array}{ll} head & : e \end{array} \right]$
	$\lambda r2 : \left[\begin{array}{ll} head & : e \end{array} \right].$
	$\left[\begin{array}{ll} x1_{=r1.head} & : e \\ x2_{=r2.head} & : e \\ e_{=eq} & : e_s \\ p1_{=subj(e,x2)} & : t \\ p2_{=obj(e,x1)} & : t \\ head_{=e} & : t \end{array} \right]))$
	$\texttt{put}(\langle\downarrow\rangle\bot)$
ELSE	ABORT

Figure 7.3: Probabilistic DS-TTR action for 'is' in the sense of an NP predicate identity: "this *is* a cat" learned from data from the induction technique in Eshghi et al. (2013)

of lexical actions– not the number of words, as re-run actions in ellipsis still count in terms of probability mass. This can be seen in the minimal extension of (6.18) to make a probabilistic ParseIU the type in (7.12).

(7.12) $$ParseIU = \left[\begin{array}{ll} \text{tree} & : DSTTRTree \\ \text{actions} & : list(DSTTRAction) \\ \text{cont} & : RecordType \\ \text{ctxt} & : list(Move) \\ \text{prob} & : \mathbb{R} \end{array} \right]$$

Given the new definition, an example record of type *ParseIU* after parsing "John" utterance-initially, given the probability of a given lexical entry for 'John', $John'^{lex}$, after initial introduction and prediction computational actions runs is 0.5, will therefore be the record in (7.13).

(7.13)
$$\left[\begin{array}{lcl}
\text{tree} & = & ?Ty(t), \left[\begin{array}{ll} x_{=john} & : e \\ e & : e_s \\ p_{=subj(e,x)} & : t \\ head_{=e} & : e_s \end{array}\right] \\
 & & \quad Ty(e), \left[\begin{array}{ll} x_{=john} & : e \\ head_{=x} & : e \end{array}\right] \qquad \Diamond, ?Ty(e \to t) \\
 & & \quad \lambda r : \left[\begin{array}{l} head : e \end{array}\right].r \boxed{\mathbb{A}} \left[\begin{array}{ll} x_{=r.head} & : e \\ e & : e_s \\ p_{=subj(e,x)} & : t \\ head_{=e} & : e_s \end{array}\right] \\
\text{actions} & = & [introduction, prediction, John^{lex}, thinning, completion, anticipation] \\
\text{cont} & = & \left[\begin{array}{ll} x_{=John} & : e \\ e & : e_s \\ p_{=subj(e,x)} & : t \end{array}\right] \\
\text{ctxt} & = & [Assert(User, \text{cont})] \\
\text{prob} & = & 0.5
\end{array}\right]$$

The characterization of the ParseIU graph stays the same as the last chapter, except now there is access to the probability of the sequences, and by simple calculation the probability contribution of each word; this brings the IU graphs closer to the DAGs used in STIR in Chapter 5. The notion of strong incremental representation and incremental interpretation now extends to probabilistic information.

This extra probabilistic information will be used to guide parsing and generation strategies. It is possible to characterize a best-first search where the most probable parse path will be the one whose path-final ParseIU has the highest *prob* value. In generation, which in DS-TTR is driven by parsing, a best-first search for a generation path of words will also be contingent on finding the most probable action sequences given the words chosen to express a goal concept. I will explain the integration of these into `DyLan`'s NLU and NLG algorithms below.

7.5 Probabilistic incremental inference for dialogue

With these tools at hand it becomes possible to model fine-grained incremental semantic inference in dialogue probabilistically and describe implementational methods for the `DyLan` dialogue system. I will describe how this is done for a dialogue micro-domain here, before explaining how these methods model Brennan and Schober (2001)'s experimental results within this domain in Section 7.6.

7.5.1 A familiar psycholinguistic experiment

This chapter models a simple reference identification task where an instructor produces utterances describing an object which an instructee comprehends and reacts to by selecting the object they think best fits the description as quickly as possible. The visual stimulus available to both parties is as in Figure 7.4.

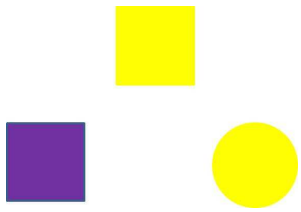


Figure 7.4: Visual scene for instructor and instructee in the reference identification game

In this game I characterize the referent set $referents$ of a purple square, yellow square and yellow circle as mutually exclusive referent situation types (record types), which can be labelled as *PSq*,*YSq* and *YC* respectively. On the interpretation side, the challenge is to predict the final reference situation type judgement $s : R$, that the situation s is of record type R, given currently available evidence in the form of current type judgements about the situation ($s : E$). So, as instructions are heard word-by-word the hearer tries to predict the maximally likely referent as in (7.14).

$$\arg\max_{R \in referents} p(s : R | s : E) \tag{7.14}$$

On the generation side, I model the incremental strategic level or conceptualisation stage as the selection of the semantic update to the evidence E which maximises the likelihood of the relevant goal concept (also a RT) which communicates the target referent. This goal may change incrementally during word-by-word surface realization and trigger repair realisation behaviour in the way described in the last chapter. In tandem with the semantic update selection task (incremental conceptualisation), I simultaneously model the surface realisation of referring expressions where the task is to select the next word that maximises the probability of that update being yielded by the parser after consuming that word. I will return to this below.

7.5.2 Probabilistic RT lattices to encode domain knowledge

Here I will explain how the domain of the simple reference game just described can be modelled in terms of a probabilistic RT lattice and explain the principal inference mechanisms for such lattices, which show some nice consistencies between Knuth (2005) and Cooper et al. (2014)'s probability equations.

I assume the hearer and speaker initially entertain three distinct reference situations, and these can be encoded as three distinct type judgements encoded simply as the atoms of a RT lattice as in Figure 7.5. The meet of any of these three situations is $\bot$ and has a probability of 0 due to its impossibility, constituting absurdity. I assume before the game has begun the atomic situations will all have equal probability ($\frac{1}{3}$), effecting a uniform distribution. With this said, the overall probability mass of the lattice $P(L)$ is a global denominator used to normalise all probability calculations in the lattice to ensure proper probability values so the initial assignment to the atoms need not in fact sum to one (Knuth, 2005).

If the reader has ontological doubt over the possibility of an unconditional type judgement being probabilistic, this can be avoided by assuming each atom's prior type judgement probability is in fact composed of a set of type judgements that the situation is of this type that are non-probabilistic– they are simply true judgements. Given this assumption, we only need to use the cardinality of these type judgement sets, notated as e.g. $\|YSq\|$ for the number of judgements that situations are those that include a yellow square. If one assumes a uniform prior distribution, each atomic prior judgement set can simply be assigned a cardinality of 1, which means in a lattice with three atoms, given we normalise by the global denominator $P(L)$ which will be 3, the prior probability of each one will be $\frac{1}{3}$ conditionally on $s:\top$. While not the focus of this chapter, characterizing each atom as a judgement set is useful in a learning game in the spirit of Cooper et al. (2014): an agent learns from the experience of making and storing type judgements.

While I abbreviate below, technically, probabilistic inference on distributive lattices is always conditional. From here on, assume the standard probabilistic type judgement $p(s:R_1)$ stands for $p(s:R_1 \mid s:\top)$, the meet probability judgement $p(s:R_1 \mathbin{\barwedge} R_2)$ stands for $p(s:R_1 \mathbin{\barwedge} R_2 \mid s:\top)$ which can be calculated as $p(s:R_1 \mid s:\top)p(s:R_2 \mid s:R_1 \mathbin{\barwedge} \top)$, the join probability $p(s:R_1 \mathbin{\barvee} R_2)$ stands for $p(s:R_1 \mathbin{\barvee} R_2 \mid s:\top)$ which can be calculated as $p(s:R_1 \mid s:\top)+p(s:R_2 \mid s:\top)-p(s:R_1 \mathbin{\barwedge} R_2 \mid s:\top)$, all of which are consistent with (Knuth, 2005)'s calculations for set-theoretic distributive lattices.

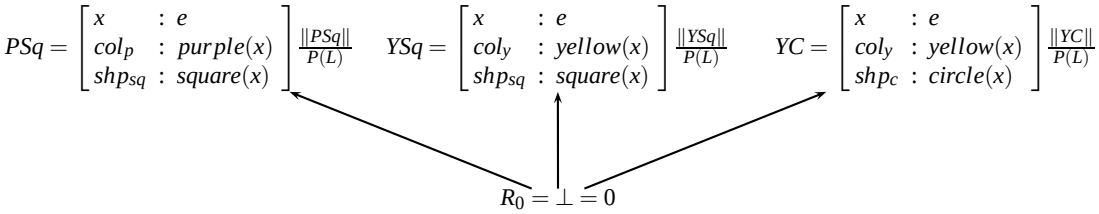


Figure 7.5: The disjunction of three types of situation encoded as record types

Algorithm 7 Probabilistic TTR record type lattice construction algorithm

INPUT: atoms ▷ Use the initial prior atomic judgements for L.
OUTPUT: L
L = newLattice(atoms) ▷ P(L) set to equal sum of the atomic probs.
agenda = atoms ▷ Initialise agenda.
while not agenda is empty **do**
  RT = agenda.pop()
  for field ∈ RT **do**
    if field ∈ RT.paths **then** ▷ Do not remove bound fields.
      continue
    superRT = RT - field ▷ Remove field.
    if superRT ∈ L **then** ▷ Not new? Order w.r.t. RT and inherit RT's atom set by union.
      L.order(RT.address,L.getNode(superRT),⊑)
    else ▷ New?
      superRTNode = L.newNode(superRT) ▷ Create new node w. empty atom set.
      for node ∈ L **do** ▷ Order superRTNode w.r.t. other nodes in L.
        if superRT.fields ⊂ node.fields. **then**
          L.order(node,superRTNode,⊑) ▷ superRTNode inherits node's atom set.
      agenda.append(superRT) ▷ Add to agenda for further supertyping.

7.5.3 RT lattice construction

Given the disjunction of type judgements is available with some prior probability assigned to each atom, it is possible to build a RT lattice from these which includes all possible supertype judgements with their probability values. I define a simple bottom-up procedure in Algorithm 7 to build a RT lattice L as a graph of all possible simple domain RT supertypes and their prior probabilistic judgements, initialised by the disjunction of possible final state judgements (the atoms), along with the absurdity $\bot$, stipulated a priori as the least element with probability 0 (see Figure 7.5).[7] The algorithm iteratively removes one field from the RT being processed at a time

[7] I repeat the fact that although the atoms' disjunctive probability sums to 1 after L is constructed when constructed via normalising over the whole probability mass $P(L)$, i.e. in Figure 7.6 $\frac{\|PSQ\|+\|YSQ\|+\|YC\|}{P(L)} = 1$, the real values initially assigned to them need not sum to unity (Knuth, 2005).

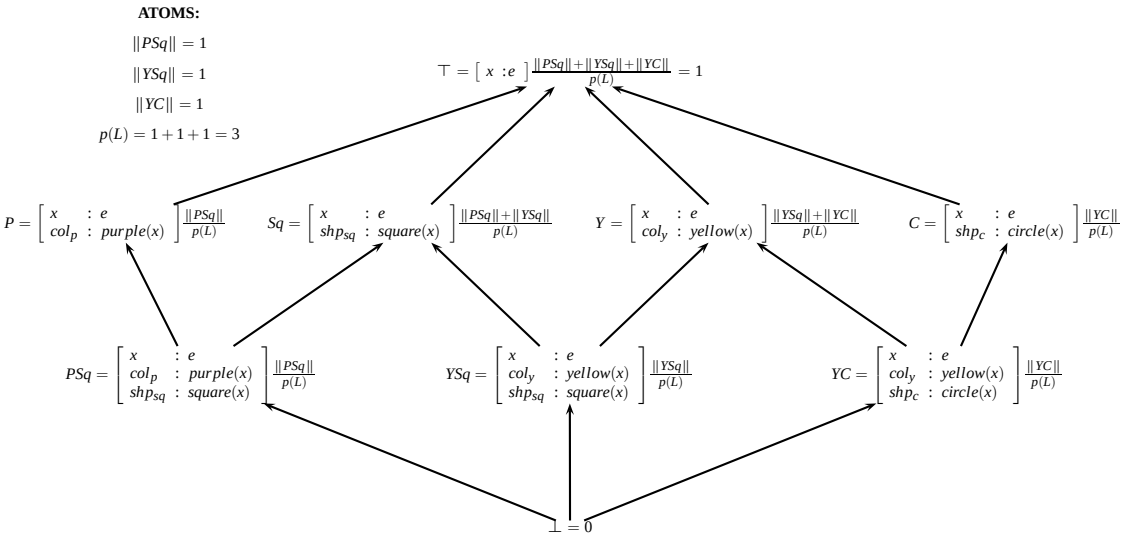


Figure 7.6: Record type lattice L with uniform atomic probabilities

(except bound fields that are referenced in any remaining *PType* fields dependent on them), then orders the new supertype RT in L appropriately.

As a graph, each node in L contains its RT R_i and a sum of probability judgements $\{\|R_k\| + .. + \|R_n\|\}$ corresponding to the probabilities of the atomic type judgement sets it stands in a supertype relation to– as these sums are stipulated to be totals of values of sets rather than bags, any duplicate values (from the same atom) are removed as the algorithm builds the lattice. These sums are propagated up from child to parent node as it is constructed. It terminates when all simple minimal common supertypes have been processed,[8] leaving the maximally common supertype of the whole lattice as $\top$ (possibly the empty type [ ]), associated with the entire probability mass $P(L)$, which constitutes the denominator to all judgements– given this, only the numerator of the equation for probability needs to be stored at each node.

7.5.4 Inference on RT lattices

The RT lattice L constructed initially upon observation of the game (by instructor or instructee) is shown in Figure 7.6, as described above using a uniform distribution for the three disjunctive final situations. Each node shows a RT R_i on the left and the derivation of its prior probability $p(s : R_i)$[9] that any game situation record will be of type R_i on the right, purely in terms of the

[8] Note again that it does not generate the join types but minimal common supertypes $\vee$ defined by field intersection– see (7.1).

[9] This should technically be $p(s : R_i | s : \top)$ and is an example of the suppression of the conditional judgement $s : \top$ as discussed above. From here these conditioning judgements will be suppressed to avoid

relevant atoms and the global denominator $P(L)$.

L is used as a reasoning system to make inferences in light of partial information from an ongoing utterance. We model inference as predicting the likelihood of relevant type judgements $R_y \in L$ of a situation s, given judgements of the form $s : R_x$ we have so far. To do this we use conditional probability judgements following Knuth's work on distributive lattices as described above, but here using the $\sqsubseteq$ relation to give (7.15).

$$p(s : R_y \mid s : R_x) = \begin{cases} 1 & \text{if } R_x \sqsubseteq R_y \\ 0 & \text{if } R_x \mathbin{\barwedge} R_y = \bot \\ p & \text{otherwise, where } 0 \leq p \leq 1 \end{cases} \tag{7.15}$$

If treated as a lazy evaluation function, in cases where the first two cases do not apply, the third case, the real-valued degree of inclusion of R_y in R_x, can be calculated using Cooper et al's conditional probability calculation (7.2) in Section 7.4.1, however replacing the $\wedge$ with $\barwedge$ to be in line with the meet operation of the RT lattice, and which is still equivalent to Cooper et al's equation due to Remark (7.6), giving (7.16).

$$p(s : R_2 \mid s : R_1) = \frac{p(s : R_1 \mathbin{\barwedge} R_2)}{p(s : R_1)} \tag{7.16}$$

When conditioning on negative RTs, given a lattice generated from Algorithm 7 will be distributive but not guaranteed to be complemented, we cannot be guaranteed to find a unique complement element on L as was the case for Knuth's Boolean lattices, however we can still calculate $p(s : R_y \mid s : \neg R_x)$ by obtaining $p(s : R_y)$ in L modulo the probability mass of R_x and that of its subtypes as in (7.17).

$$p(s : R_y \mid s : \neg R_x) = \begin{cases} 0 & \text{if } R_y \sqsubseteq R_x \\ \frac{p(s:R_y) - p(s:R_x \mathbin{\barwedge} R_y)}{p(s:\top) - p(s:R_x)} & \text{otherwise} \end{cases} \tag{7.17}$$

Efficiency gains through graphical search

The subtype relations and atomic and meet type probabilities required for (7.15) - (7.17) can be calculated efficiently through graphical search algorithms by characterising L as a DAG: the

notational clutter and they do not affect the calculations.

reverse direction of the subtype ordering edges can be viewed as reachability edges, making $\top$ the source and $\bot$ the sink. With this characterisation, if R_x is reachable from R_y then $R_x \sqsubseteq R_y$.

In DAG terms, the probability of the meet of two RTs R_x and R_y, $R_x \barwedge R_y$, can be found at their highest common descendant node – e.g. $p(s : Y \barwedge Sq)$ in Figure 7.6 can be found as $\frac{1}{3}$ directly at node *YSq*. Note if R_x is reachable from R_y, i.e. $R_x \sqsubseteq R_y$, then due to the equivalences listed in (7.5) and by Remark (7.6), $p(s : R_x \barwedge R_y)$ can be found directly at R_x. If the meet of two nodes is $\bot$ (e.g. *YSq* and *PSq* in Figure 7.6), then their meet probability is 0 as $p(\bot)$=0. In addition to this, due to Remark (7.6), the probability of all *meet types* of any combination of elements can also be found in exactly the same way.

As for the probability of the join of two RTs R_x and R_y, $R_x \barvee R_y$, this can be found at their lowest common ancestor node – e.g. $p(s : YSq \barvee PSq$ in Figure 7.6 can be found as $\frac{2}{3}$ directly at node *Sq*. Note if R_x is reachable from R_y, i.e. if $R_x \sqsubseteq R_y$, then due to the equivalence available for this ordering situation $R_y \barvee R_x = R_y$, then $p(s : R_x \barvee R_y)$ can be found directly at node R_y. It is worth pointing out here again the fact the lattice does not have direct access to the extensional equivalents of all the join types of all its elements, a join type probability $p(s : R_x \vee R_y)$ can be calculated in terms of $p(s : R_x \barwedge R_y)$ by Cooper et al's join equation in (7.3) and the equivalence of $\barwedge$ and $\wedge$ in extension and therefore probability. This way of calculating the join in terms of the meet as per probability theory holds for all probabilistic distributive lattices (Knuth, 2005), and is not restricted to Boolean lattices.

As regards search efficiency, worst case complexity for finding the meet probability at the common descendant of R_x and R_y is a linear $O(m+n)$ where m and n are the number of edges in the downward (possibly forked) paths $R_x \to \bot$ and $R_y \to \bot$. The search for the probability of the meet of two elements is generalisable to general meet probability of multiple elements by searching for the conjuncts' highest common descendant. The join probability is generalisable to the generalised join probability of multiple types, used, albeit programatically, in Algorithm 7 for calculating a node's probability from its child nodes.[10]

7.5.5 The question lattice and relevance

It is possible to generate a question lattice $Q(L)$ as a lattice of L's down-sets ordered by set inclusion as Knuth (2005, 2006) describes. The question lattice $Q(L)$ generated by starting with

[10]While I do not give details here, simple graphical search algorithms for general conjunctive and disjunctive multiple types are linear in the number of conjuncts and disjuncts, saving considerable time in comparison to the algebraic calculations of the generalised sum and product rules for distributive lattices.

L as shown in Figure 7.6 is shown as a Hasse diagram in Figure 7.7. The join-irreducible elements from which $Q(L)$ is generated are isomorphic to L (black nodes) and the real questions, those we are concerned with are white nodes.

Given this lattice-theoretic characterization, questions are therefore sets of record type judgements on the situation– to avoid clutter we abbreviate the set names $s:R_x \vee s:R_y$ to 'R_x or R_y' in Figure 7.7. The central issue in $Q(L)$ is $s:PSq \cup s:YSq \cup s:YC$ or in short 'YSq or PSq or YC' ($\approx$ "which of the three objects is it?"), which I will label I and whose content is as in (7.18).

$$I = s:PSq \cup s:YSq \cup s:YC = \{s:PSq, s:YSq, s:YC, s:\bot\} \tag{7.18}$$

The prior relevance measures $d(Q_x|\top)$ of questions $Q_x \in Q(L)$ are calculated from the initial assignments to the join-irreducible elements of $Q(L)$ (black nodes), which are simply the probability-weighted surprise of their top element in the assertion lattice L from which they were generated. The other relevances are calculated in a bottom up fashion by calculating joins by using the generalised sum rule (7.19) adapted from Knuth (2005), which follows the inclusion-exclusion principle for set union: it adds the relevances of its individual join elements, subtracts all pair-wise meets of those elements, adds all three-way meets, subtracts all four-way meets and so on in an alternating fashion until it reaches an n-way meet where n is the number of elements in the join. Fortunately in the lattice $Q(L)$ discussed here there is only one four-way join to calculate.

$$\begin{aligned} d(Q_1 \cup Q_2 \cup \cdots \cup Q_n \mid \top) = & \sum_{i=1}^{n} d(Q_i \mid \top) \\ & - \sum_{i<j} d(Q_i \cap Q_j \mid \top) \\ & + \sum_{i<j<k} d(Q_i \cap Q_j \cap Q_k \mid \top) \\ & - \cdots \end{aligned} \tag{7.19}$$

To focus on the real questions we extract the real sub-lattice $R(Q(L))$ of questions which are guaranteed a true answer, and this is shown in Figure 7.8. The bottom of this lattice is the central issue I and the relevance of each real question Q to it, i.e. $d(I|Q)$ is calculated using (7.11) assigning c the maximal relevance constant as the maximal relevance, i.e. $d(I \mid \top)$, which given

Knuth's calculation of this being the Shannon entropy of the top elements of its disjuncts' top element on the assertion lattice (given it is a join of ideal questions) to 4 s.f. is:

$$\begin{aligned} d(I \mid \top) &= d(s:PSq \cup s:YSq \cup s:YC \mid \bot) \\ &= -log_2(\frac{1}{3}) - log_2(\frac{1}{3}) - log_2(\frac{1}{3}) \\ &= 1.585 \end{aligned} \tag{7.20}$$

The relevance to the central issue $d(I|Q_x)$ is shown in the nodes, with the maximal relevance being 1.585. Several of the questions share the maximal relevance, and these are unsurprising: three of the partition questions 'C or P or YSQ', 'C or PSQ or YSQ' and 'P or YSQ or YC' if resolved are guaranteed to resolve the central issue I, due to the uniqueness of the circle shape and the purple colour, and so can be thought of as equivalent to I.

7.5.6 Extending NLU and NLG with probability and relevance

In addition to finding out the relevance of a question to the central issue I, motivation from NLU and NLG suggests we are also interested in how relevant a given type judgement is to it, or the degree to which it answers I. I assume that the relevance of a given type judgement in this domain is proportional to the most relevant question it answers, assuming for now this proportionality is in fact identity as in (7.21).

$$relevance(s:R \mid I) = \arg\max_{Q \in Q(L)} d(I|Q) \text{ where } s:R \in Q \tag{7.21}$$

For this to become a discourse model, it would need to include a record of which questions have been answered and which are yet to be answered, and also a notion of what it is for a question to be *resolved* in the sense of (Ginzburg, 1996). Given the characterization of questions as sets of answers, one could say a question has been answered when one of its members (type judgements) has been witnessed as being true. Given the structure of the question lattice, if a question has been answered then its entire upper bound has also been answered and in the assertion type lattice L if a type judgement has been witnessed, then the judgements of its supertypes (its upper bound) have also been witnessed. However the question to be resolved, or fully answered, one of its disjunct questions must be true and the others false. In $Q(L)$ interestingly this

is not possible for all questions. For example if the judgement made on the assertion lattice is $s : YC$ then the question 'Y or C' cannot be resolved as both disjuncts are true, so it can only be partially answered. The only questions that are resolvable are the partition questions, which in the real lattice are shown as ovals with an outer-ring in Figure 7.8.

As for the discourse record, the record of the judgements made and the questions answered at a given point in time can be kept track of by storing the position on L of the most recent type judgement and the position(s) of the most relevant question(s) resolved by it on $Q(L)$ word-by-word. Increased relevance therefore amounts to downward movement on both lattices. The notion of the most relevant question answered so far and the most relevant remaining questions are important when modelling repair, and for incremental inference more generally.

Redefining NLU and NLG

With the formulation of the relevance of a type judgement at hand (7.21), I make a modification to the NLU process in the last chapter which simply took Boolean parseability results and structural similarity to RT judgements in the domain as its measures of communicative felicity. I redefine the best-first NLU process as: given a word W_n and a DS-TTR parse state created from the current best ParseIU, select the ParseIU S_x such that $s : S_x$ in L leads to the combined maximally relevant question(s) being resolved in $Q(L)$ and maximal value of its prob field. These two constraints are combined to define the best parse as in (7.22).

$$(NLU)\text{Select ParseIU } S_n^j \text{ triggered by word } W_n \text{ according to:}$$

$$\arg\max_{S_n^j} \mathit{BestParse}(S_n^j \mid W_n) = \arg\max_{S_n^j.\text{prob}, S_n^j.\text{cont}} S_n^j.\text{prob} \times \mathit{relevance}(s : S_n^j.\text{cont}, I) \tag{7.22}$$

This definition makes the relevance of the current semantics built up to the central issue and the syntactic DS parseability both important in choosing the best update to the ParseIU graph. This simple multiplication means extremes in syntactic fluency or relevance will have a strong effect on parsing scores– weighting the contributions of these terms could be achieved through empirical testing.

For generation, which is contingent on the parsing model, it is possible to use the *BestParse* values within the decision of selecting the optimal ParseIU increment and word to generate. I

define this decision of solving the joint problem of maximising the probability of the word's *BestParse* (most relevant and most likely syntactically) from the given context and also maximising the likelihood of the goal concept GC_n given the ParseIU increment the word will trigger. This is formulated in (7.23).

(NLG)Select WordIU W_n^k and ParseIU S_n^j triggered by goal concept GC_n according to:

$$\arg\max_{S_n^j, W_n^k} BestGenerate(S_n^j, W_n^k | GC_n) = \arg\max_{S_n^j, W_n^k} BestParse(S_n^j \mid W_n^k) \times p(s : GC_n \mid s : S_n^j) \quad (7.23)$$

This formulation of NLG balances the maxims of quality, relevance and manner in Gricean terms. The quality part of the generation goal is the degree of adherence to the goal concept given the semantic update chosen, the relevance is just the degree to which the semantic update answers the central issue on $Q(L)$ and manner is the recoverability of the intended semantic content given the word, i.e. the parse probability.

7.6 Simulating incremental inference and self-repair processing

For the two test case utterances containing self-repairs, "the yellow, uh, purple square" and "the purple square, no, yellow", I show the probability distribution over the referent set generated by the model just described at each word in Figure 7.10. The second row in each table also shows the incremental type judgement on L by which these values are calculated conditionally from equations (7.15)-(7.17). The type judgements are available from the maximal semantics and also negation of the reparandum type judgement after self-repair detection from the `DyLan` NLU model– see the last chapter.

Probabilistic NLU in `DyLan` which interfaces with the RT lattice L described above follows evidence that dialogue agents parse self-repairs efficiently and that repaired dialogue content (reparanda) is given special status but not removed from the discourse context. To model Brennan and Schober (2001)'s finding of disfluent spoken instructions speeding up object recognition, I demonstrate a self-repair parse in Figure 7.9 for "The yell-, uh, purple square" in the simple game of predicting the final situation from $\{PSq, YSq, YC\}$ continuously given the type judgements made so far. I describe the stages T1-T4 in terms of the current word being processed- see Figure 7.9.

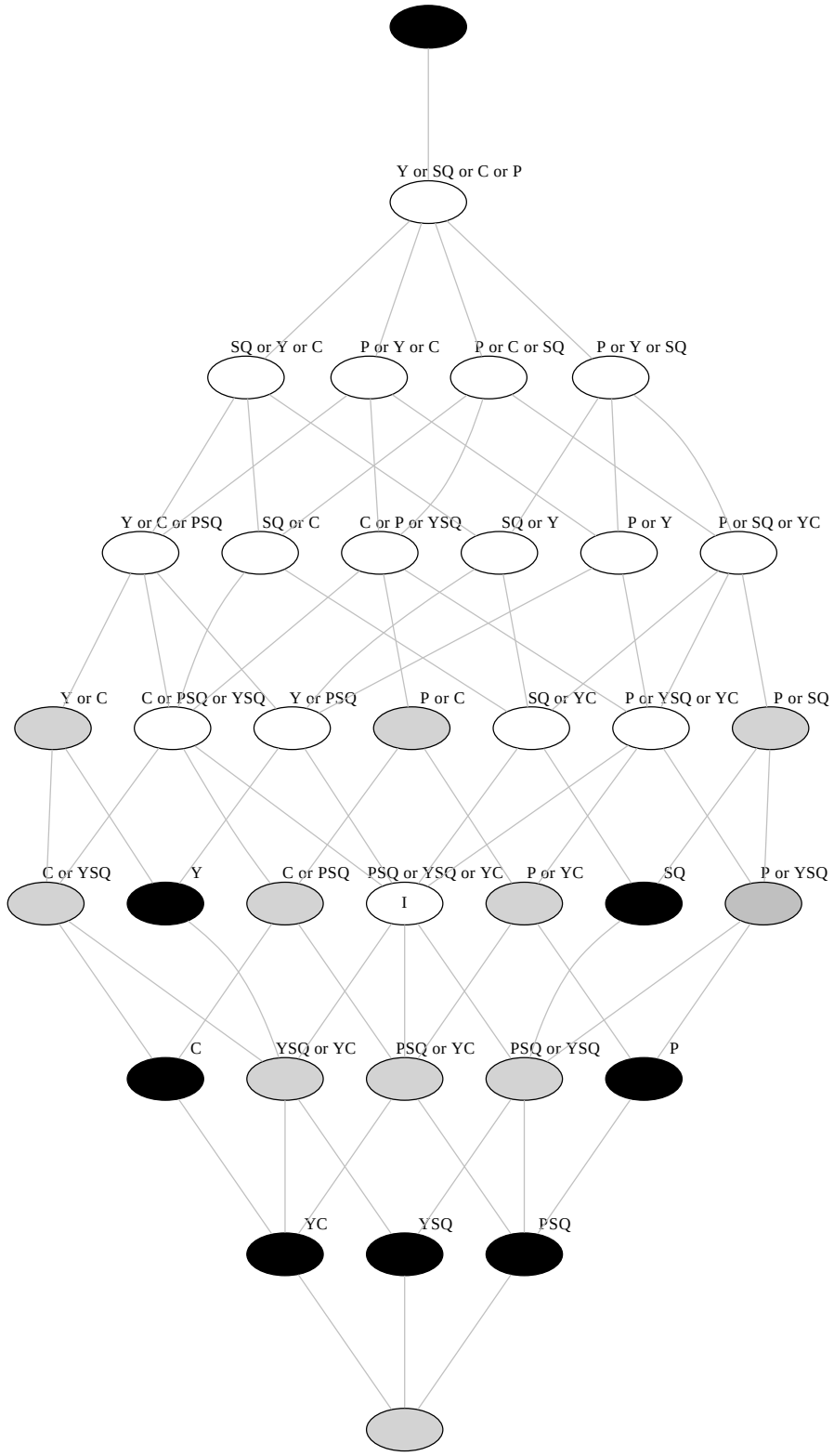


Figure 7.7: The Question Lattice $Q(L)$ formed from the ordered down-sets of the assertion lattice L. Black nodes indicate the join-irreducible elements, isomorphic to L's elements. White nodes are the real questions.

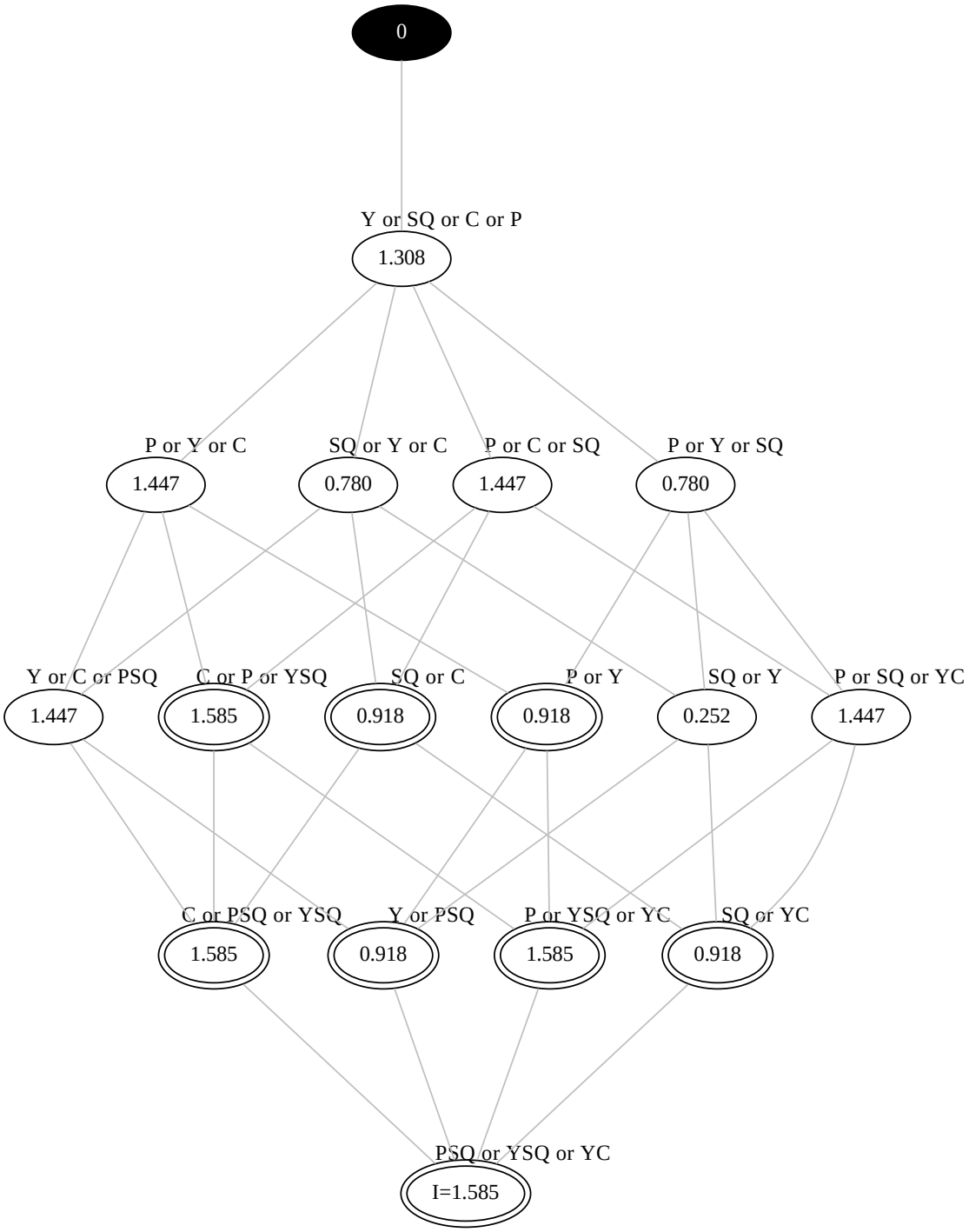


Figure 7.8: The lattice of real questions $R(Q(L))$, the sub-lattice of $Q(L)$ where answers of the question elements cover all the possibilities of the central issue I. The partition questions are ovals with an outer ring. Each question Q's relevance to I ($d(I \mid Q)$) is given within the node in its position, with maximal relevance 1.585

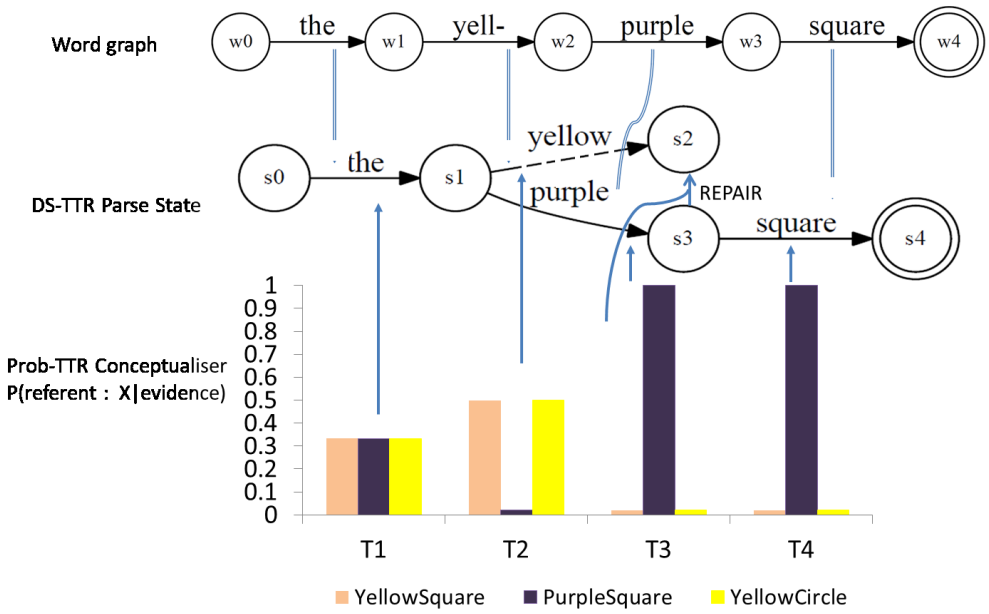


Figure 7.9: Incremental DS-TTR self-repair parsing with probabilistic TTR inference for reference resolution. Inter-graph *grounded in* links go bottom to top.

At **T1:'the'** the interpreter will not yield a subtype checkable in L so it is only possible to condition on $\top$, giving a uniform $p(s:x \mid s:\top) = \frac{1}{3}$ for $x \in \{PSq, YSq, YC\}$, equivalent to the atomic priors. At **T2: 'yell-'**, the best partial word hypothesis is now "yellow";[11] the interpreter therefore outputs a RT which matches the type judgement $s:Y$ (i.e. that the referent is a yellow object). Taking this judgement as the conditioning evidence using function (7.15) we get $p(s:PSq \mid s:Y) = 0$ and using (7.2) we get $p(s:YSq \mid s:Y) = 0.5$ and $p(s:YC \mid s:Y) = 0.5$ (see the schematic probability distribution at stage T2 in Figure 7.9 for the three objects). The meet type probabilities required for the conditional probabilities can be found graphically as described above.

At **T3:'uh purple'**, low probability in the interpreter output causes a self-repair to be recognised, enforcing backtracking on the parse graph which operates as per the last chapter, but with the definition for $BestParse()$ being replaced with that in (7.22) and a drop below the given threshold ε for $BestParse()$ will cause the repair onset function to operate.

Upon detection of a self-repair that revokes $s:Y$, the type judgement $s:\neg Y$, i.e. that this is not

[11] This part is not implemented. In practice, ASR modules yielding partial results are less reliable than their non-incremental counterparts, but progress is being made here (Schlangen and Skantze, 2009).

	the	yell-	uh	purple	square
conditioning type judgement $s:$	$\top$	Y	Y	$\neg Y, P$	PSq
$p(s:PSq)$ (purple square)	$\frac{1}{3}$	0	0	1,1	1
$p(s:YSq)$ (yellow square)	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{2}$	0,0	0
$p(s:YC)$ (yellow circle)	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{2}$	0,0	0

	the	purple	square	no	yellow
conditioning type judgement $s:$	$\top$	P	PSq	$\neg PSq$	$Y \wedge Sq$
$p(s:PSq)$ (purple square)	$\frac{1}{3}$	1	1	0	0
$p(s:YSq)$ (yellow square)	$\frac{1}{3}$	0	0	$\frac{1}{2}$	1
$p(s:YC)$ (yellow circle)	$\frac{1}{3}$	0	0	$\frac{1}{2}$	0

Figure 7.10: Probability distributions for the objects given maximal incremental semantic information

a yellow object, is immediately available as conditioning evidence using the *Revoke* information added when the continuation problem is solved, even before the full parse has been made– see Figure 6.9 in the last chapter. Using (7.17) our distribution of RT judgements now shifts: $p(s:PSq \mid s:\neg Y) = 1$, $p(s:YSq \mid s:\neg Y) = 0$ and $p(s:YC \mid s:\neg Y) = 0$ before "purple" has been parsed – thus providing a probabilistic explanation for increased subsequent processing speed. The model does not stipulate when would be realistic for the negative type inference to be made in terms of phonetic form, but Brennan and Schober (2001)'s results suggest this information becomes available very quickly upon the onset of the next word and the recognition of the substitution repair. I illustrate this in Figure 7.10 by including the effect of the negative type judgement in 'purple' before the positive judgement's effect (which is semantically identical)- I suggest this would be available before the positive judgement. Finally at **T4: 'square'** given $p(s:PSq \mid s:PSq) = 1$ and due to the fact $R_1 \wedge R_2 = R_1 \wedge R_3 = \bot$, the distribution remains unchanged. The last word could be taken as an instance of over-specification here, however, I follow Fernández (2013)'s idea that rather this being an inherent element of REG as per traditional accounts, this is just evidence syntactic completeness is preferred to incompleteness.

The system's processing models how listeners reason about the revocation itself rather than predicting the outcome through positive evidence alone, in line with Brennan and Schober (2001)'s results.

To explain the second example as shown in the lower table in Figure 7.10 in terms of NLU and NLG, I use the notion of relevance of the word 'no' in terms of the question lattice. I assume

it is more likely the **"no"** here is a negation of the judgement that the object is a purple square, i.e. $s : \neg PSq$, rather than the judgement it is purple, or that it is a square. In the spirit of optimal relevance (Sperber and Wilson, 1986), I assume the hearer is interpreting each word as an answer to the most relevant question under discussion incrementally. The interregnum "no" is interpreted as an answer to the most relevant question to the central issue just answered given the context "the purple square", which in this case is the central issue I. However this negative answer does not *resolve* I, as $s : YC$ and $s : YSq$ are still possible. It does however resolve 'PSq or Y', which is a more relevant question than resolving 'P or Y' if the inference was $s : \neg P$ and more relevant than resolving 'Sq or C' if the inference was $s : \neg SQ$– see Figure 7.8. So, the most likely inference from "no" is $s : \neg PSq$ due to its high relevance (see (7.21)), and all previously answered questions remain answered unless $s : \neg PSq$ changes them. The answer to the previously answered question 'Sq or C', where it has been established that $s : Sq$ is not revoked as it is not inconsistent with $s : \neg PSq$.

Next, **'yellow'** explicitly answers the question 'Y or PSq'. If the model only conditions on the evidence $s : \neg PSq$, then this adds no information, or is *irrelevant*. If the previous judgement that this was a square $s : Sq$ is incorporated however (given there is no evidence in the repair this has changed to $s : \neg Sq$) this answers the most relevant question I. This ellipsis resolution on this account is driven by interpreting the fragment as an answer to the most relevant outstanding question for reference resolution. Having said this, DyLan's NLU algorithm will also give this judgement through maximal re-use of the reparandum syntactically through use of asymmetric merge, as discussed in the last chapter, however here the interpretation is aided by a notion of relevance given a new notion of interpretation in *BestParse* (7.22). The judgement $s : YSq$ is of course defeasible if "circle" were to be the following word, however at this point the hearer has assumed optimal relevance in terms of the speaker answering the central issue as efficiently as possible, that is, assuming Gricean maxims.

In NLG, there is now a strategic reason for generating 'no'– it is the most efficient way of negating the answer to the most relevant question that has just been resolved. In the first example, the combination of the hiatus of the word, the interregnum and the onset of a substitution form indicate the revocation of the reparandum, whereas here this is done explicitly in terms of an answer. Conversely on the NLU side, interpreting any yes or no answer involves a question accommodation search process to find the most relevant question it answers. There is also a

strategic reason for the elliptical form being generated here, in that it is the increment that resolves the most relevant question under discussion yet to be resolved, given the parse state's maximal re-use of the reparandum.

7.7 Discussion

I have presented a novel way of using Knuth (2005, 2006)'s work on probabilistic lattices for type theory which has some nice predictions for small reference domains. DS-TTR, whilst currently not fully implemented probabilistically, has potential for fully probabilistic parsing and generation in practice. The question-based account has the same spirit as Ginzburg et al. (2014), however here I have introduced a real-valued measure of relevance, a measure which I believe should be present in any theory of dialogue.

One of the potential draw-backs of the approach is complexity blow-up and scalability. There is exponentiation of the size of the lattices in the size of the disjoint atoms. The other obvious difficulty when scaling to bigger domains is defining the domain of type judgements, however the motivation of TTR is a good one: an agent should only reason with the relevant types to a situation, rather than regarding the whole universe and all the type judgements therein.

The model does not capture over-specification in REG directly in the way the incremental algorithm does (Dale and Reiter, 1995), however given the cross-linguistic evidence (Rubio-Fernández, 2011) this may not be a weakness: as over-specification may be tied to specific constructions in specific situations for a given language, it may not do to model it in the conceptualisation stage, but rather as a side-effect of incremental informativity, which my model attempts to capture, like Fernández (2013). However, uniquely, the model accounts for the reasoning of both the listener and the producer of the instructions, due to its reversibility. In NLU, choosing the optimal next parse increment given an input word conditions on the relevance of candidate increments to the central issue and their syntactic probability (7.22), while in NLG, choosing the optimal next word conditions both on its likelihood to be parsed appropriately, and that parse increment's closeness to the goal concept (7.23). In future work, this could be used to model the cause of self-repair as a time critical trade-off between relevance and accuracy.

A computational view of relevance in terms of information content and answering questions incrementally also has scope for extension beyond self-repair to other-repair. If each dialogue partner is modelled as having their own assertion lattice and question lattice, the potential for

clarification, and the knowledge of that clarification, becomes clear– this is a clearly vital form of dialogue act that has had treatments in terms of QUD structures (Purver, 2004; Ginzburg, 2012) but here this can be cashed out in terms of probabilistic and information-theoretic situational inference. Beginning with these repair phenomena, along with simple questions and assertions may open the door to integrating an incrementally constructed QUD into the type of dialogue context I have set out here more generally, and provide some unification between KoS, DS-TTR and a thorough-going computational formulation of relevance.

7.7.1 Extensions

I now mention a few other possible future extensions to the model.

Dialogue and self-repair in the wild

To move towards domain-generality, generating the lattice of all possible dialogue situations for interesting domains is computationally intractable. It would be more sensible instead to consider incrementally occurring *issues* that can be modelled as questions (Larsson, 2002). Given one or more issues manifest in the dialogue at any time, it is plausible to generate small assertion lattices dynamically to estimate possible answers, and also assign the real-valued relevance measures to questions generated from the assertion lattices. This is an approach that can start small and build up.

Learning in a referential game

The question of how a dialogue system could automatically learn through observation and interaction to best predict reference given other type judgements of words and ParseIUs is prescient. While not the focus here, I briefly explain how this could be done in the framework heretofore presented.

When dealing with referring expression games, or indeed any language game, we need a way of storing perceptual experience. In probabilistic TTR this was achieved by positing a judgement set J in which an agent stores probabilistic type judgements.[12] As decribed above, I extend this notion of the overall judgement set to the lattice of judgement sets for a given agent J, which I will call here L_J, and the sum of the value of probabilistic judgements that agent J has judged a situation be of type R_i within L_J as $\|R_i\|_J$ so far, and the sum of all probabilistic judgements in the lattice simply as $P(L_J)$; thus the prior probability that J judges anything is of type R_i under

[12] Cooper et al. (2014) characterise a type judgement as an Austinian proposition that a situation is of a given type with a given probability, encoded in a TTR record.

the set of judgements J is $\frac{\|R_i\|_J}{P(L_J)}$. The conditional probability $p(s : R_1 \mid s : R_2)$ under J can be reformulated in terms of these sets of judgements:

$$p_J(s : R_1 \mid s : R_2) = \begin{cases} \frac{\|R_1 \wedge R_2\|_J}{\|R_2\|_J} & \text{iff} \quad \|R_2\|_J \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.24}$$

where the sample spaces $\|R_1 \wedge R_2\|_J$ and $\|R_2\|_J$ constitute the observations of agent J so far.

In learning through experience, lattice L_J's probabilities can be updated through observations after its initial construction by adding judgements to the relevant judgement sets. If a reference game is played over several rounds, the choice of referring expression can change based on mutually salient functions from words to situations- see e.g. DeVault and Stone (2009). A frequentist approach to learning in this scenario would be: given an observation of an existing RT R_i is made by J with probability v, then $\|R_i\|_J$, the overall denominator $P(L_J)$, and the nodes in the upward path from $\|R_i\|_J$ to $\top$ are incremented by v. The approach could be converted to Bayesian update learning by using the prior probabilities in L_J for calculating v before it is added. Furthermore, observations can be added to L_J that include novel RTs: due to the DAG structure of L_J, their subtype ordering and probability effects can be integrated efficiently.

The reason for the seemingly trivial $_J$ subscript here is to emphasize an agent and perception-centric view. The lattice built by agent J may not be the same lattice as their interlocutor's lattice, or the probability values for the type judgements may be different. Consequently coordination is required to settle on the most effective values for the assertion lattice, and clarification questions generated from the question lattice dual to this could help guide this. As Knuth (2006) explains, the central issue is the central issue by virtue of the fact it is the most relevant in the domain– maximising its relevance is in fact maximising the entropy of the distribution of its possible answers. In future work I intend to see how this framework could be used for on-line negotiation of conceptual meaning between two agents.

7.8 Conclusion

I have discussed a dialogue model for incremental probabilistic inference and efficient methods for constructing probabilistic RT domain concept lattices ordered by the subtype relation, demonstrating their efficacy for realistic self-repair processing. The model recreates experimental results (Brennan and Schober, 2001) and develops ideas of ellipsis processing in terms of real-valued relevance measures to questions under discussion. While I model a simple reference

domain here, this is intended to be a general NLU and NLG model for dialogue, and as such, a question-based semantics is more suitable than one conditioning on pre-defined properties of objects as is the tendency for specific reference resolution and generation algorithms in the literature. I wish to explore the scalability of RT lattices to other domains and their learning capacity in future work.

Chapter 8

Conclusion and Future Directions

This brief concluding chapter summarizes the contributions of the thesis and their consequences. Future directions for research are also outlined.

8.1 Summary of contributions

From the four principal analysis chapters the following contributions were made:

- Chapter 4: A comprehensive study of the forms of repair showed how a string alignment approach to repair detection and one reliant on interregnum recognition has major weaknesses, and how an incremental processing account sensitive to local information in the utterance has the potential to be much stronger. Meaning of self-repair has regularity, though this may be gradient rather than categorical.

- Chapter 5: The strongly incremental repair detection system STIR is defined. STIR is able to achieve good performance on the Switchboard disfluency identification task whilst improving the state-of-the-art in incremental performance. Enriched language n-gram models show some useful information-theoretic features for fluency measurements. STIR is practically useful for psychiatric purposes.

- Chapter 6: An incremental semantics driven model of self-repair incorporated directly in both parsing and generation models is proposed and has some pleasing consequences. The account begins to fulfil the incremental requirements for dialogue systems, such as strong incremental interpretation and incremental representation.

- Chapter 7: The model from the previous chapter is extended to have a situation-based semantic model using lattice-theoretic measures of probability and relevance in a novel way for dialogue models, modelling psycholinguistic evidence on self-repair processing in a reference identification domain. The chapter shows that going probabilistic is the way towards realistic inference.

8.2 Consequences for theoretical dialogue models

Self-repairs and edit terms have an equal status semantically and pragmatically to fluent utterances

Through reviewing empirical evidence in Chapter 2 and observing real self-repairs in Chapter 4, it is clear speakers and hearers make use of the structure and type of self-repair to compute their meaning beyond that of filtering out their disfluency effects. Chapters 6 and 7 modelled this explicitly.

Actions speak louder than words in dialogue context

One theme of this thesis is that the notion of dialogue semantics should not be restricted to one that is just about propositional content, but one that has a fine-grained processing context recording *how* the content was produced. Chapter 6 showed how context could be characterized as action-based, which allows it to be modelled time-linearly in the way dialogue occurs in practice. It is fruitful, in the spirit of the original DS generation account in Purver and Kempson (2004), hearers and speakers have access to a similar action sequence structure which allows them to repair and process elliptical phenomena interactively.

Probability and information theory should be first class citizens of dialogue semantics

To model empirical phenomena, inclusion of probability and information theory seems to be, rather than for the purposes of optimising dialogue system performance, necessarily part of the semantics. There is now a well defined relationship between logic and probability (Knuth, 2005, 2006) which I hopefully showed in Chapter 7 linguists and dialogue system engineers can profit from. Real-valued degrees of relevance for questions is something that needs to be explored further to model realistic inference in dialogue models in general.

8.3 Consequences for computational linguistics and empirical models

Self-repair needs to be modelled as a mechanism, not a string-alignment process and its structure needs to be preserved

Chapter 4 and 5 showed how aligning strings is an ineffective method for detecting repairs, and this will only capture a subset and provide insufficient coverage and poor incremental performance. Self-repair needs to be viewed as a mechanism at work, and accurately modelling this in a strongly incremental way, as shown to be possible in Chapter 5 allows state-of-the-art incremental performance in detecting repair and edit term structures.

Left-to-right operability on its own is not sufficient for good incremental performance

Through the desiderata provided at the end of Chapter 3 and the novel evaluation measures provided in Chapter 5, it is clear incrementality is a multi-faceted capacity, and systems purporting to be incremental must perform well across these measures. These criteria extend beyond the phenomena studied in this thesis to dialogue processing generally.

Robustness for repair processing comes from robustness in the grammar

As Johnson (2011) points out, a *closed-world assumption* view of parsing– that is one regarding any analysis the grammar does not generate as ungrammatical– is not a healthy element of a parsing system, not only for reasons of robustness, but also for psychological plausibility. N-gram models as described in Chapter 5 are the most robust resource currently available to model this gradient effect, however as parsers become equally robust for coverage and probability output, incorporating self-repair processing within the parsing process is the most natural step, but using an approach that is based on detection and interpretation of repair rather than removing reparanda.

Information-theoretic measures are more useful than lexical values

The work here supports the idea of information density (Keller, 2004; Jaeger and Tily, 2011) being at the core of incremental dialogue processing. Approaches that put information theory directly into the representation, rather than being used peripherally for machine learning, are in a

good position to model interesting phenomena. This was shown in Chapter 5 where information-theoretic measures were shown to strongly correlated to self-repair structure, and automatic classifiers can use these measures without requiring big feature spaces encoding every word in the lexicon to perform well. It is clear investigation into the informational properties of dialogue is a very fruitful step to take.

8.4 Future directions

This is only the beginning. Integration of the mechanisms presented here with speech recognition and voice synthesis is a clear next step. This work is well placed for incremental situated dialogue, so should result in more interactive, likeable systems.

The notion of incremental context for dialogue sketched in this thesis needs extension. Using the insights of KoS (Ginzburg, 2012) to enrich this with grounding, clarification and other capabilities, whilst adhering to the principals of incrementality described here, could eventually result in a unifying theory of dialogue context.

Finally, as the models presented here have actions and time-linear updates to context at their core, they could generalise to non-linguistic repair. The optimisation of interactive action under time-pressure causes repair, and this is none more so the case than in pure spoken dialogue, however there are other domains with this property involving interactive action, for example in physical motor skill acquisition and teaching, where a multi-modal approach to repair could be a fruitful direction for embodied agents.

8.5 Final word

This thesis concludes that at least as fine-grained a model of context as word-by-word is required for realistic models of self-repair, and this context must include linguistic action sequences and information update effects. The way dialogue participants process self-repairs to make inferences in real time, rather than filter out their disfluency effects, has been modelled formally and in practical systems.

Bibliography

G. Aist, J. Allen, E. Campana, C.A. Gomez Gallo, S. Stoness, M. Swift, and M.K. Tanenhaus. 2007. Incremental dialogue system faster than and preferred to its nonincremental counterpart. In *Proceedings of the 29th Annual Conference of the Cognitive Science Society*.

James F. Allen, Mary Swift, and Will de Beaumont. 2008. Deep Semantic Analysis of Text. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 343–354. College Publications.

Anne Anderson, Miles Bader, Ellen Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, Catherine Sotillo, Henry Thompson, and Regina Weinert. 1991. The HCRC map task data. *Language and Speech*, 34 (4):351–366.

Douglas E. Appelt. 1987. Bidirectional grammars and the design of natural language generation systems. In *Proceedings of the 1987 workshop on Theoretical issues in natural language processing*, TINLAP '87, pages 206–212, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jennifer E Arnold, Carla L Hudson Kam, and Michael K Tanenhaus. 2007. If you say¡ em¿ thee uh¡/em¿ you are describing something hard: The on-line attribution of disfluency during reference comprehension. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 33(5):914.

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

T. Baumann, O. Buß, and D. Schlangen. 2011. Evaluation and optimisation of incremental processors. *Dialogue & Discourse*, 2(1):113–141.

Timo Baumann. 2013. *Incremental Spoken Dialogue Processing: Architecture and Lower-level Components*. PhD thesis.

A. Belz, M. White, J. van Genabith, D. Hogan, and A. Stent. 2010. Finding common ground: Towards a surface realisation shared task. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 268–272. Association for Computational Linguistics.

Gustavo Betarte and Alvaro Tasistro. 1998. Extension of Martin-Löf type theory with record types and subtyping. In G. Sambin and J. Smith, editors, *25 Years of Constructive Type Theory*. Oxford University Press.

Patrick Blackburn and Wilfried Meyer-Viol. 1994. Linguistics, logic and finite trees. *Bulletin of the IGPL*, 2:3–31.

H. Bortfeld, S.D. Leon, J.E. Bloom, M.F. Schober, and S.E. Brennan. 2001. Disfluency rates in conversation: Effects of age, relationship, topic, role, and gender. *Language and Speech*, 44 (2):123–147.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

S.E. Brennan and M.F. Schober. 2001. How listeners compensate for disfluencies in spontaneous speech* 1. *Journal of Memory and Language*, 44(2):274–296.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics.

E. J. Briscoe. 1987. *Modelling human speech comprehension: a computational approach*. Ellis Horwood ltd., Chichester.

Lou Burnard. 2000. *Reference Guide for the British National Corpus (World Edition)*. Oxford University Computing Services `http://www.natcorp.ox.ac.uk/docs/userManual/`.

Hendrik Buschmeier, Timo Baumann, Benjamin Dosch, Stefan Kopp, and David Schlangen. 2012. Combining incremental language generation and incremental speech synthesis for adaptive information presentation. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 295–303, Seoul, South Korea.

O. Buß and D. Schlangen. 2011. Dium–an incremental dialogue manager that can produce self-corrections. In *Proceedings of the 15th Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL)*, pages 47–54, Los Angeles, California.

Okko Buß, Timo Baumann, and David Schlangen. 2010. Collaborating on utterances with a spoken dialogue system using an ISU-based approach to incremental dialogue management. In *Proceedings of the SIGDIAL 2010 Conference*, pages 233–236, Tokyo, Japan. Association for Computational Linguistics.

S. Calhoun, J. Carletta, J.M. Brenier, N. Mayo, D. Jurafsky, M. Steedman, and D. Beaver. 2010. The nxt-format switchboard corpus: A rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language Resources and Evaluation*, 44(4):387–419.

Ronnie Cann. 2011. Towards an account of the english auxiliary system. In E. Gregoromichelaki, R. Kempson, and C. Howes, editors, *The Dynamics of Lexical Interfaces*, pages 279–317. CSLI.

Ronnie Cann, Tami Kaplan, and Ruth Kempson. 2005. Data at the grammar-pragmatics interface: the case of resumptive pronouns in English. *Lingua*, 115(11):1475–1665. Special Issue: On the Nature of Linguistic Data.

E. Charniak and M. Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–9. Association for Computational Linguistics.

Nick Chater, Joshua B Tenenbaum, and Alan Yuille. 2006. Probabilistic models of cognition: Conceptual foundations. *Trends in cognitive sciences*, 10(7):287–291.

Eunah Cho, Jan Niehues, and Alex Waibel. 2014. Tight integration of speech disfluency removal into smt. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 43–47, Gothenburg, Sweden. Association for Computational Linguistics.

N. Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT, Cambridge, MA.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Foris, Dordrecht.

Noam Chomsky. 1995. *An Essay on Minimalism*. MIT Press.

Alexander Clark, Gianluca Giorgolo, and Shalom Lappin. 2013. Statistical representation of grammaticality judgements: the limits of n-gram models. In *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, pages 28–36, Sofia, Bulgaria. Association for Computational Linguistics.

Herbert H. Clark. 1996. *Using Language*. Cambridge University Press.

Herbert H. Clark and Jean E. Fox Tree. 2002. Using *uh* and *um* in spontaneous speaking. *Cognition*, 84(1):73–111.

Marcus Colman and Patrick Healey. 2011. The distribution of repair in dialogue. In c. Hoelscher and T.F. Shipley, editors, *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, pages 1563–1568, Boston, Massachussetts. Austinm TX:Cognitive Science Society.

Robin Cooper. 2005. Records and record types in semantic theory. *Journal of Logic and Computation*, 15(2):99–112.

Robin Cooper. 2012. Type theory and semantics in flux. In Ruth Kempson, Nicholas Asher, and Tim Fernando, editors, *Handbook of the Philosophy of Science*, volume 14: Philosophy of Linguistics, pages 271–323. North Holland.

Robin Cooper, Simon Dobnik, Shalom Lappin, and Staffan Larsson. 2014. A probabilistic rich type theory for semantic interpretation. In *Proceedings of the EACL Workshop on Type Theory and Natural Language Semantics (TTNLS)*, Gothenburg, Sweden. Association for Computational Linguistics.

Mark G. Core and Lenhart K. Schubert. 1999. A syntactic framework for speech repairs and other disruptions. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 413–420, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

K. De Smedt and G. Kempen. 1991. Segment grammar: A formalism for incremental sentence generation. *Natural language generation in artificial intelligence and computational linguistics*, 119:329.

Koenraad De Smedt. 1990. IPF: An incremental parallel formulator. In *Current research in natural language generation*, pages 167–192. Academic Press, London.

Koenraad De Smedt. 1991. Revisions during generation using non-destructive unification. In *Proceedings of the Third European Workshop on Natural Language Generation*, pages 63–70.

V. Demberg and F. Keller. 2008. A psycholinguistically motivated version of tag. In *Proceedings of the 9th International Workshop on Tree Adjoining Grammars and Related Formalisms. Tübingen*, pages 25–32.

David DeVault and Matthew Stone. 2009. Learning to interpret utterances using dialogue history. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 184–192, Athens, Greece. Association for Computational Linguistics.

David DeVault, Kenji Sagae, and David Traum. 2011. Incremental interpretation and prediction of utterance meaning for interactive dialogue. *Dialogue and Discourse*, 2(1):143–170.

Simon Dobnik, Robin Cooper, and Staffan Larsson. 2013. Modelling language, action, and perception in type theory with records. In *Constraint Solving and Language Processing*, pages 70–91. Springer.

Pedro Domingos. 1999. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164. ACM.

Robert Eklund. 2004. *Disfluency in Swedish human–human and human–machine travel booking dialogues*. PhD thesis, Linköping University, Sweden.

Arash Eshghi. 2009. *Uncommon ground: The distribution of dialogue contexts*. PhD thesis, School of Electronic Engineering and Computer Science, Queen Mary University of London.

Arash Eshghi, Julian Hough, Matthew Purver, Ruth Kempson, and Eleni Gregoromichelaki. 2012. Conversational interactions: Capturing dialogue dynamics. In S. Larsson and L. Borin, editors, *From Quantification to Conversation: Festschrift for Robin Cooper on the occasion of his 65th birthday*, volume 19 of *Tributes*, pages 325–349. College Publications, London.

Arash Eshghi, Julian Hough, and Matthew Purver. 2013. Incremental grammar induction from child-directed dialogue utterances. In *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, pages 94–103, Sofia, Bulgaria. Association for Computational Linguistics.

Lauren Faust and Ron Artstein. 2013. People hesitate more, talk less to virtual interviewers than to human interviewers. In *Proceedings of the 17th SemDial Workshop on the Semantics and Pragmatics of Dialogue (DialDam)*, pages 35–43, Amsterdam.

Raquel Fernández. 2013. Rethinking overspecification in terms of incremental processing. In *Proceedings of the PRE-CogSci 2013 Workshop on the Production of Referring Expressions*, Berlin,Germany.

Raquel Fernández. 2006. *Non-Sentential Utterances in Dialogue: Classification, Resolution and Use*. PhD thesis, King's College London, University of London.

Fernanda Ferreira, Ellen F Lau, and Karl GD Bailey. 2004. Disfluencies, language comprehension, and tree adjoining grammars. *Cognitive Science*, 28(5):721–749.

Michael C Frank and Noah D Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998.

Andrew Gargett, Eleni Gregoromichelaki, Ruth Kempson, Matthew Purver, and Yo Sato. 2009. Grammar resources for modelling dialogue dynamically. *Cognitive Neurodynamics*, 3(4):347–363.

Kallirroi Georgila. 2009. Using integer linear programming for detecting speech disfluencies. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 109–112. Association for Computational Linguistics.

Sebastian Germesin, Tilman Becker, and Peter Poller. 2008. Hybrid multi-step disfluency detection. In *Machine Learning for Multimodal Interaction*, pages 185–195. Springer.

Jonathan Ginzburg. 2012. *The Interactive Stance: Meaning for Conversation*. Oxford University Press.

Jonathan Ginzburg. 1996. Interrogatives: Questions, facts and dialogue. In S. Lappin, editor, *The Handbook of Contemporary Semantic Theory*, pages 385–422. Blackwell.

Jonathan Ginzburg, Raquel Fernández, and David Schlangen. 2007. Unifying self- and other-repair. In *Proceedings of the 11th Workshop on the Semantics and Pragmatics of Dialogue (DECALOG)*.

Jonathan Ginzburg, Raquel Fernández, and David Schlangen. 2014. Disfluencies as intra-utterance dialogue moves. *Semantics and Pragmatics*, 7(9):1–64.

John J. Godfrey, Edward Holliman, and J. McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of IEEE ICASSP-92*, pages 517–520, San Francisco, CA.

H.P. Grice. 1975. Logic and Conversation. *Syntax and Semantics*, 3(S 41):58.

Jeroen Groenendijk and Martin Stokhof. 1984. On the semantics of questions and the pragmatics of answers. In F. Landman and F. Veltman, editors, *Varieties of Formal Semantics*, volume 3 of *Groningen-Amsterdam Studies in Semantics (GRASS)*, pages 143–170. Foris.

M. Guhe and F. Schilder. 2002. Incremental generation of self-corrections using underspecification. *Language and Computers*, 45(1):118–132.

Markus Guhe. 2007. *Incremental Conceptualization for Language Production*. NJ: Lawrence Erlbaum Associates.

Markus Guhe and Christopher Habel. 2001. The influence of resource parameters on incremental conceptualization. In *Proceedings of the 2001 Fourth International Conference on Cognitive Modeling*, pages 26–28.

John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the 2nd Conference of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, PA.

John Hale, Izhak Shafran, Lisa Yung, Bonnie Dorr, Mary Harper, Anna Krasnyanskaya, Matthew Lease, Yang Liu, Brian Roark, Matthew Snover, et al. 2006. Pcfgs with syntactic and prosodic indicators of speech repairs. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 161–168. Association for Computational Linguistics.

Charles L Hamblin. 1973. Questions in montague english. *Foundations of language*, pages 41–53.

P. G. T. Healey, M. Colman, and M. Thirlwell. 2005. Analysing multi-modal communication: Repair-based measures of human communicative co-ordination. In J. van Kuppevelt, L. Dybkjaer, and N. Bernsen, editors, *Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems*. Kluwer, Dordrecht.

Patrick Healey, Matthew Purver, James King, Jonathan Ginzburg, and Greg Mills. 2003. Experimenting with clarification in dialogue. In *Proceedings of the 25th Annual Meeting of the Cognitive Science Society*, Boston, Massachusetts.

Patrick G. T. Healey, Arash Eshghi, Christine Howes, and Matthew Purver. 2011. Making a contribution: Processing clarification requests in dialogue. In *Proceedings of the 21st Annual Meeting of the Society for Text and Discourse*, Poitiers.

P.G.T. Healey and M. Thirlwell. 2002. Analysing multi-modal communication: Repair-based measures of communicative co-ordination. In *Proceedings of the International CLASS Workshop on Natrual, Intelligent and Effective Interaction in Multimodal Dialogue Systems*, pages 83–92.

Peter Heeman and James Allen. 1995. The TRAINS 93 dialogues. Trains Technical Note 94-2, Computer Science Department, University of Rochester.

Peter Heeman and James Allen. 1999. Speech repairs, intonational phrases, and discourse markers: modeling speakers' utterances in spoken dialogue. *Computational Linguistics*, 25 (4):527–571.

D. Hindle. 1983. Deterministic parsing of syntactic non-fluencies. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 123–128. Association for Computational Linguistics.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association of Computational Linugistics (TACL)*, 2: 131–142.

Julian Hough. 2009. A study in natural language generation: Linguistic issues in developing a describer system for london buildings. Master's thesis, University College London, 2009.

Julian Hough. 2011. Incremental semantics driven natural language generation with self-repairing capability. In *Proceedings of the Student Research Workshop associated with RANLP 2011*, pages 79–84, Hissar, Bulgaria.

Julian Hough and Matthew Purver. 2012. Processing self-repairs in an incremental type-theoretic dialogue system. In *Proceedings of the 16th SemDial Workshop on the Semantics and Pragmatics of Dialogue (SeineDial)*, pages 136–144, Paris, France.

Julian Hough and Matthew Purver. 2013. Modelling expectation in the self-repair processing of annotat-, um, listeners. In *Proceedings of the 17th SemDial Workshop on the Semantics and Pragmatics of Dialogue (DialDam)*, pages 92–101, Amsterdam.

Julian Hough and Matthew Purver. 2014a. Probabilistic type theory for incremental dialogue processing. In *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pages 80–88, Gothenburg, Sweden. Association for Computational Linguistics.

Julian Hough and Matthew Purver. 2014b. Lattice theoretic relevance for incremental reference processing. Edinburgh. Poster presentation at REFNET Workshop on Computational and Psychological Models of Reference Comprehension and Production.

Julian Hough and Matthew Purver. 2014c. Strongly incremental repair detection. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 78–89, Doha, Qatar. Association for Computational Linguistics.

Christine Howes, Matthew Purver, Patrick G. T. Healey, Gregory J. Mills, and Eleni Gregoromichelaki. 2011. On incrementality in dialogue: Evidence from compound contributions. *Dialogue and Discourse*, 2(1):279–311.

Christine Howes, Matthew Purver, Rose McCabe, Patrick G. T. Healey, and Mary Lavelle. 2012. Helping the medicine go down: Repair and adherence in patient-clinician dialogues. In *Proceedings of the 16th Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2012)*, pages 155–156, Paris.

Christine Howes, Julian Hough, Matthew Purver, and Rose McCabe. 2014. Helping, I mean assessing psychiatric communication: An application of incremental self-repair detection. In *Proceedings of the 18th SemDial Workshop on the Semantics and Pragmatics of Dialogue (DialWatt)*, pages 80–89, Edinburgh.

T Florian Jaeger and Harry Tily. 2011. On language utility: Processing complexity and communicative efficiency. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(3):323–335.

Mark Johnson. 2011. How relevant is linguistics to computational linguistics? *Linguistic Issues in Language Technology*, 6(7).

Mark Johnson and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

D. Jurafsky, E. Shriberg, and D. Biasca. 1997. Switchboard swbd-damsl shallow-discourse-function annotation coders manual, draft 13. Technical report, University of Colorado, Boulder Institute of Cognitive Science Technical Report. URL `http://ics.colorado.edu/techpubs/pdf/97-02-part1.pdf`.

Daniel Jurafsky and James Martin. 2009. *Speech and Language Processing*. Pearson Prentice Hall, 2nd edition.

Hans Kamp. 1981. A theory of truth and semantic representation. In Anderson C. Groenendijk, J. and J. Owens, editors, *Truth, Interpretation and Information*, pages 1–32. Foris, Dordrecht.

Martin Kay. 1973. The mind system. In Randall Rustin, editor, *Natural Language Processing*, pages 155–188. Algorithmics Press, New York.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 200–204.

Frank Keller. 2004. The entropy rate principle as a predictor of processing effort: An evaluation against eye-tracking data. In *EMNLP*, pages 317–324.

Gerard Kempen. 1987. A framework for incremental syntactic tree formation. In *Proceedings, lOth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 655–660.

Gerard Kempen and Edward Hoenkamp. 1987. An incremental procedural grammar for sentence formulation. *Cognitive Science*, 11(2):201–258.

Ruth Kempson, Wilfried Meyer-Viol, and Dov Gabbay. 2001. *Dynamic Syntax: The Flow of Language Understanding*. Blackwell, Oxford.

Ruth Kempson, Eleni Gregoromichelaki, Wifried Meyer-Viol, Matthew Purver, Graham White, and Ronnie Cann. 2011. Natural-language syntax as procedures for interpretation: the dynamics of ellipsis construal. In A. Lecomte and S. Tronçon, editors, *Ludics, Dialogue and Interaction*, number 6505 in Lecture Notes in Computer Science, pages 114–133. Springer-Verlag, Berlin/Heidelberg.

Ruth Kempson, Ronnie Cann, Arash Eshghi, Eleni Gregoromichelaki, and Matthew Purver. forthcoming. Ellipsis. In S. Lappin and C. Fox, editors, *Handbook of Contemporary Semantic Theory*. Wiley, 2nd edition.

Casey Kennington and David Schlangen. 2014. Situated incremental natural language understanding using markov logic networks. *Computer Speech & Language*, 28(1):240–255.

Casey Kennington, Spyros Kousidis, and David Schlangen. 2014. Situated incremental natural language understanding using a multimodal, linguistically-driven update model. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1803–1812, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Kevin H Knuth. 2005. Lattice duality: The origin of probability and entropy. *Neurocomputing*, 67:245–274.

Kevin H Knuth. 2006. Valuations on lattices and their application to information theory. In *Fuzzy Systems, 2006 IEEE International Conference on*, pages 217–224. IEEE.

Spyros Kousidis, Casey Kennington, Timo Baumann, Hendrik Buschmeier, Stefan Kopp, and David Schlangen. 2014. Situationally aware in-car information presentation using incremental speech generation: Safer, and more effective. In *Proceedings of the EACL 2014 Workshop on Dialogue in Motion*, pages 68–72, Gothenburg, Sweden. Association for Computational Linguistics.

Emiel Krahmer and Kees Van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Staffan Larsson. 2002. *Issue-based Dialogue Management*. PhD thesis, Göteborg University. Also published as Gothenburg Monographs in Linguistics 21.

Staffan Larsson. 2010. Accommodating innovative meaning in dialogue. *Proc. of Londial, SemDial Workshop*, pages 83–90.

Staffan Larsson. 2011. The TTR perceptron: Dynamic perceptual meanings and semantic coordination. In *Proceedings of the 15th Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2011 - Los Angelogue)*, pages 140–148.

Jey Han Lau, Alexander Clark, and Shalom Lappin. 2014. Measuring gradience in speakers' grammaticality judgements. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society (CogSci 2014)*, pages 821–826.

Matthew Lease, Mark Johnson, and Eugene Charniak. 2006. Recognizing disfluencies in conversational speech. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(5): 1566–1573.

Ivan Leudar, Philip Thomas, and Margaret Johnston. 1992. Self-repair in dialogues of schizophrenics: effects of hallucinations and negative symptoms. *Brain and language*, 43 (3):487–511.

W.J.M. Levelt. 1983. Monitoring and self-repair in speech. *Cognition*, 14(1):41–104.

W.J.M. Levelt. 1989. *Speaking: From intention to articulation*. Mit Pr.

Stephen C. Levinson. 1983. *Pragmatics*. Cambridge Textbooks in Linguistics. Cambridge University Press.

Yang Liu. 2004. *Structural event detection for rich transcription of speech*. PhD thesis, Purdue University.

Yang Liu, Elizabeth Shriberg, and Andreas Stolcke. 2003. Automatic disfluency identification in conversational speech using multiple knowledge sources. In *Preceedings of Eurospeech*, pages 957–960.

Christopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Mitchell P. Marcus, Beatrice Santrorini, Mary Ann Marchinkiewicz, and Ann Taylor. 1999. Treebank 3. Technical report, Linguistic Data Consortium.

S. Maskey, B. Zhou, and Y. Gao. 2006. A phrase-level machine translation approach for disfluency detection using weighted finite state transducers. pages 749–752, Pittspurg, Pennsylvania.

R. McCabe, P. G. T. Healey, S. Priebe, M. Lavelle, D. Dodwell, R. Laugharne, A. Snell, and S. Bremner. 2013. Shared understanding in psychiatrist-patient communication: Association with treatment adherence in schizophrenia. *Patient Education and Counselling*.

David McDonald. 1987. Natural language generation. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence, Volume 1*, pages 642–654. New York.

David McKelvie. 1998. The syntax of disfluency in spontaneous spoken language. Technical report, Paper HCRC/RP-95, Human Communication Research Centre, Edinburgh.

M. Meteer, A. Taylor, R. MacIntyre, and R. Iyer. 1995. Disfluency annotation stylebook for the switchboard corpus. ms. Technical report, Department of Computer and Information Science, University of Pennsylvania. URL `ftp://ftp.cis.upenn.edu/pub/treebank/swbd/doc/DFL-book.ps`.

Margot Mieskes and Michael Strube. 2006. Part-of-speech tagging of transcribed speech. In *Proceedings of LREC*, pages 935–938.

Tim Miller and William Schuler. 2008. A syntactic time-series model for parsing fluent and disfluent speech. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 569–576. Association for Computational Linguistics.

D. Milward. 1995. Incremental interpretation of categorial grammar. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, pages 119–126. Morgan Kaufmann Publishers Inc.

David Milward. 1991. *Axiomatic Grammar, Non-Constituent Coordination and Incremental Interpretation*. PhD thesis, University of Cambridge.

Richard Montague. 1974. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press.

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48 (3):443–453.

Günter Neumann. 1994. *A Uniform Computational Model for Natural Language Parsing and Generation*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany.

Günter Neumann. 1998. Interleaving natural language parsing and generation through uniform processing. *Artificial Intelligence*, 99:121–163.

Günter Neumann and Wolfgang Finkler. 1990. A head-driven approach to incremental and parallel generation of syntactic structures. In *Proceedings of the 13th conference on Computational linguistics - Volume 2*, COLING '90, pages 288–293, Stroudsburg, PA, USA. Association for Computational Linguistics.

Masayuki Otsuka and Matthew Purver. 2003. Incremental generation by incremental parsing. In *Proceedings of the 6th CLUK Colloquium*, pages 93–100, Edinburgh. CLUK.

D.B. Paul and J.M. Baker. 1992. The design for the wall street journal-based csr corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics.

Andreas Peldszus, Okko Buß, Timo Baumann, and David Schlangen. 2012. Joint satisfaction of syntactic and pragmatic constraints improves incremental spoken language understanding. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 514–523, Avignon, France. Association for Computational Linguistics.

Martin Pickering and Simon Garrod. 2004. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, 27:169–226.

Massimo Poesio and Hannes Rieser. 2010. Completions, coordination, and alignment in dialogue. *Dialogue and Discourse*, 1:1–89.

Massimo Poesio and David Traum. 1997. Conversational actions and discourse situations. *Computational Intelligence*, 13(3).

Christopher Potts. 2011. The switchboard dialogue act corpus (educational distribution). Technical report, LSA Linguistic Institute 2011: Language in the World. University of Colorado, Boulder.

Matthew Purver. 2004. CLARIE: the Clarification Engine. In J. Ginzburg and E. Vallduví, editors, *Proceedings of the 8th Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL)*, pages 77–84, Barcelona, Spain.

Matthew Purver and Ruth Kempson. 2004. Incremental context-based generation for dialogue. In A. Belz, R. Evans, and P. Piwek, editors, *Proceedings of the 3rd International Conference on Natural Language Generation (INLG04)*, number 3123 in Lecture Notes in Artifical Intelligence, pages 151–160, Brockenhurst, UK. Springer.

Matthew Purver and Masayuki Otsuka. 2003. Incremental generation by incremental parsing: Tactical generation in Dynamic Syntax. In *Proceedings of the 9th European Workshop in Natural Language Generation (ENLG)*, pages 79–86.

Matthew Purver, Jonathan Ginzburg, and Patrick Healey. 2003. On the means for clarification in dialogue. In R. Smith and J. van Kuppevelt, editors, *Current and New Directions in Discourse & Dialogue*, pages 235–255. Kluwer Academic Publishers.

Matthew Purver, Ronnie Cann, and Ruth Kempson. 2006. Grammars as parsers: Meeting the dialogue challenge. *Research on Language and Computation*, 4(2-3):289–326.

Matthew Purver, Eleni Gregoromichelaki, Wilfried Meyer-Viol, and Ronnie Cann. 2010. Splitting the 'I's and crossing the 'You's: Context, speech acts and grammar. In P. Łupkowski and M. Purver, editors, *Aspects of Semantics and Pragmatics of Dialogue. SemDial 2010, 14th Workshop on the Semantics and Pragmatics of Dialogue*, pages 43–50, Poznań. Polish Society for Cognitive Science.

Matthew Purver, Arash Eshghi, and Julian Hough. 2011. Incremental semantic construction in a dialogue system. In J. Bos and S. Pulman, editors, *Proceedings of the 9th International Conference on Computational Semantics*, pages 365–369, Oxford, UK.

Matthew Purver, Julian Hough, and Eleni Gregoromichelaki. 2014. Dialogue and compound contributions. In A. Stent and S. Bangalore, editors, *Natural Language Generation in Interactive Systems*, pages 63–92. Cambridge University Press.

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *Proceedings of NAACL-HLT*, pages 820–825.

Mohammad Sadegh Rasooli and Joel Tetreault. 2014. Non-monotonic parsing of fluent umm I mean disfluent sentences. *EACL 2014*, pages 48–53.

Mohammad Sadegh Rasooli and Joel R Tetreault. 2013. Joint parsing and disfluency detection in linear time. In *EMNLP*, pages 124–129.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press.

Hannes Rieser and David Schlangen. 2011. Introduction to the special issue on incremental processing in dialogue. *Dialogue & Discourse*, 2(1):1–10.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Brian Roark, Asaf Bachrach, Carlos Cardenas, and Christophe Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 324–333. Association for Computational Linguistics.

Kepa Rodríguez and David Schlangen. 2004. Form, intonation and function of clarification requests in German task-oriented spoken dialogues. In *Proceedings of the 8th Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL)*, Barcelona, Spain.

Paula Rubio-Fernández. 2011. Colours colores. Invited talk at the 4th Biennial Experimental Pragmatics Conference.

Yo Sato. 2011. Local ambiguity, search strategies and parsing in Dynamic Syntax. In E. Gregoromichelaki, R. Kempson, and C. Howes, editors, *The Dynamics of Lexical Interfaces*, pages 205–233. CSLI.

E.A. Schegloff. 1992. Repair after next turn: The last structurally provided defense of intersubjectivity in conversation. *American Journal of Sociology*, pages 1295–1345.

E.A. Schegloff. 1997. Third turn repair. *Amsterdam Studies in the Theory and History of Linguistic Science Series 4*, pages 31–40.

Emanuel A. Schegloff, Gail Jefferson, and Harvey Sacks. 1977. The preference for self-correction in the organization of repair in conversation. *Language*, 53(2):361–382.

David Schlangen. 2009. What we can learn from dialogue systems that dont work. In *Proceedings of DiaHolmia (Semdial 2009*, pages 51–58.

David Schlangen and Gabriel Skantze. 2009. A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 710–718, Athens, Greece. Association for Computational Linguistics.

David Schlangen and Gabriel Skantze. 2011. A general, abstract model of incremental dialogue processing. *Dialogue and Discourse*, 2(1):83–111.

Claude E. Shannon. 1948. A mathematical theory of communication. technical journal. *AT & T Bell Labs*.

Stuart Shieber. 1988. A uniform architecture for parsing and generation. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, pages 614–619.

Stuart Shieber. 1993. The problem of logical-form equivalence. *Computational Linguistics*, 19 (1):179–190.

Stuart M Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th conference on Computational linguistics-Volume 3*, pages 253–258. Association for Computational Linguistics.

E. Shriberg, R. Bates, A. Stolcke, P. Taylor, D. Jurafsky, K. Ries, N. Coccaro, R. Martin, M. Meteer, and C. van Ess-Dykema. 1998. Can prosody aid the automatic classification of dialog acts in conversational speech? *Language and Speech*, 41:443–492.

Elizabeth Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. PhD thesis, University of California, Berkeley.

Elizabeth Shriberg. 1996. Disfluencies in switchboard. In *In Proceedings of the International Conference on Spoken Language Processing*, volume 96, pages 3–6. Citeseer.

Elizabeth Shriberg and Andreas Stolcke. 1998. How far do speakers back up in repairs? A quantitative model. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2183–2186.

Gabriel Skantze. 2014. Gabriel Skantze's research webpage. `http://www.speech.kth.se/~gabriel/research.html` Accessed: 2014-09-30.

Gabriel Skantze and Anna Hjalmarsson. 2010. Towards incremental speech generation in dialogue systems. In *Proceedings of the SIGDIAL 2010 Conference*, pages 1–8, Tokyo, Japan. Association for Computational Linguistics.

Gabriel Skantze and David Schlangen. 2009. Incremental dialogue processing in a micro-domain. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 745–753, Athens, Greece. Association for Computational Linguistics.

M. Snover, B. Dorr, and R. Schwartz. 2004. A lexically-driven algorithm for disfluency detection. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 157–160. Association for Computational Linguistics.

Dan Sperber and Deirdre Wilson. 1986. *Relevance: Communication and Cognition*. Blackwell.

Raman Srinivas Sundaresh and Paul Hudak. 1991. A theory of incremental computation and its application. In *Proceedings of the 18th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 1–13. ACM.

Henry Thompson. 1977. Strategy and tactics: A model for language production. In *Papers from the 13th regional meeting of the Chicago Linguistic Society*, pages 651–668.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*, pages 252–259.

K. Van Deemter and M.M. Halldórsson. 2001. Logical form equivalence: The case of referring expressions generation. In *Proceedings of the 8th European workshop on Natural Language Generation-Volume 8*, pages 1–8. Association for Computational Linguistics.

Carel van Wijk and Gerard Kempen. 1987. A dual system for producing self-repairs in spontaneous speech: Evidence from experimentally elicited corrections. *Cognitive Psychology*, 19 (4):403 – 440.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Jason Williams and Steve Young. 2007. Scaling POMDPs for spoken dialog management. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2116–2129.

Sina Zarrieß and Jonas Kuhn. 2013. Combining referring expression generation and surface realization: A corpus-based investigation of architectures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1547–1557, Sofia, Bulgaria. Association for Computational Linguistics.

Simon Zwarts and Mark Johnson. 2011. The impact of language models and loss functions on repair disfluency detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 703–711, Stroudsburg, PA, USA. Association for Computational Linguistics.

Simon Zwarts, Mark Johnson, and Robert Dale. 2010. Detecting speech repairs incrementally using a noisy channel approach. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1371–1378, Stroudsburg, PA, USA. Association for Computational Linguistics.

Appendix A

Feature Ranking

```
RM start detection features:
average merit      average rank  attribute
 0.516 +- 0.004      1   +- 0     WMLboostDrop
 0.505 +- 0.004      2   +- 0     POSWMLboostDrop
 0.485 +- 0.012      3.3 +- 0.46  POSKL(P(X|rms-1,rms),P(X|rms-1,rps))
 0.483 +- 0.004      3.7 +- 0.46  DistanceBackToRPStart
 0.417 +- 0.004      5   +- 0     rmsW3=rpsW3
 0.384 +- 0.005      6.4 +- 0.49  POSrmsW3=POSrpsW3
 0.381 +- 0.005      6.6 +- 0.49  rmsW3=<rps>(embed)
 0.258 +- 0.009      8.1 +- 0.3   POSEntropy
 0.245 +- 0.005      8.9 +- 0.3   WML(rms-2,rms-1,rps)
 0.226 +- 0.005     10.3 +- 0.46  LocalSurprisalBoost
 0.221 +- 0.004     11.6 +- 1.02  POSWML(rms-2,rms-1,rps)
 0.219 +- 0.008     12   +- 1.26  LocalWMLboost
 0.213 +- 0.007     13.3 +- 1.79  POSLocalSurprisalBoost
 0.211 +- 0.004     13.9 +- 1.14  WMLboost
 0.21  +- 0.003     14.4 +- 0.8   POSWMLboost
 0.202 +- 0.004     16.4 +- 1.36  POSSurprisal(rms-2,rms-1,rps)
 0.201 +- 0.007     17   +- 1.48  POSSurprisalBoost
 0.193 +- 0.009     19   +- 1.48  POSRMstartEntropy
 0.192 +- 0.007     19.1 +- 1.64  Surprisal(rms-2,rms-1,rps)
 0.19  +- 0.009     19.6 +- 1.56  POSLocalWMLboost
 0.189 +- 0.004     19.7 +- 0.9   lengthIntoUtt
 0.176 +- 0.005     21.7 +- 0.9   POSEntropyReduce
 0.159 +- 0.006     23.1 +- 0.3   SurprisalBoost
 0.146 +- 0.008     24.4 +- 0.92  WordEntropy
 0.142 +- 0.005     24.8 +- 0.6   POSSurprisal(rms-2,rms-1,rms)
 0.136 +- 0.005     25.7 +- 0.46  POSWML(rms-2,rms-1,rms)
 0.113 +- 0.008     27   +- 0     WordEntropyReduce
 0.081 +- 0.007     28.6 +- 0.66  WordRMPEntropy
 0.079 +- 0.004     28.6 +- 0.49  Surprisal(rms-2,rms-1,rms)
 0.062 +- 0.008     29.8 +- 0.6   WML(rms-2,rms-1,rms)
 0.028 +- 0.001     31   +- 0     rmsW2=<rps>(embed)
 0.021 +- 0.001     32   +- 0     rp1confidence
 0     +- 0         33   +- 0     edit_rps
```

```
RP end detection features:
average merit      average rank  attribute
0.925 +- 0.001      1   +- 0     repairLength
0.381 +- 0.002      2.2 +- 0.4   embeddedRPS
0.376 +- 0.008      3   +- 0.63  POSReparandumRepairDiff
0.364 +- 0.002      3.8 +- 0.4   ReparandumRepairDiff
0.294 +- 0.006      5   +- 0     POSKL(P(X|RMn-1,RMn),P(X|RPn-1,RPn))
0.252 +- 0.003      6   +- 0     POSReparandum=Repair
0.247 +- 0.003      7   +- 0     RMend=RPend
0.237 +- 0.003      8   +- 0     Reparandum=Repair
0.207 +- 0.015      9.3 +- 0.46  POSLocalSurprisalBoost
0.203 +- 0.003      9.7 +- 0.46  POS_RMend=RPend
0.167 +- 0.013     11.4 +- 0.8   POSLocalWMLBoost
0.159 +- 0.002     12   +- 0.45  ReparandumLength
0.157 +- 0.003     12.9 +- 0.54  RPsClassifierConfidence
0.136 +- 0.007     14.4 +- 0.66  LocalWMLBoost
0.125 +- 0.008     15   +- 0.45  LocalSurprisalBoost
0.091 +- 0.034     15.3 +- 1.55  POSInfoGainReduce
0.06  +- 0.021     17.5 +- 0.67  InfoGainReduce
0.051 +- 0.008     17.9 +- 0.83  POSKL(P(X|RMs-1,RMs),P(X|RPs-1,RPs))
0.035 +- 0.002     19   +- 0.63  RMsClassifierConfidence
0.017 +- 0.001     20.4 +- 0.49  RMstart=RPstart
0.025 +- 0.02      20.9 +- 1.7   Entropy(P(X|RMs-1,RMs))
0.012 +- 0         21.5 +- 0.5   POSRMstart=RPstart
0.008 +- 0.003     22.8 +- 0.6   POSEntropy(P(X|RMs-1,RMs))
```

Appendix B

Dynamic Syntax computational actions

- All actions are tree construction actions IF..THEN..ELSE... where the IF preconditions predicate on the current pointer position in the tree and THEN..ELSE.. effects are given relative to the current pointer position.
- Possible tree types are standard (matrix) trees, Linked trees and unfixed nodes (starred nodes).
- Tree decorations are $Tn(x)$ (node addresses), $Ty(X)$ (node types), $Fo(x)$ (semantic formulae), $CLASS(x)$ and $PERSON(x)$ (subtpyes of semantic formulae with values indicating gender and plurality values respectively), and $+eval$ (Link evaluated node if a Linked tree stems from it). All decorations can be prefixed ? to make them requirements (meaning they require these decorations rather than have them).
- Basic tree building and pointer movement operations: `go`(*node/tree modality*), `make`(*node*), `put`(*tree decoration at current node*), `delete`(*tree decoration at current node*) and `do`(*action*)
- Tree modalities:$\langle\uparrow\rangle$ (above pointer), $\langle\downarrow\rangle$ (below pointer), $\langle Y\rangle$ (some modality Y). All modalities can be specified as being in the direction of a specific node type $1, 0, Link$ or $*$ where 1 is a functor node (right daughter) and 0 is an argument node (left daughter), *Link* is a Linked tree and $*$ is an unfixed node (star node).
- *ContextTree* is the last tree processed before the current tree.
- $TriggeredBy(X, A)$ is where X is a tree element and A is the action that constructed it.
- Rules with names prefixed with * are non-optional.
- Rules with names prefixed with + are applied repeatedly, exhausting all successful rule metavariable combinations.
- Node type and tree modality metavariables: X, Y, Z
- Formula (including *CLASS* and *PERSON*) and node metavariables: v, w, x, y, z
- Rule metavariables: A, B, C

INTRODUCTION PREDICTION

IF	$?Ty(t)$
	$\neg\langle\downarrow 1\rangle\exists x.x$
	$\neg\langle\downarrow 0\rangle\exists x.x$
THEN	make($\downarrow 1$)
	go($\downarrow 1$)
	put($?Ty(e \rightarrow t)$)
	go($\uparrow 1$)
	make($\downarrow 0$)
	go($\downarrow 0$)
	put($?Ty(e)$)
ELSE	abort

ANTICIPATION $\downarrow 0$

IF	$\langle\downarrow 0\rangle\exists x.?x$
THEN	go($\downarrow 0$)
ELSE	abort

ANTICIPATION $\downarrow 1$

IF	$\langle\downarrow 1\rangle\exists x.?x$
THEN	go($\downarrow 1$)
ELSE	abort

ANTICIPATION LINK

IF	$Ty(t) \vee Ty(e \rightarrow t)$
	$\langle\downarrow Link\rangle\exists x.x$
THEN	go($\downarrow Link$)
ELSE	abort

*THINNING

IF	$?X$
	X
THEN	delete($?X$)
ELSE	abort

COMPLETION

IF	$Ty(X)$
	$\neg?Ty(X)$
	$\langle\uparrow\rangle\exists x.x$
THEN	go($\uparrow$)
ELSE	abort

*ELIMINATION

IF	$\neg?Ty(X)$
	$\neg\exists x.Fo(x)$
	$\langle\downarrow 1\rangle Ty(Y \rightarrow X)$
	$\langle\downarrow 0\rangle Ty(Y)$
	$\langle\downarrow 1\rangle Fo(y)$
	$\langle\downarrow 0\rangle Fo(z)$
	$\neg\langle\downarrow 1\rangle\exists x.?x$
	$\neg\langle\downarrow 0\rangle\exists x.?x$
THEN	put($(Fo(y))Fo(z)$) (β-reduce)
ELSE	abort

STAR ADJUNCTION

IF	$Ty(X)$
	$\neg ?Ty(X)$
	$\langle\uparrow\rangle\exists x.x$
	$\neg ?\langle\downarrow 1\rangle Ty(e \rightarrow t)$
	$\neg ?\langle\downarrow 0\rangle Ty(e)$
	$\neg\langle\downarrow 1\rangle\exists x.x$
	$\neg\langle\downarrow 0\rangle\exists x.x$
	$\neg\langle\downarrow *\rangle\exists x.x$
THEN	make($\downarrow *$)
	go($\downarrow *$)
	put($?Ty(e)$)
	put($?\exists x.Tn(x)$)
ELSE	abort

LINK ADJUNCTION

IF	$Ty(e)$
	$Fo(x)$
	$\neg\langle\downarrow Link\rangle\exists x.x$
THEN	make($\downarrow Link$)
	go($\downarrow Link$)
	put($?Ty(t)$)
	put($?\langle\downarrow *\rangle Fo(x)$)
	put($?+eval$)
ELSE	abort

MERGE

IF	$Tn(x)$
	$\langle Y\rangle ?Tn(y)$
THEN	$Tn(x) = Tn(x) \cup Tn(y)$ (Conjoin nodes)
ELSE	abort

*LINK EVALUATION

IF	$?+eval$
	$Fo(x)$
	$\langle\downarrow Link\rangle Fo(y)$
THEN	put($+eval$)
	$Fo(y)$ $\boxed{\wedge}$ $Fo(x)$
ELSE	abort

LOCAL SUBSTITUTION (pronouns only)

IF	$Ty(X)$
	$?\exists x.Fo(x)$
	$CLASS(v)$
	$PERSON(w)$
	$\langle Y\rangle Ty(X)$
	$\langle Y\rangle Fo(z)$
	$\langle Y\rangle CLASS(v)$
	$\langle Y\rangle PERSON(w)$
	$\neg\langle\uparrow 0\uparrow *\downarrow 0\rangle Fo(z)$
THEN	put($Fo(z)$)
ELSE	abort

CONTEXT SUBSTITUTION PRONOUN

IF	$Ty(e)$
	$?\exists x.Fo(x)$
	$CLASS(y)$
	$PERSON(z)$
	$Ty(e) \wedge CLASS(y) \wedge PERSON(z) \wedge Fo(v) \in ContextTree$
THEN	put($Fo(v)$)
ELSE	abort

CONTEXT SUBSTITUTION OTHER

IF	$\neg Ty(e)$
	$Ty(X)$
	$?\exists x.Fo(x)$
	$Ty(X) \wedge Fo(y) \in$ ContextTree
THEN	put($Fo(y)$)
ELSE	abort

REGENERATION

IF	$Ty(X)$
	$?\exists x.Fo(x)$
	$TriggeredBy(Ty(X),A)$
THEN	do(A)
ELSE	abort