Cops and CoCoWeb
Infrastructure for Confluence Tools*

Nao Hirokawa[1], Julian Nagele[2], and Aart Middeldorp[3]

[1] hirokawa@jaist.ac.jp
School of Information Science, JAIST, Japan
[2] j.nagele@qmul.ac.uk
School of Electronic Engineering and Computer Science
Queen Mary University of London, UK
[3] aart.middeldorp@uibk.ac.at
Department of Computer Science, University of Innsbruck, Austria

Abstract. In this paper we describe the infrastructure supporting confluence tools and competitions: Cops, the confluence problems database, and CoCoWeb, a convenient web interface for tools that participate in the annual confluence competition.

1 Introduction

In recent years several tools have been developed to automatically prove confluence and related properties of a variety of rewrite formats. These tools compete annually in the confluence competition [1] (CoCo).[4] Confluence tools run on confluence problems which are organized in the confluence problems (Cops) database. Cops is managed via a web interface

http://cops.uibk.ac.at/

that comes equipped with a useful tagging system. Cops has recently been revamped and we describe its unique features in this paper.

Most of the tools that participate in CoCo can be downloaded, installed, and run on one's local machine, but this can be a painful process.[5] Only few confluence tools—we are aware of CO3 [8], ConCon [11], and CSI [7,14]—provide a convenient web interface to painlessly test the status of a confluence problem that is provided by the user. In this paper we present CoCoWeb, a web interface to execute confluence tools on confluence problems. This provides a single entry point to all tools that participate in CoCo. CoCoWeb is available at

http://cocoweb.uibk.ac.at/

* This research is supported by FWF (Austrian Science Fund) project P27528, JSPS Core to Core Program, and JSPS KAKENHI Grant Nos. 25730004 and 17K00011.

[4] http://coco.nue.ie.niigata-u.ac.jp/

[5] StarExec provides a VM image with their environment, which can be helpful in case a local setup is essential.

The typical use of CoCoWeb is to test whether a given confluence problem is known to be confluent or not. This is useful when preparing or reviewing an article, preparing or correcting exams about term rewriting, and when contemplating submitting a challenging problem to Cops. In particular, CoCoWeb is useful when crafting or looking for examples to illustrate a new technique. For instance, in [4] a rewrite system is presented that can be shown to be confluent with the technique introduced in that paper. The authors write "Note that we have tried to show confluence [...] by confluence checker ACP and Saigawa, and both of them failed." Despite having an easy to use web interface, CSI was not tried. CoCoWeb could also be useful for the CoCo steering committee when integrating newly submitted problems into Cops and also when investigating dubious answers of confluence tools.

The remainder of the paper is organized as follows. In the next section we describe the functionality of Cops, and in Section 3 we present the web interface of CoCoWeb by means of a number of screenshots. Both sections contain a few implementation details as well. In Section 4 we list some possibilities for extending the functionality of Cops and CoCoWeb in the future.

2 Cops: Confluence Problems Database

Cops is an online database for confluence problems in term rewriting. It was created in 2012 to facilitate the organization of CoCo and development of confluence tools. Via its web interface, everyone can retrieve and download confluence problems, and also upload new problems. Uploaded problems are reviewed by the CoCo steering committee and then integrated into Cops. Figure 1 shows the web interface of Cops. Below, we explain basic features of the interface. The interface is designed in a way that novice users can easily learn problem formats, and at the same time experts on confluence can retrieve a problem set for their experiments.

Problems While there are several variations of rewrite systems, Cops comprises the following five rewriting formats: ordinary term rewrite systems (TRS), extended term rewrite systems (eTRS) that do not impose the variable conditions of TRS, conditional term rewrite systems (CTRS), higher-order term rewrite systems (HRS), and many-sorted term rewrite systems (MSTRS). In the database, confluence problems are maintained as text files, and identification numbers are assigned to them. Currently, Cops contains 765 systems and more than half of them have been collected from the literature.

The main box in Figure 1 shows confluence problem number 1 (`1.trs`), which consists of five rewrite rules. To increase readability, Cops supports syntax highlighting for the above five formats. By clicking the hyperlinked number in brackets in the comment field, the source of the problem with a corresponding BibTeX entry is displayed. Typically the comment field also includes the name of the person(s) who submitted the problem. This is to acknowledge the

Fig. 1. The web interface of Cops.

effort involved in locating interesting problems and making these available to the community.

Tags Cops has no directory structure. Instead, *tags*—which can be seen as a multi-dimensional directory structure—are assigned to problems. Different kinds of tags are supported. On the one hand, properties of rewrite systems like left-linearity, groundness, and termination are useful to filter the database for those problems that are supported by a particular tool or technique. These include the tags to distinguish the four different input formats, and they are automatically computed when problems are submitted. A second category of tags refers to tools that could solve the problem (i.e., prove or disprove confluence) in earlier confluence competitions. An example is `acp2017` which is assigned to all problems selected for CoCo 2017 that were solved by ACP [2].

Finally there is the `literature` tag that is assigned to problems that appear in the literature, which includes papers presented at informal workshops like the International Workshop on Confluence and PhD theses. This tag is important since CoCo uses problems from the literature, rather than generated problems that are biased towards one particular tool or technique.

The data of Cops consists of confluence problems and tags. Every tag file is a list of problem numbers in text format. Most of the tag files are generated automatically or updated by a collection of scripts that call external tools. The current collection includes tools to check syntactical properties like left-linearity or right-groundness, ConCon [11] for tags that are specific to CTRSs, and T$_\text{T}$T$_2$ [5] for checking termination and non-termination of TRSs. Since some properties (e.g. termination) are undecidable, tags like `non_terminating` also exist. In addition, Felgenhauer's duplication checker for TRSs (modulo symbol renaming) is included.[6] Duplication is not desirable for fair evaluations. The tag "`duplicate`" is assigned to such a problem.

Queries Problems can be filtered by typing queries in the search box. Queries are specified by Boolean combinations of tags and problem numbers:

$$\phi \ ::= \ \mathit{tag} \mid \mathit{number} \mid \texttt{!}\ \phi \mid \phi\ \phi \mid \phi\ \texttt{OR}\ \phi \mid \{\phi\}$$

Conjunction is denoted by juxtaposition and negation by an exclamation mark. For instance, the query "`left_linear trs`" yields all problems with the two tags `left_linear` and `trs`. In order to search for non-left-linear TRSs whose termination is not known "`!{left_linear OR terminating} trs`" is used. This functionality is also useful for comparing tools. The query "`csi2017 !acp2017`" shows all confluence problems that were solved by CSI but not by ACP in CoCo 2017. It is worth noting that advisory board members of CoCo exploit the tag-based queries (besides random seeds) to compile problem sets used for the live competitions of CoCo. Problems resulting from search queries can be downloaded as a zip file. Optionally, tag files and BibTeX files are included too.

The search engine of Cops consists of only 235 lines of Ruby code. This is implemented as a command-line tool and bundled with a problem set when the aforementioned download option is selected. The tool name is `cops` and one can run it in a Unix environment. For example, the command

```
./cops 'oriented deterministic 3_ctrs'
```

outputs all problem numbers of oriented deterministic 3-CTRSs in the downloaded problem set. The web interface is built on it. The corresponding code is about 5,000 lines of PHP, Ruby, and JavaScript code. Syntax highlighting in the submission page has been implemented on the top of CodeMirror.[7] Finally, BibTeX2HTML[8] is used for generating HTML for the references.

3 CoCoWeb: Web Interface for Confluence Tools

CoCoWeb is a web service to access confluence tools in a web browser. Figure 2 shows a screenshot of the entry page of CoCoWeb. Problems can be entered in three different ways:

[6] https://github.com/haskell-rewriting/canonical-trs
[7] https://codemirror.net/
[8] https://www.lri.fr/~filliatr/bibtex2html/

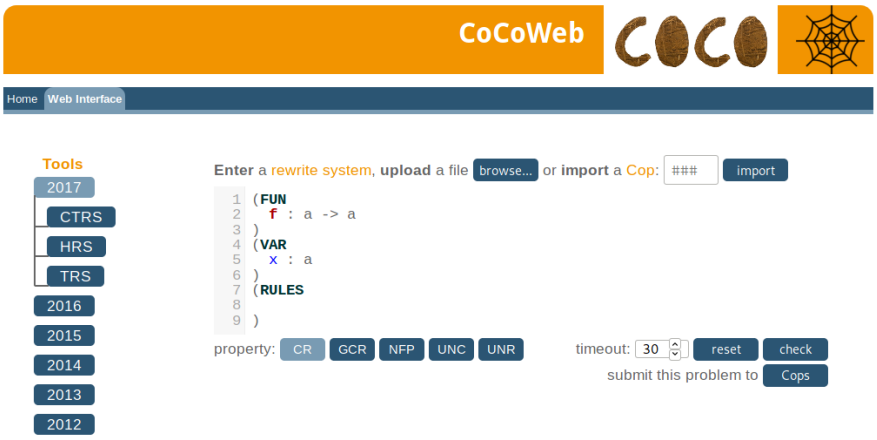

Fig. 2. The entry page of CoCoWeb.

1. using the text box,
2. uploading a file,
3. entering the number of a problem in Cops.

The problem can be submitted to Cops via the submit button. The tools that should be executed can be selected from the tools panel on the left. Tools are grouped into categories, similar to the grouping in CoCo. Multiple tools can be selected. This is illustrated in Figure 3. Here we selected CR as property, the CoCo 2016 and 2015 versions of ACPH [9] and the CoCo 2015 version of CSI^ho [6], and Cop 500 is chosen is input problem.

The final screenshot (in Figure 4) shows the output of CoCoWeb after clicking the check button. The output of the selected tools is presented in separate tabs. The colors of these tabs reveal useful information: Green means that the tool answered yes, red (not shown) means that the tool answered no, and a maybe answer or a timeout is shown in blue. By clicking on a tab, the color is made lighter and the output of the tool is presented. The final line of the output is timing information provided by CoCoWeb.

The tools in CoCoWeb run on hardware compatible with a single node of StarExec [12] that is used for CoCo, allowing for a proper comparison of tools. By specializing the service to confluence, CoCoWeb offers easy access to all tools that participated in CoCo without requiring users to register first, and immediate scheduling of executions as well as syntax highlighting.

Most of CoCoWeb is built using PHP. User input in forms, i.e., rewrite systems and tool selections, is sent using the HTTP POST method. The dynamic parts of the website, namely folding and unfolding in the tool selection menu and the tabs used for viewing tool output are implemented using JavaScript. To layout the tool selection menu we made extensive use of CSS3 selectors. For instance, the buttons to select tools are implemented as checkboxes with labels that are styled according to whether the checkbox is ticked or not:

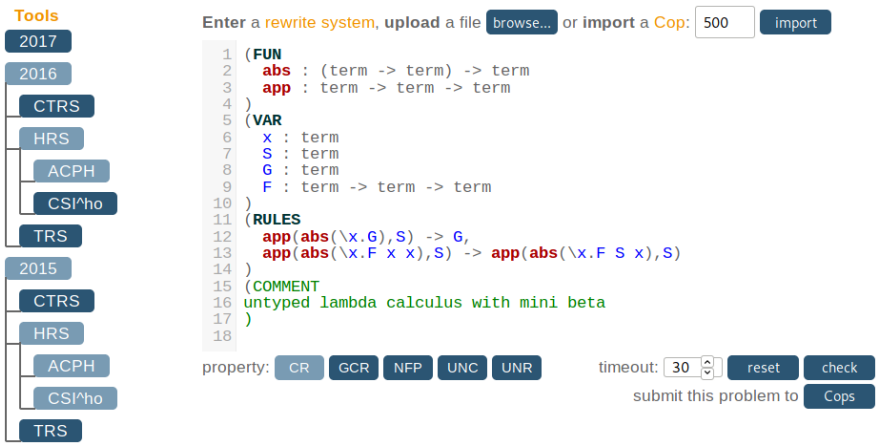

Fig. 3. Problem and tool selection in CoCoWeb.

```
.tools input[type="checkbox"]:checked + label
   {  color: white;  background-color: #799BB3;  }
```

Drawing the edges of the tree menu is also done using CSS, relying mainly on its `::before` selector.

Since its second edition, CoCo has adopted StarExec as competition platform. Competition participants upload binary executables of their tools together with necessary files to StarExec. Importing and complementing them with missing software, we reproduce the competition versions of tools on the CoCoWeb server. The collection of tools is maintained and associated with the web interface with help of small scripts. The content of the tool menu, i.e., years, the grouping by categories, and the actual tools, is generated automatically from a directory tree that has the structure of the menu in CoCoWeb. The directories contain small configuration files that specify how the tools are to be run, in case they are selected. Two environment variables are set in such a file, for example the one for the 2012 version of Saigawa [3] reads as follows:

```
TOOLDIR="Saigawa-2012/bin"
TOOL="./starexec_run_saigawa -t $TO $FILE"
```

The variable `TOOLDIR` specifies the directory that contains the tool binary, while `TOOL` gives the tool invocation, which in turn refers to `TO`, the timeout, and `FILE`, the input rewrite system. Using such configuration files tools are run using the following script, whose first, second, and third argument are the configuration of the tool, input rewrite system, and timeout respectively:

```
DIR=$(pwd -P)
FILE=$(readlink -f $2)
TO=$3;  TOT=$((TO + 2));  TOK=$((TOT + 2))
source $1
```

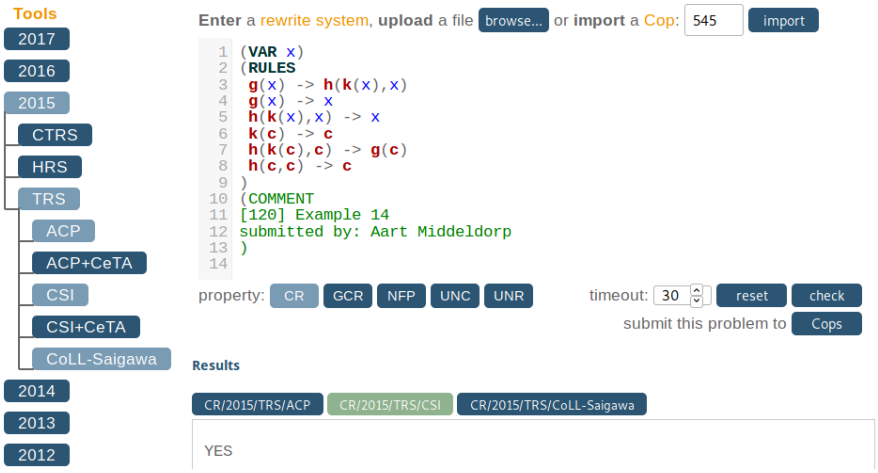

Fig. 4. Testing Example 14 from [4] in CoCoWeb.

```
pushd $DIR/bin/$TOOLDIR > /dev/null
/usr/bin/time -f "\\nTook %es" timeout -k $TOK $TOT $TOOL
popd > /dev/null
```

The script uses three different timeouts: `TO` is the timeout passed to the tool itself if supported, while after `TOT` and `TOK` the signals `SIGTERM` and `SIGKILL` are sent to the tool, using GNU coreutils timeout, in case it did not terminate on its own.[9] When multiple tools are selected, CoCoWeb runs them sequentially, in order to avoid interference.

4 Conclusion

In this paper we introduced Cops and CoCoWeb, two convenient systems that provide support for researchers that are interested in (developing tools for) confluence and related properties of rewrite systems. We believe the developed infrastructure could be useful for other competitions besides CoCo.

Both systems can be extended in several ways, which we plan to address in future work. For Cops, we are mainly concerned with two issues. One is about the reorganization of tags. Every year CoCo extends its scope to capture emerging trends, causing some tags to be redefined or renamed. Another is about reproducibility of search queries, which is crucial as Cops has been used as a standard benchmark for confluence techniques. To address these issues, we are seeking for a way to support versioning the database.

For CoCoWeb, preselection of tools based on the input problem would be a nice feature. This is not as trivial as it sounds, since different properties share

[9] To account for timing imprecisions and tools performing internal cleanup, 2 extra seconds are granted to the tool before sending `SIGTERM` and another 2 before `SIGKILL` is sent, which turned out to work well in practice.

the same problem format. We plan to investigate the selection method for ATP systems [13]. Supporting pretty-printing for XML output is another future task. While several tools support output of (non-)confluence proofs in the Certification Problem Format [10], the current web interface just displays the raw XML code.

Acknowledgments. We thank Harald Zankl, Christian Nemeth, and Takahito Aoto for their involvement in CoCo and the first release of Cops. Suggestions by the former helped to improve the paper.

References

1. Aoto, T., Hirokawa, N., Nagele, J., Nishida, N., Zankl, H.: Confluence Competition 2015. In: Proc. 25th CADE. LNAI, vol. 9195, pp. 101–104 (2015), doi: 10.1007/978-3-319-21401-6_5
2. Aoto, T., Yoshida, J., Toyama, Y.: Proving confluence of term rewriting systems automatically. In: Proc. 20th RTA. LNCS, vol. 5595, pp. 93–102 (2009), doi: 10.1007/978-3-642-02348-4_7
3. Hirokawa, N., Klein, D.: Saigawa: A confluence tool. In: Proc. 1st IWC. p. 49 (2012), available from http://cl-informatik.uibk.ac.at/iwc/iwc2012.pdf
4. Ishizuki, S., Oyamaguchi, M., Sakai, M.: Conditions for confluence of innermost terminating term rewriting systems. In: Proc. 5th IWC. pp. 70–74 (2016), available from http://cl-informatik.uibk.ac.at/iwc/iwc2016.pdf
5. Korp, M., Sternagel, C., Zankl, H., Middeldorp, A.: Tyrolean Termination Tool 2. In: Proc. 20th RTA. LNCS, vol. 5595, pp. 295–304 (2009), doi: 10.1007/978-3-642-02348-4_21
6. Nagele, J.: CoCo 2015 participant: CSIˆho 0.1. In: Proc. 4th IWC. p. 47 (2015), available from http://cl-informatik.uibk.ac.at/iwc/iwc2015.pdf
7. Nagele, J., Felgenhauer, B., Middeldorp, A.: CSI: New evidence – a progress report. In: Proc. 26th CADE. LNAI, vol. 10395, pp. 385–397 (2017), doi: 10.1007/978-3-319-63046-5_24
8. Nishida, N., Kuroda, T., Yanagisawa, M., Gmeiner, K.: CO3: A COnverter for proving COnfluence of COnditional TRSs (version 1.2). In: Proc. 4th IWC. p. 42 (2015), available from http://cl-informatik.uibk.ac.at/iwc/iwc2015.pdf
9. Onozawa, K., Kikuchi, K., Aoto, T., Toyama, Y.: ACPH: System description for CoCo 2016. In: Proc. 5th IWC. p. 76 (2016), available from http://cl-informatik.uibk.ac.at/iwc/iwc2016.pdf
10. Sternagel, C., Thiemann, R.: The certification problem format. In: Proc. 11th UITP. EPTCS, vol. 167, pp. 61–72 (2014), doi: 10.4204/EPTCS.167.8
11. Sternagel, T., Middeldorp, A.: Conditional confluence (system description). In: Proc. Joint 25th RTA and 12th TLCA. LNCS (ARCoSS), vol. 8560, pp. 456–465 (2014), doi: 10.1007/978-3-319-08918-8_31
12. Stump, A., Sutcliffe, G., Tinelli, C.: StarExec: A cross-community infrastructure for logic solving. In: Proc. 7th IJCAR. LNAI, vol. 8562, pp. 367–373 (2014), doi: 10.1007/978-3-319-08587-6_28
13. Sutcliffe, G., Suttner, C.: Evaluating general purpose automated theorem proving systems. Artificial Intelligence 131(1), 39–54 (2001), doi: 10.1016/S0004-3702(01)00113-8
14. Zankl, H., Felgenhauer, B., Middeldorp, A.: CSI – A confluence tool. In: Proc. 23rd CADE. LNAI, vol. 6803, pp. 499–505 (2011), doi: 10.1007/978-3-642-22438-6_38