CoCoWeb

A Convenient Web Interface for Confluence Tools*

Julian Nagele and Aart Middeldorp

Department of Computer Science, University of Innsbruck, Austria
{julian.nagele,aart.middeldorp}@uibk.ac.at

Abstract

We present a useful 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).[1] Most of the tools can be downloaded, installed, and run on one's local machine, but this can be a painful process.[2] Few confluence tools—we are aware of CO3 [4], ConCon [5], and CSI [3, 7]—provide a convenient web interface to painlessly test the status of a system that is provided by the user.

Inspired by the latest web interface of CSI [3], in this note we present CoCoWeb, a web interface that provides a single entry point to all tools that participate in CoCo. CoCoWeb is available at

http://cl-informatik.uibk.ac.at/software/cocoweb

The typical use of CoCoWeb will be 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 to submit a challenging problem to the confluence problems (Cops)[3] database. In particular, CoCoWeb is useful is when looking for (killer) examples to illustrate a new technique. For instance, in [2] 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.

The tools run on the same hardware, which is compatible with a single node of StarExec [6] that is used for CoCo, allowing for a proper comparison of tools.

In the next section we present the web interface of CoCoWeb by means of a number of screenshots. Implementation details are presented in Section 3 and in Section 4 we list some possibilities for extending the functionality of CoCoWeb in the future.

*This work is supported by the Austrian Science Fund (FWF): project P27528.

[1] http://coco.nue.riec.tohoku.ac.jp

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

[3] http://cops.uibk.ac.at

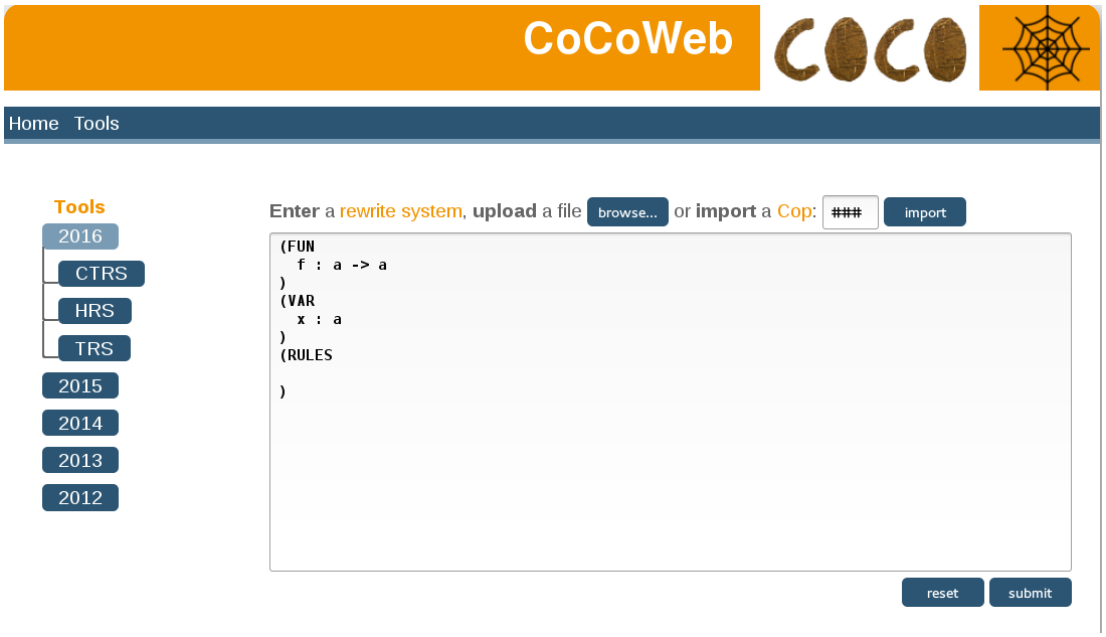

Figure 1: The entry page of CoCoWeb.

2 Web Interface

Figure 1 shows a screenshot of the entry page of CoCoWeb. Problems can be entered in three different ways: (1) using the text box, (2) uploading a file, (3) entering the number of a system in Cops. 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 except that we merged the

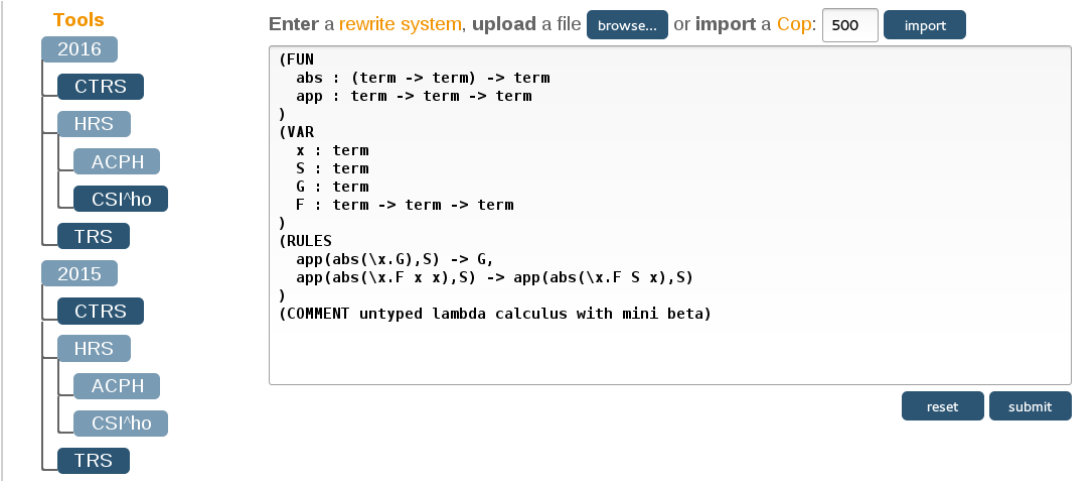

Figure 2: Problem and tool selection in CoCoWeb.

certified categories with the corresponding uncertified categories. Multiple tools can be selected. This is illustrated in Figure 2. Here we selected the CoCo 2016 and 2015 versions of ACPH and the CoCo 2015 version of CSIˆho, and Cop 500 is chosen is input problem.

The screenshot in Figure 3 shows the output of CoCoWeb after clicking the submit 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 final screenshot (in Figure 4) is concerned with the example in [2] that we referred to in the introduction.

3 Implementation

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:

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

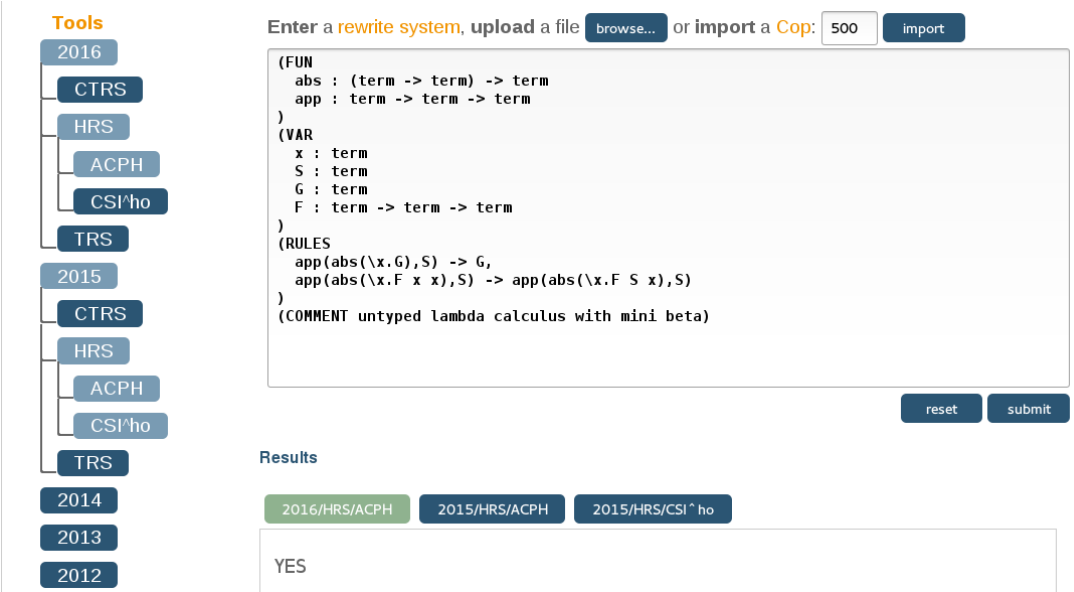

Figure 3: Result displaying in CoCoWeb.

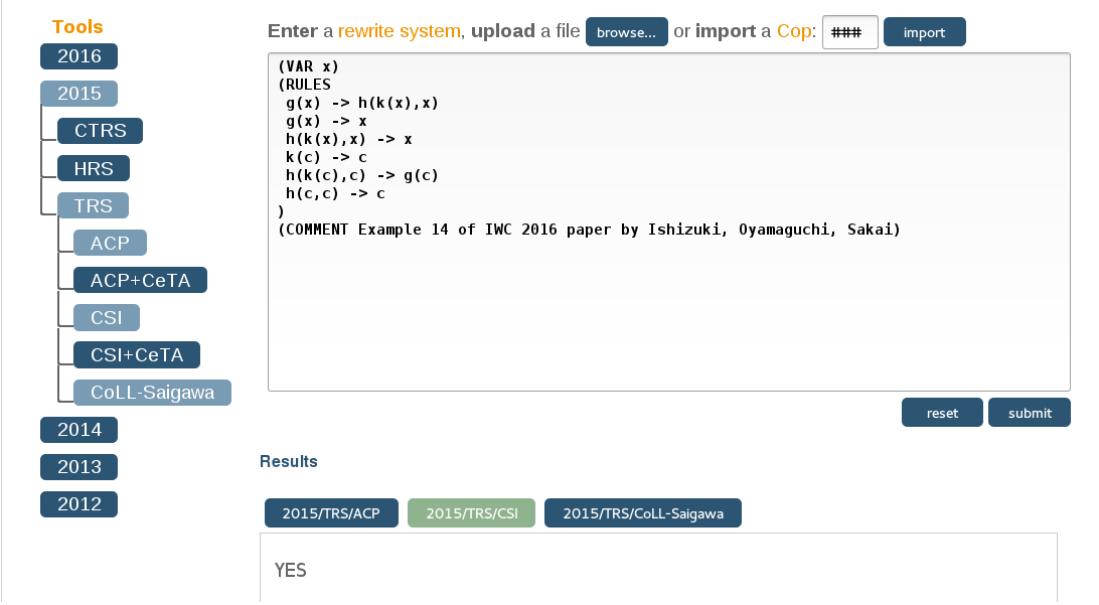

Figure 4: Testing Example 14 from [2] in CoCoWeb.

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

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. There small configuration files reside 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 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 and second argument are the configuration of the tool and input rewrite system respectively:

```
DIR=$(pwd -P)
FILE=$(readlink -f $2)
TO=59
TOT=61
TOK=63
source $1
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 in case it did

not terminate on its own volition. When multiple tools are selected, CoCoWeb runs them sequentially, in order to avoid interference.

4 Possible Extensions

We conclude this note with some ideas for future extensions of the functionality of CoCoWeb.

- By analyzing the format of the input problem, it is often possible to determine the category to which the problem belongs. This information can then be used to restrict tool selection accordingly.
- Adding support for other competition categories like ground confluence and unique normal forms is an obvious extension. This, however, limits the possibilities to restrict tool selection mentioned in the previous item.
- Selected tools are executed sequentially and so if many tools are selected, the 60 seconds time limit per tool may result in an unacceptably slow response. In that case a timer option is a welcome feature.

References

[1] T. Aoto, N. Hirokawa, J. Nagele, N. Nishida, and H. Zankl. Confluence Competition 2015. In *Proc. 25th International Conference on Automated Deduction*, volume 9195 of *Lecture Notes in Artificial Intelligence*, pages 101–104, 2015. doi: 10.1007/978-3-319-21401-6_5.

[2] S. Ishizuki, M. Oyamaguchi, and M. Sakai. Conditions for confluence of innermost terminating term rewriting systems. In *Proc. 5th International Workshop on Confluence*, pages 70–74, 2016. Available from http://cl-informatik.uibk.ac.at/iwc/iwc2016.pdf.

[3] J. Nagele, B. Felgenhauer, and A. Middeldorp. CSI: New evidence – a progress report. In *Proc. 26th International Conference on Automated Deduction*, volume 10395 of *Lecture Notes in Artificial Intelligence*, pages 385–397, 2017. doi: 10.1007/978-3-319-63046-5_24.

[4] N. Nishida, T. Kuroda, M. Yanagisawa, and K. Gmeiner. CO3: A COnverter for proving COnfluence of COnditional TRSs (version 1.2). In *Proc. 4th International Workshop on Confluence*, page 42, 2015. Available from http://cl-informatik.uibk.ac.at/iwc/iwc2015.pdf.

[5] T. Sternagel and A. Middeldorp. Conditional confluence (system description). In *Proc. Joint 25th International Conference on Rewriting Techniques and Applications and 12th International Conference on Typed Lambda Calculi and Applications*, volume 8560 of *Lecture Notes in Computer Science (Advanced Research in Computing and Software Science)*, pages 456–465, 2014. doi: 10.1007/978-3-319-08918-8_31.

[6] A. Stump, G. Sutcliffe, and C. Tinelli. StarExec: A cross-community infrastructure for logic solving. In *Proc. 7th International Joint Conference on Automated Reasoning*, volume 8562 of *Lecture Notes in Artificial Intelligence*, pages 367–373, 2014. doi: 10.1007/978-3-319-08587-6_28.

[7] H. Zankl, B. Felgenhauer, and A. Middeldorp. CSI – A confluence tool. In *Proc. 23rd International Conference on Automated Deduction*, volume 6803 of *Lecture Notes in Artificial Intelligence*, pages 499–505, 2011. doi: 10.1007/978-3-642-22438-6_38.