Competition 2007 of termination tools

C. Marché J. Waldmann H. Zantema

April 6, 2007

1 Overview of the event

In June 2007, the fourth termination competition will be run. Novelties with respect to previous edition are:

- A new category FP for functional programs
- A new "option" for tool which are able to generate automatically checkable proof traces.
- The “second round” with 5 minutes, from last year edition, will not happen anymore.

The competition will be a completely automatic process that will be run without user interaction. During this competition, a large set of termination problems: the termination problem database (TPDB), will be considered, and each of these problems will be submitted to each tool willing to participate. An extra set of problems, called the *secret* problems, submitted by participants together with their tools, will also be considered.

Given a termination problem, each tool is supposed to answer whether the given system/program is terminating. Each tool must run without any user interaction.

The precise rules of the competition are detailed in the next section. Those who are willing to participate should register to the mailing list dedicated to the competition, following instructions of the web page

`http://lists.lri.fr/mailman/listinfo/termtools`

or alternatively by sending a mail to

`termtools-request@lists.lri.fr`

with

`subscribe termtools`

in the body of the message.

2 Rules of the competition

A tool participating to the competition will be evaluated by a completely automatic process. There are four categories of input problems: term rewrite systems, string rewrite systems, logic programs, and functional programs. A tool may participate to any number of categories.

The tool must be available as an executable that takes as argument the name of a file describing a termination problem, and an integer giving the maximal CPU time in seconds allowed to give an answer. The tool is of course free to ignore this second information. The tool must run without any user interaction, and the answer must finally be printed on standard output. The tool may print anything on standard error, such as information on the differents attempts made.

For the term or string rewrite system category, the input file will be in the common format of the TPDB, available as a separate document. The answer must start by either YES or NO, meaning that the given rewrite system is terminating (respectively, not terminating) under the STRATEGY given in the input file. If there is no strategy annotation, strong termination is taken as the default. This text YES or NO should be followed by a *proof trace* of the claimed result : some text providing enough information for the termination being checked by a third party. Termination criteria involved must be clearly stated, with reference to relevant published paper. This proof trace must be in English, but alternatively can be stated in a formal language of some proof assistants like ACL2, Coq, Isabelle/HOL, PVS. In this second case, the tool is eligible to participate to the *certifying tool option*: see related section below.

Any output not starting by YES or NO is considered as DON'T KNOW. If a tool is still running when this maximal CPU time is reached, its process will be killed.

For the logic program category, the input file will be a Prolog program in a standard syntax as in the TPDB. Additionally, in the command line will be given an *input query*, of the form $id(x_1,\ldots,x_n)$ where each x_i is either *i* or *o*. The tool should answer whether the given program terminates on every query with that *moding*.

For the new functional program category, the input file is a Haskell module with one pragma. The details are given below in Section 5.

If it appears that for some termination problem a wrong answer is given, then the tool will be "disqualified".

In the competition every tool is applied on every system in TPDB in the corresponding categories, each with a given time limit. Like previous editions of the competition, the time limit is likely to be between 1 and 5 minutes. The tool receives the given time limit as a command-line argument (see below)

On the web page a score table will be presented for each category, by giving one point for each (correct) answer. Classifications will be computed for each category, several classifications will be made in the rewriting categories, with respect to strategies. The CPU time used will also be listed in the score table; the generated proofs or proof sketches will be accessible from the web page. There is nothing

to win, except of being declared as "the best automatic termination prover in the world in 2006" in the given category. The main results of the competition will be reported at the Ninth International Workshop on Termination (WST 2007), June 29, 2007, Paris, France.

3 The set of termination problems

The set of termination problems will be the termination problem database (TPDB). Any one may submit new termination problems for this database until May 1, 2007.

In order to make the competition more exciting, the set of termination problems used for the competition is extended by a secret list. This secret list is composed from systems that are submitted by the participants just before the competition. More precisely,

- the deadline for submitting new publicly available termination problems for the data base is May 1,
- the deadline for registration and submitting running versions of the tools is May 21,
- the deadline for submitting the final versions of the tools is June 1st
- the deadline for submitting termination problems for the secret list is June 1st, and
- the competition will start on Monday, June 4.

Every participant may submit at most 10 termination problems for the secret list for every category he participates. Typically, he may try to choose problems that can be solved by his tool while they are expected not to be solvable by the other tools.

4 Technical details for participants

If you are willing to participate to the competition, here is what you have to do in details:

- One the authors of the tool (aka the corresponding author) must send a mail to Claude Marché (`Claude.Marche@lri.fr`) giving the following informations :
  1. name of the tool
  2. URL of the web page of the tool
  3. author(s)
  4. short description of the tool features (20 lines max)

- One or more of the authors must register to the `termtools` mailing list as described above
- As soon as possible, provide the program itself, via a web page. The preferred form is directly a stand-alone executable for i386/Linux, since the evaluation will be performed on this architecture. If it is not possible for you to provide this, discuss with `Claude.Marche@lri.fr` to decide what to do.

The program must follow this specification:

- The program which has to be run has to be called `runme`. It could be a shell script which runs the actual tool.
- It runs without user interaction, only in "batch" mode writing on its standard output and standard error.
- It takes as command line arguments the file name containing the termination problem to solve, a maximum allowed CPU time (an integer, in seconds), and for logic programs, an input query.
- The answer must be given on the first line of the standard output, which must be either "YES" if termination has been established, or "NO" if non-termination has been established. Any other first line of answer is considered as "DON'T KNOW". The remaining lines of answer are required to be a proof trace as presented above. The full answer will be accessible from the result table. The standard error may be used to provide information on the attempts made by the tool, it will be also available from the results table.

  Making the first of answer be "YES" or "NO" can of course be done by the `runme` script.
- The program may create temporary files, but only in its current directory, and they must be erased after execution.

5 New category: Haskell programs

The input for the FP category is a Haskell module (as specified in `http://haskell.org/onlinereport/modules.html`) containing exactly one pragma (cf. `http://haskell.org/onlinereport/pragmas.html#pragmas`) that contains the start term for the termination problem, of the form

```
{-# h-terminating x #-}
```

where `x` is any closed Haskell expression (closed = containing no free variables).

Here is an example input:

```
data Nat = S Nat | Z
data List a = Cons a (List a) | Nil

take Z xs = Nil
take n Nil = Nil
take (S n) (Cons x xs) = Cons x (take (p(S n)) xs)

p (S Z) = Z
p (S x) = S (p x)

from x = Cons x (from (S x))

{-# h-terminating  \ u m -> take u (from m) #-}
```

The semantics of "h-terminating" are as given in Definition 3 on page 4 of:

> J. Giesl, S. Swiderski, P. Schneider-Kamp, and R. Thiemann. *Automated Termination Analysis for Haskell: From Term Rewriting to Programming Languages*, In Proceedings of the 17th International Conference on Rewriting Techniques and Applications (RTA-06), Seattle, USA, Lecture Notes in Computer Science 4098, pages 297-312, 2006. Springer-Verlag `http://www-i2.informatik.rwth-aachen.de/giesl/papers/RTA06-distribute.ps`

There will be two sub-categories:

- basic
- Haskell98

The basic sub-category contains the following construct:

- To be completed ASAP

The Haskell98 sub-category is exactly as defined in the Standard (In particular, the input module may reference the Haskell98 Prelude `http://haskell.org/onlinereport/standard-prelude.html` and standard libraries `http://haskell.org/onlinereport/`).

6 New: The certifying option

A tool may register for participation to the *certifying option*. In that case, the proof trace given by the program is supposed to be expressed in a formal language that is able to be automatically checked. This formal proof trace must be exactly the data sent to standard output by the "runme" command, except for the first line which contains the YES/NO answer.

Participation to this certifying option additionally requires to submit a command "checkme", which takes one argument: a file name containing the generated proof trace. This command must return with 0 as exit status if the check went OK, and a non-null number otherwise.

With registration of the tool to this option, participants must provide a short text to explain their checking procedure. The organizers will decide whether the checking procedure is eligible for the certifying option.

Remarks:

- It is allowed for a tool to register 2 versions, one for the certifying option and another one.
- The execution of the "checkme" command will also be limited in time.