



institut für informatik



Project outline

Termination Competition

Simon Bailey
simon.bailey@uibk.ac.at

15th May 2008

Version:

Id: content.tex 33 2008-04-22 10:06:55Z binabik

Abstract

This document outlines the basic structure of the software which will be implemented to manage and run the termination competition.

Contents

1	Initial Component Structure	1
1.1	Software environment	1
2	Execution Environment	2
2.1	Directory Structure	2
2.2	Capturing STDOUT/STDERR	2
2.3	Storing Results	3
2.4	Running The Competition	3
3	Termination Problem Importer	3
4	Problem Submission	4
4.1	Submission Workflow	4
4.1.1	Secret Submissions	5
4.2	Web Interface	5
4.3	Multiple Submissions	6
4.4	Remote access to the database	6
5	Tool Submission	6
5.1	Submission Workflow	6
5.2	Web Interface	6
5.2.1	Deleting Certain Versions of Tools	7
5.3	Version History	7
5.4	Test Run	8
5.4.1	Malicious Programs	8
6	Batch Processor	8
6.1	Job Submission	8
7	Result Tracking	8
8	Administrative Interface	9

1 Initial Component Structure

The software for running the competition will initially consist of the following components:

- Termination problem importer
- Problem submission interface
- Tool submission interface
- Batch processor
- Result tracker
- Administrative interface

The relationship between these components can be seen in Figure 1.

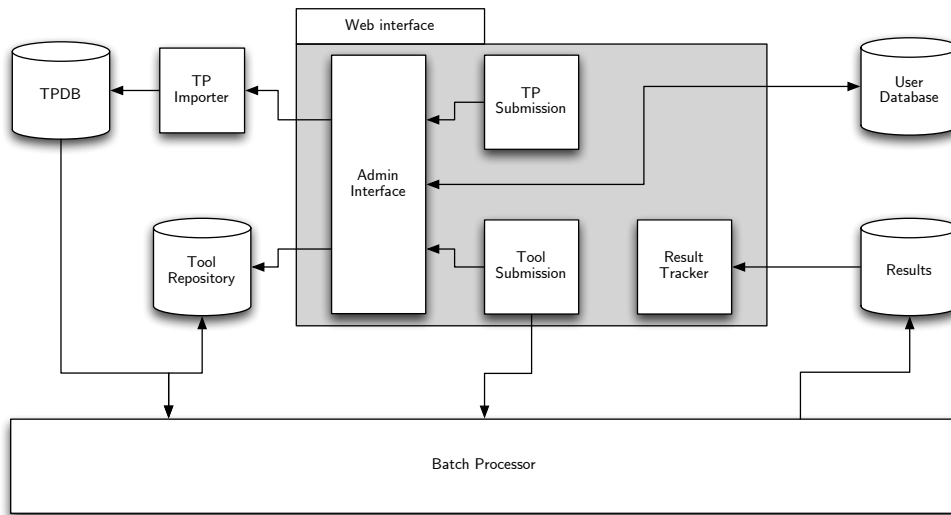


Figure 1: Proposed application architecture.

This paper will outline the functionality of each of these components and offer a guideline for the development of the competition software.

1.1 Software environment

It is our intention to use JavaEE (Enterprise Edition) to create the competition management software. There are multiple reasons for this decision, including unit testing, strong architectural separation (MVC), simple database access, etc. The application server we intend to use is JBoss Community Edition in the latest stable version (at time of writing this is JBoss AS 4.2.2GA). The web view will use the JBoss Seam application framework, the scheduling system will be implemented with Quartz.

2 Execution Environment

This section should answer the following questions:

- How are tools run?
- How will we capture `STDOUT/STDERR`?
- How will we store results?
- How does the competition work?

2.1 Directory Structure

The tools will be run from subdirectories of `/opt/competition/${teamDir}/tools/`. In order to facilitate a batch processing environment, the tools should be able to be run from their own directory—i.e. the directory of any given tool should contain all non-standard libraries and utilities the tool needs. If a tool creates temporary files, it is a requirement that these files are created either in `/opt/competition/tmp/${toolname}` or in a `./tmp/` directory in the current working directory. The reason for this requirement is so that if a tool crashes during execution, any temporary files leftover can be cleaned out by a garbage collecting mechanism.

The `tools` subdirectory will have a further directory for each tool; when a tool is submitted, the archive will be unpacked in the tool’s directory. This means that tool maintainers are responsible for the directory structure of their tool’s directory. In case of naming conflicts, however, the versioning system will overrule the maintainer’s choice of directory names. The web-interface for tool management will allow the tool maintainers to modify existing directories. The tool maintainer will be able to inform the batch processing system which version of a tool it should run (i.e. the directory and name of a binary or script in that directory¹). For more information on the versioning of tools, see Section 5.3.

2.2 Capturing `STDOUT/STDERR`

In previous iterations of the competition, the first line of `STDOUT` was evaluated as the definitive answer, following that it was recommended a *proof trace* should be appended which was then stored as a text file somewhere on the server running the competition. To the extent of our knowledge, all output on `STDERR` was ignored.

We intend to store the output from any run of a program into the main competition database. Depending on the size of the data accumulated after an initial run of the competition, we will evaluate whether the `STDOUT/STDERR` data should be stored for eternity or deleted after a set period of time.

¹This differs slightly from previous iterations where the script was required to be called `runme`.

2.3 Storing Results

Following the convention of the previous competitions, the following results will be stored:

- Termination for given strategy (YES/NO/UNSURE/TIMEOUT)
- CPU time required
- Wall-clock time required
- Correctness of answer (TRUE/FALSE)
- STDERR/STDOUT

These results will be stored in the database, associated with the tool, its version and the date and time of the experiment. If the data was determined during the course of a competition, this will also be reflected in the stored data.

2.4 Running The Competition

In this new version of the termination competition, the original intention is to automate the competition sufficiently for it to be able to run more frequently. With a functional batch processor, this is a given and the periodicity of the competition can easily be changed to run more or less often.

The competition steering committee (or a delegate) will be able to select a set of problems in advance for an upcoming competition—depending on how the committee wants this implemented, this information can be either completely secret, partially secret or publicly available. Tool maintainers will be able to submit their tools for participation in the competition up to 24 hours² before the start of the competition. A tool will automatically take part in the selected categories if the maintainer has selected a “current” version and has not excluded the tool from the next competition (optional opt-out).

In order to make the competition more appealing to watch, it has been suggested that the execution environment select a termination problem at random from the available queue and then run all tools in a random order on this termination problem. The time limits for solving problems will currently remain the same as in previous years (between 1 and 5 minutes); the exact limits should be set by the steering committee.

3 Termination Problem Importer

The current Termination Problem Database (TPDB) managed by Claude Marché is a large collection of text files, each representing a single termination problem. In order to make the collection of termination problems more manageable, we have implemented a database structure for storing the termination problems. If this structure is optimal still remains to be seen. This section will describe the structure of a termination problem importer which can be used to convert the TPDB input syntax to our database structure. A well-implemented importer can in future also be used for submission of new

Correctness is supposed to designate whether the tool reported the correct result, Harald has pointed out that this is not necessarily possible. Comments?

²Arbitrary value, basis for discussion.

problems in the TPDB syntax, which seems to be well established in the termination community (see Section 4 for more details). If, in future, a different format is preferred, the implementation of the importer will ensure that it will be easy to support multiple formats.

Importing a termination problem to the database basically involves the following steps:

1. Parsing
2. Converting
3. Storing

Step 1 will probably utilise the relevant portions of AProVE’s parsing library, which René Thiemann and Jürgen Giesl have kindly agreed to supply. As AProVE is written in Java, the use of this library in the management application will be easy. Once the data has been parsed and is available in a suitable object structure, it can then easily be converted and stored to the database using Hibernate entity objects.

4 Problem Submission

Previously, the TPDB was managed by hand by Claude Marché. For each iteration of the competition, competitors could submit up to 10 secret problems which were added to the publicly available termination problems during the competition. In order to make the TPDB more universally accessible as well as making it easier to search for certain problems, we have decided to implement the TPDB in a relational SQL database with suitable searching tools. This section will describe the submission of new problems to the database, how users can find problems in the database and how multiple problems can be submitted at once.

4.1 Submission Workflow

Problem submission will be available for registered users in the competition management application. A registered user will typically be a tool maintainer and a competitor or a member of a tool’s team. When a user submits a new problem into the database, it will be marked for approval by either the steering committee or a delegate in charge of managing the database. This basic process can be seen in the right-hand (not greyed-out) path in the flow-diagram in Figure 2. Once suitable software has been developed, newly submitted problems can be compared to the existing database to verify that they do not yet exist in the database.³ The database manager can then approve or decline the submission, at which point it is stored in the TPDB and becomes publicly available or is deleted from the submission queue. Once the database manager has made his decision, the user is notified by email.

³The AProVE parsing library already has some rudimentary equality testing implemented, this can be extended in future for automated testing of the database.

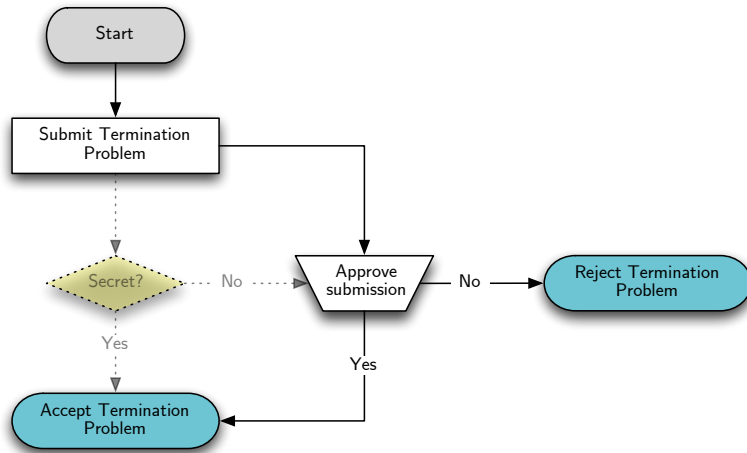


Figure 2: Termination problem submission workflow.

4.1.1 Secret Submissions

At time of writing there is—to the author’s knowledge—no consensus on whether secret problem submissions will be allowed in future. If so, then the method of submission has to be decided upon by the steering committee. An idea would be to allow a set number of submissions per team per competition which are automatically approved (in case the database manager is a member of a competing team) and not publicly available until after the competition has been run. This alternate form of submission is detailed in the left-hand path (greyed-out) in the flow-diagram in Figure 2.

4.2 Web Interface

The proposed web interface will offer users the possibility to submit a termination problem for consideration. This user will be able to submit either a text file with the termination problem or use a suitable web form to create the termination problem directly in the database. The interface will also offer search possibilities so users can find existing problems with certain criteria and download their search results either singly or as an archive. Contrary to the complete archive (see below), these archives will be created on the fly.

The application will also offer the possibility of downloading the complete database in the current TPDB format as an archive. Of course with the termination problems stored in a database, it will also be possible to export the data in a yet to be defined XML format. In order to reduce server load, the complete archive of problems will not be created on the fly, but rather be created on a daily basis by an automated job or by the database manager on approval of new submissions.

If the community or the steering committee would prefer better defined versions of the TPDB, then a periodical generation can be run at longer periods

or on demand by the database maintainer.

4.3 Multiple Submissions

If a user wishes to submit multiple problems, the workflow is more or less the same as outlined above, except that the user will not have the possibility of submitting the problems by entering them into a form. Instead he will be able to upload a set of problems in a ZIP or a gzipped TAR archive which the server extracts, parses and enters into the approval workflow. The importer will determine the nature of the files from the already established file extension (.trs for term-rewriting systems, ...).

4.4 Remote access to the database

If the termination community feels such a tool is required, it may also be possible to create a web-service for searching and retrieving termination problems from the database. This would allow the creation of a command-line client which can be run remotely to retrieve certain termination problems from the database or even to allow tools to integrate such functionality themselves.

5 Tool Submission

In previous iterations of the competition, competitors informed Claude Marché of their intent to participate, and were requested to supply a binary of their tool for download somewhere. This section describes the approach we intend to follow, moving the onus of ensuring the version of the tool is up-to-date from the competition operator to the tool maintainer.

5.1 Submission Workflow

Registered users will be able to create teams; each team will be able to submit one or more tools for participation in the competition. After submission via the web interface, the tool will be scheduled to run on a small test set of problems. If this test run is successful, the tool may be approved for participation, otherwise the user is informed of the problem.⁴ This process is detailed in the flow-diagram in Figure 3.

5.2 Web Interface

The web interface will offer tool maintainers the possibility to upload new versions of their tools, select which version of their tool will be used for the next run of the competition and also request deletion of older versions of their tool, if they so wish. The members of the team will also be able to view the output for test runs of their tools. Tools have to be uploaded in an archive which when

⁴It may be in the tool maintainer's best interest to supply suitable debugging information (stacktraces, etc.) on `STDERR` if the tool detects that errors have occurred, as this information will be stored and made available to the maintainer.

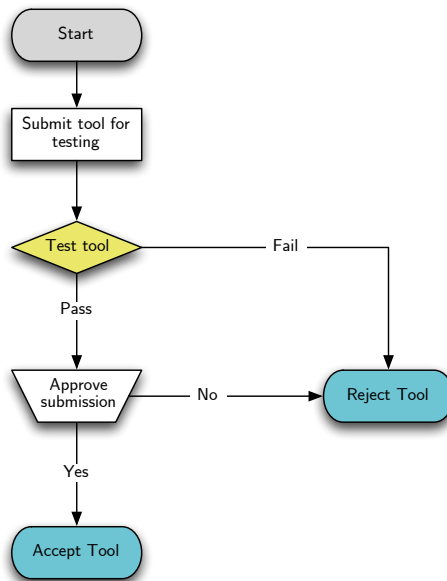


Figure 3: Tool submission workflow.

extracted allows the tool to be run from the extracted directory, which will be created under `/opt/competition/${teamDir}/tools/${toolname}`, as mentioned in Section 2.1. If the directory which would be created already exists, the system will automatically append the version number supplied with the upload to the newly created directory. See Section 5.3 for more information on version numbers.

5.2.1 Deleting Certain Versions of Tools

The general wish for the competition system seems to be that older versions of tools can be compared to newer versions. Thus the steering committee should decide upon criteria for deletion of tools from the competition system. These may include bugs, obsolete versions of libraries, etc. However, deleting a tool from the (file-) system will not delete its results obtained in previous competitions, only remove the tool from the file-system and prevent it from being used in subsequent runs of the competition.

5.3 Version History

Tool maintainers will be able to assign version numbers to their tools when uploading or else let the system assign an automatic version number. This will allow maintainers to select previous versions of their tools for participation in the competition rather than the most current version. A maintainer will also be offered the possibility of opting-out of the next iteration of the competition.

5.4 Test Run

In order to verify that a submitted tool will actually be able to participate in the competition, the competition management will be able to define a small set (c. 10) of termination problems which will be used to conduct a test run with a newly submitted program. This test run should verify that the tool conforms to the competition guidelines (in regard to handling of temporary files, conformity of output on `STDOUT` and `STDERR`, etc.). If a tool successfully completes the test run, it will be allowed participation, otherwise the maintainer will be notified of problems.

5.4.1 Malicious Programs

We trust that there will be no malicious misuse of this system, however there should be the possibility to exclude teams from participation if their programs show malicious misuse of the competition system. This, however, remains for the steering committee to discuss.

6 Batch Processor

The batch processor envisioned for the competition should be flexible enough to be able to conduct the following tasks:

- Accept test runs
- Run the competition for all categories where tools are entered
- Potentially allow exclusive scheduling for tool runs
- Ensure it can utilise multiple servers (clustering)

As these points are fairly self-explanatory, a detailed discussion will be supplied at a later date, once the application design has been finalised.

6.1 Job Submission

If the steering committee and the community feel that the competition system should also be used for on-the-fly comparison of different tools or different versions of a tool on a reference system, the batch processor will also be able to support the submission of such jobs by users, thus scheduling exclusive access to the reference system (i.e. the server or servers which are otherwise used for running the competition).

7 Result Tracking

A competition with no results would be fairly boring. Even though there is currently no prize awarded (except for the monicker “Best Termination Tool in $\$\{\text{year}\}$ ”), the participants are still interested in detailed results. The result tracker component interoperates with the previously mentioned batch processor to execute the following tasks:

- Store in database
- Create different report structures
- Display in a web interface

Once multiple iterations of the competition have been run, the result tracker interface will offer users the possibility to create custom reports, comparing the results of their tool to those achieved in previous competitions. If the competition steering committee deems it important and useful, then the possibility of creating reports comparing different tools and different competition runs will also be possible.

8 Administrative Interface

The competition management will have a number of administrative tasks to perform. This section of the application will mainly be used by the competition management and not be directly accessible for regular users. The administrative tasks are currently as follows:

- Create tests
- Create competitions
- Approve submissions
- User management

This list will probably be extended as the application matures, for this reason the administrative interface will be very modular. It will also be possible to grant users certain roles so that some administrative tasks can be delegated.