An Interface between *Theorema* and External Automated Deduction Systems

Teimuraz Kutsia, Koji Nakagawa

Research Institute for Symbolic Computation

Johannes Kepler University

A−4232 Castle of Hagenberg, Austria

{Teimuraz.Kutsia, Koji.Nakagawa}@risc.uni−linz.ac.at

Abstract

The interface between *Theorema* and external automated deduction systems implements a link providing a user with a tool for using external provers within a *Theorema* session in the same way as "internal" *Theorema* provers. Currently it supports 11 external systems, and support of links to other systems can be easily done. Also, the TPTP (Thousands of Problems of Theorem Provers − problem library) problem converter to *Theorema* format is described.

This research is supported by the Autsrian Science Foundation (FWF) project FO−1302 (SFB).

1. Introduction

The *Theorema* system aims at extending current computer algebra systems by facilities for supporting mathematical proving. It was designed by Bruno Buchberger ([Buchberger, 1996a], [Buchberger, 1996b]) and provides

- a front end for composing formal mathematical text consisting of a hierarchy of axioms, definitions, propositions, algorithms etc. in a common logic frame with user−extensible syntax,
- a library of both well−established and new provers, solvers and simplifiers for proving, solving and simplifying mathematical formulae.

For the resent survey about *Theorema* see [Buchberger et al, 2000]. One of the important design features of *Theorema* is that it is a multi−method system. A large arsenal of general and special provers is available from which the user can choose the appropriate method for the proof problem at hand. Besides its own, "internal" provers *Theorema* can be naturally linked with the existing "external" automated deduction systems, like, e.g.

a theorem prover Otter ([McCune, 1994]). The interface described in this report implements such a link providing a *Theorema* user with a tool for using most of the existing ATP systems within *Theorema* session in the same way as "internal" *Theorema* provers. The interface creates one coherent working environment for different ATP systems and existing *Theorema* provers.

Currently, the following 11 external systems are supported:

- Bliksem –automated theorem prover for first–order logic with equality ([de Nivelle, 1999]) .
- EQP – automated theorem prover for first–order equational logic ([McCune, 1999]).
- E – equational theorem prover ([Schulz, 2000]).
- Gandalf – automated theorem prover for first–order logic with equality ([Tammet, 1997]).
- Ivy – preprocessor and proof checker for first order logic ([McCune and Shumsky, 2000]).
- Mace – finite model and counter–example searcher for first–order statements ([McCune, 2000]).
- Otter – automated theorem prover for first–order logic with equality ([McCune 1994]).
- Scott –automated theorem prover and model searcher for first–order logic with equality – an extension of Otter ([Hodgson and Slaney, 2000]).
- Spass –automated theorem prover for first–order logic with equality ([Weidenbach et al, 1999]).
- Vampire – automated theorem prover for first–order logic with equality ([Riazanov and Voronkov, 1999]).
- Waldmeister – equational theorem prover ([Buch and Hillenbrand, 1996]).

In the current design there are two types of links – direct and indirect – from *Theorema* to external ATP systems. Waldmeister and Scott are linked to *Theorema* indirectly – using first an intermediate translation into TPTP (Thousands of Problems for Theorem Provers – problem library [Suttner and Sutcliffe, 1997]) format and then the TPTP2X converter which converts TPTP format files into a format of the specified ATP system. All the other provers are linked directly to *Theorema*, without any intermediate routine.

Additionally, *Theorema* has one more tool – the TPTP2Theorema converter which converts the TPTP problems to *Theorema* format.

The interface is implemented in Mathematica–4.0/*Theorema* and can be used on all platforms on which Mathematica is available.

The report is organized as follows: Section 1 is the introduction. In Section 2 the direct link is described. Section 3 is devoted to the indirect link. In Section 4 the converter of TPTP problems to *Theorema* format is presented. Section 4 is conclusion. The overall picture of the interface with the basic organization of the report is shown in Figure 1.

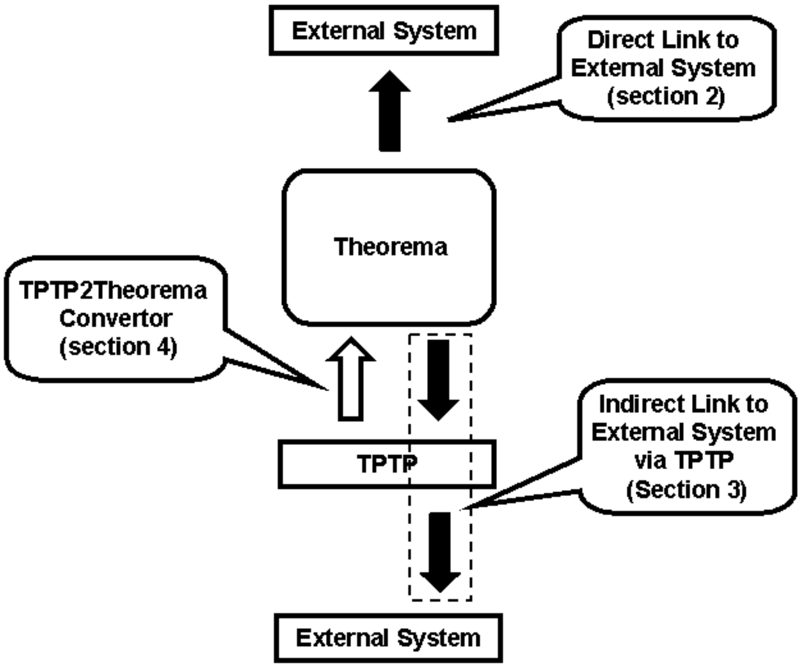

Figure 1. The Interface

2. Direct Link to External ATP Systems

A direct link from *Theorema* to an external system consists of two parts: the translator component and the linking component to the system. The components and the sequence of operations are illustrated in Figure 2.

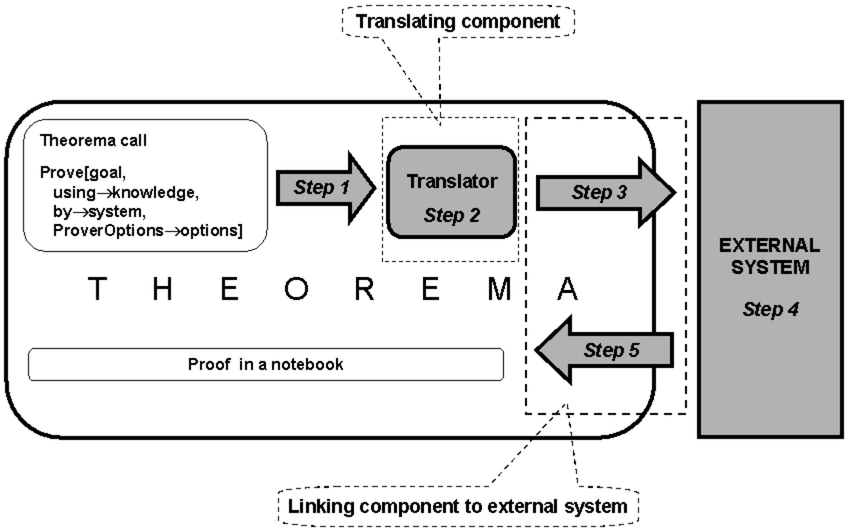

Figure 2. Direct link

The direct link works as follows: first, the translator gets the *Theorema* goal, knowledge base and options and translates them into the prover format thus preparing the input for the prover call. The linking component gets the input string and options, writes the string in a temporary file and calls the prover with the specified options. Finally, the prover output is put into a new notebook and passed back to *Theorema*.

The advantage of the direct link against the indirect one is its flexibility, independence from the intermediate converting routine and speed. The user has a full control over the external system using prover options.

Currently *Theorema* has a direct link with the following external systems: Otter, Spass, EQP, Gandalf, Bliksem, Vampire, E (automated theorem provers), Mace (model builder) and Ivy (proof checker). The example below illustrates each step of the interface.

Example 1. Suppose one wants to prove that the union of the powerset of a set is the original set itself, using the definitions of union, powerset, set equality and set inclusion. Then the goal is the following proposition

Proposition["Union of powerset equals...",
$\quad \underset{A}{\forall} (\bigcup \mathcal{P}[A] = A)]$

and the knowledge base consists of the following definitions

Definition["Set equality",
$\quad \underset{A,B}{\forall} ((A = B) \Longleftrightarrow ((A \subseteq B) \wedge (B \subseteq A)))]$

Definition["Inclusion",
$\quad \underset{A,B}{\forall} \left((A \subseteq B) \Longleftrightarrow \underset{x}{\forall} ((x \in A) \Longrightarrow (x \in B))\right)]$

Definition["Union",
$\quad \underset{A,x}{\forall} \left((x \in \bigcup A) \Longleftrightarrow \underset{Y}{\exists} ((x \in Y) \wedge (Y \in A))\right)]$

Definition["Powerset",
$\quad \underset{A,x}{\forall} ((x \in \mathcal{P}[A]) \Longleftrightarrow (x \subseteq A))].$

To prove this by Otter without any options one has to call

```
Prove[Proposition["Union of powerset equals..."],
    using -> {Definition["Set equality"],Definition["Inclusion"],
            Definition["Union"],Definition["Powerset"]},
    by -> Otter]
```

Then the following steps are performed:

Step 1 (Preparing an input for the translating component). The translator gets the following expressions:

$\bullet\mathrm{lf}\Big[\mathrm{Proposition\ (Union\ of\ powerset\ equals...)},$
$\quad \underset{A}{\forall} (\mathrm{Union}[\mathcal{P}[A]] = A), \bullet\mathrm{finfo}[]\Big]$

for the goal,

$\bullet\mathrm{asml}\Big[\bullet\mathrm{lf}\Big[\mathrm{Definition\ (Set\ equality)},$
$\quad \underset{A,B}{\forall} ((A = B) \Leftrightarrow A \subseteq B \wedge B \subseteq A), \bullet\mathrm{finfo}[]\Big],$

$$\bullet\texttt{lf}\Big[\texttt{Definition (Inclusion)},\ \underset{A,B}{\forall}\Big(A \subseteq B \Leftrightarrow \underset{x}{\forall}(x \in A \Rightarrow x \in B)\Big),\ \bullet\texttt{finfo}[\,]\Big],$$

$$\bullet\texttt{lf}\Big[\texttt{Definition (Union)},\ \underset{A,x}{\forall}\Big(x \in \texttt{Union}[A] \Leftrightarrow \underset{Y}{\exists}(x \in Y \wedge Y \in A)\Big),\ \bullet\texttt{finfo}[\,]\Big],$$

$$\bullet\texttt{lf}\Big[\texttt{Definition (Powerset)},\ \underset{A,x}{\forall}(x \in \mathcal{P}[A] \Leftrightarrow x \subseteq A),\ \bullet\texttt{finfo}[\,]\Big]\Big]$$

for the knowledge and

{}

for the options.

Step 2 (Translating). The translator translates the expressions obtained on the previous step into Otter input format:

```
set(auto).
set(prolog_style_variables).

formula_list(usable).
- (( all A  (union(powerset(A)) = A) )).
( all A B  ((A = B) <-> (subsetequal(A, B) & subsetequal(B, A))) ).
( all A B  (subsetequal(A, B) <->
    ( all X  (element(X, A) -> element(X, B)) )) ).
( all A X  (element(X, union(A)) <->
    ( exists Y  (element(X, Y) & element(Y, A)) )) ).
( all A X  (element(X, powerset(A)) <-> subsetequal(X, A)) ).
end_of_list.
```

Here, `auto` and `prolog_style_variables` flags and `formula_list(usable)` are set by default. Default options can be overridden by explicitly stated prover options.

Step 3 (Preparing an input for the external system). The linking component of the interface creates a temporary input file for Otter and writes the string obtained at the previous step in the file.

Step 4 (Calling the external system). The linking component calls the Otter executable on the input file created on the previous step:

```
% otter < inputfile > outputfile
```

Step 5. (G*etting output from the external system*). Otter output is taken from the `outputfile` and is put into a new notebook as it is shown on Figure 3.

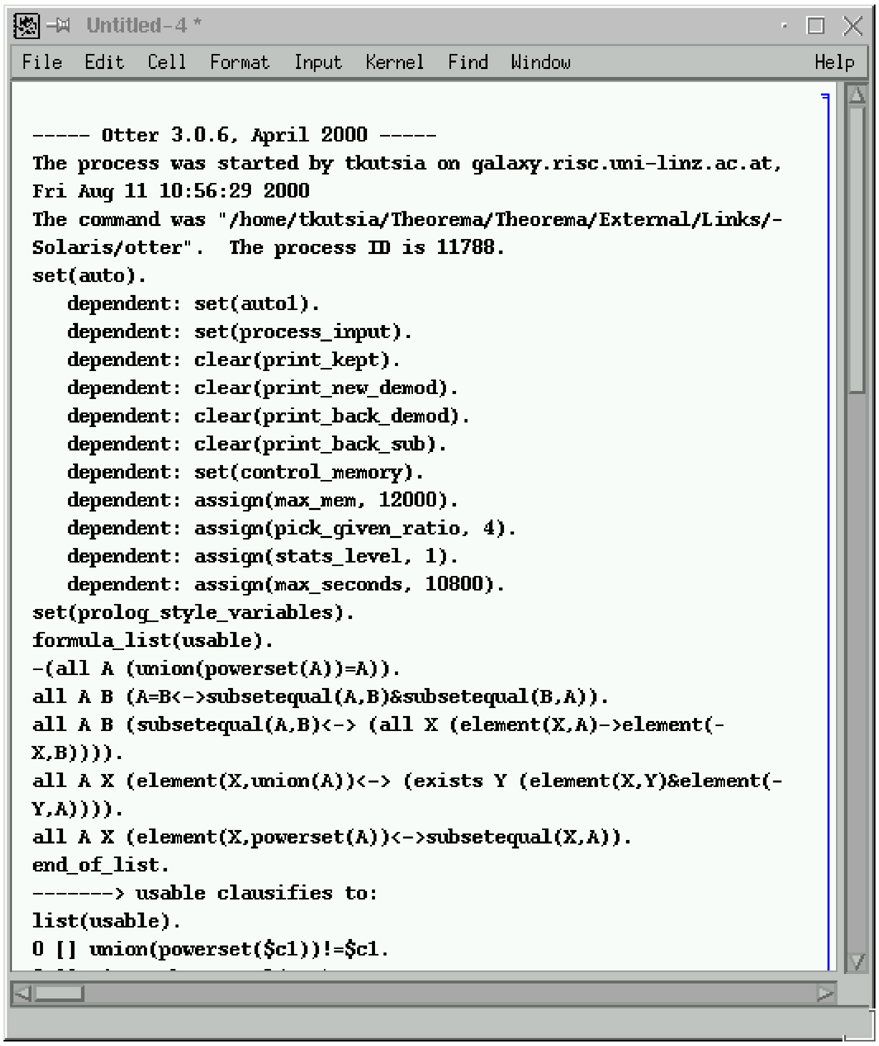

Figure 3. Notebook with Otter output

Among the other cells the notebook contains the proof section:

```
----------------PROOF----------------
1 [] union(powerset($c1))!=$c1.
2 [] A!=B|subsetequal(A,B).
3 [] A!=B|subsetequal(B,A).
4 [] A=B| -subsetequal(A,B)| -subsetequal(B,A).
5 [] -subsetequal(A,B)| -element(C,A)|element(C,B).
6 [] subsetequal(A,B)| -element($f1(A,B),B).
7 [] -element(A,union(B))|element(A,$f2(B,A)).
8 [] -element(A,union(B))|element($f2(B,A),B).
9 [] element(A,union(B))| -element(A,C)| -element(C,B).
10 [] -element(A,powerset(B))|subsetequal(A,B).
11 [] element(A,powerset(B))| -subsetequal(A,B).
```

```
12 [factor,4,2,3] A=A| -subsetequal(A,A).
14 [] subsetequal(A,B)|element($f1(A,B),A).
15 [hyper,14,12] element($f1(A,A),A)|A=A.
16 [hyper,14,11] element($f1(A,B),A)|element(A,powerset(B)).
23 [hyper,15,6] A=A|subsetequal(A,A).
26 [hyper,23,12,factor_simp] A=A.
27 [hyper,26,3] subsetequal(A,A).
28 [hyper,27,11] element(A,powerset(A)).
33 [hyper,16,7] element(union(A),powerset(B))|element($f1(union(A),B),
$f2(A,$f1(union(A),B))).
45 [hyper,28,9,16] element($f1(A,B),union(powerset(A)))|element(A,
powerset(B)).
1190 [hyper,45,6] element(A,powerset(union(powerset(A))))|subsetequal(A,
union(powerset(A))).
3083 [hyper,1190,11,factor_simp] element(A,powerset(union(powerset(A)))).
3403 [hyper,3083,10] subsetequal(A,union(powerset(A))).
3500 [hyper,3403,4,14] union(powerset(A))=A|element($f1(union(powerset(A)),
A),union(powerset(A))).
5742 [para_from,3500.1.1,1.1.1,unit_del,26] element($f1(union(powerset($c1)),
$c1),union(powerset($c1))).
5749 [hyper,5742,8] element($f2(powerset($c1),$f1(union(powerset($c1)),$c1)),
powerset($c1)).
5756 [hyper,5749,10] subsetequal($f2(powerset($c1),$f1(union(powerset($c1)),
$c1)),$c1).
5761 [hyper,5756,5,33] element($f1(union(powerset($c1)),$c1),$c1)|
element(union(powerset($c1)),powerset($c1)).
5763 [hyper,5761,6] element(union(powerset($c1)),powerset($c1))|
subsetequal(union(powerset($c1)),$c1).
5766 [hyper,5763,11,factor_simp] element(union(powerset($c1)),
powerset($c1)).
5767 [hyper,5766,10] subsetequal(union(powerset($c1)),$c1).
5815 [hyper,5767,4,3403,flip.1] union(powerset($c1))=$c1.
5817 [binary,5815.1,1.1] $F.
```

------------ end of proof -------------

```
Search stopped by max_proofs option.
```

3. Indirect Link to External ATP Systems

An indirect link from *Theorema* to an external system consists of three parts: the translator into TPTP format, the linking component to TPTP2X and the linking component to the external prover. The components and the sequence of operations are illustrated in Figure 4.

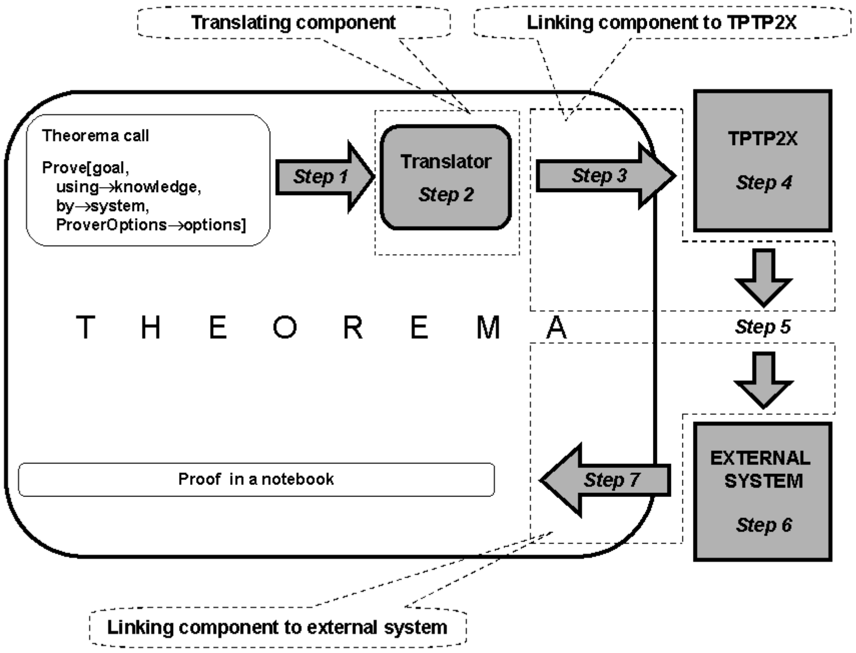

Figure 4. Indirect link

We have chosen the TPTP format for the intermediate translation because of the tptp2X converter, which can convert TPTP format files basically in all existing ATP system format. We maintain the indirect link because it is easy to program and to extend for a new system. The indirect link works as follows: the *Theorema* call `Prove[goal, using->knowledge, by->prover, ProverOptions->Options]` invokes the translator into TPTP format which gets the goal and knowledge base, translates them into TPTP format and puts the result into a temporary file. Next, the linking component to TPTP2X calls the script tptp2X provided with the TPTP library on the file with the command line options relevant to the prover. The obtained result is a text file with the goal and knowledge specification in the prover format which is passed to the linking component to the prover together with the prover options. The linking component calls the prover, gets the result back and puts it into a new notebook.

The main advantage of the indirect link is its easy extendability – it is almost straightforward to add and maintain an indirect link to a new prover, but is has a disadvantage as well – the user has less control over the system then in the case of the direct link.

Currently, *Theorema* has an indirect link with the equational prover Waldmeister and the general purpose prover Scott.The example below illustrates each step of the interface.

Example 2. Suppose one wants to prove that every Robbins algebra satisfying idempotence of addition is a Boolean algebra ([Winker 1990]). Then the knowledge base consists of the Robbins algebra axioms

```
Axiom["Commutativity-of-add",any[X,Y],add[X,Y]=add[Y,X]]

Axiom["Associativity-of-add",any[X,Y,Z],add[add[X,Y],Z]=add[X,add[Y,Z]]]

Axiom["Robbins-axiom",any[X,Y],negate[add[negate[add[X,Y]],
        negate[add[X,negate[Y]]]]]=X]
```

and the assumption about idempotence of addition

```
Assumption["Idempotence",add[c,c]=c].
```

To show that under these assumptions the Robbins algebra is a Boolean algebra it is enough to prove Huntington axiom, i.e. to prove the goal

```
Proposition["Prove-huntingtons-axiom",add[negate[add[a,negate[b]]],
        negate[add[negate[a],negate[b]]]]=b].
```

To prove it by Waldmeister one has to call

```
Prove[Proposition["Prove-huntingtons-axiom"],
    using ->{Axiom["Commutativity-of-add"],Axiom["Associativity-of-add"],
            Axiom["Robbins-axiom"],Assumption["Idempotence"]},
    by -> Waldmeister]
```

This call invokes the indirect link to Waldmeister which preforms the following steps:

Step 1 (Preparing an input for the translating component). The translator gets the following expressions:

```
•lf["Proposition (Prove-huntingtons-axiom)",add[negate[add[a,negate[b]]],
     negate[add[negate[a],negate[b]]]]=b,•finfo[]]
```

for the goal,

```
•asml[•lf[Axiom (Commutativity-of-add),
                ∀_{X,Y}(add[X,Y]=add[Y,X]), •finfo[]],
    •lf[Axiom (Associativity-of-add),
                ∀_{X,Y,Z}(add[add[X,Y],Z]=add[X,add[Y,Z]]), •finfo[]],
    •lf[Axiom (Robbins-axiom),
          ∀_{X,Y}(negate[add[negate[add[X,Y]]],negate[add[X,negate[Y]]]]=X)],
    •lf[Assumption (Idempotence), add[c,c]=c, •finfo[]]]
```

for the knowledge and

{}

for the options.

Step 2 (Translating). The translator translates them into TPTP CNF format:

```
input_clause(provehuntingtonsaxiomclause41,conjecture,
    [--equal(add(negate(add(a, negate(b))),
            negate(add(negate(a), negate(b)))), b)]).

input_clause(commutativityofaddclause42,axiom,
    [++equal(add(X, Y), add(Y, X))]).

input_clause(associativityofaddclause43,axiom,
    [++equal(add(add(X, Y), Z), add(X, add(Y, Z)))]).

input_clause(robbinsaxiomclause44,axiom,
    [++equal(negate(add(negate(add(X, Y)),
            negate(add(X, negate(Y))))), X)]).

input_clause(idempotenceclause45,axiom, [++equal(add(c, c), c)]).
```

Step 3 (Preparing input for tptp2X script). The linking component to TPTP2X creates a temporary file and puts in it the string obtained in the previous step.

Step 4 (Calling tptp2X). The linking component to TPTP2X calls the script tptp2X with the options

```
-f waldmeister -t rm_equality:rstfp
```

on the file got in the previous step. The result is another temporary file with the Waldmeister input:

```
%--------------------------------------------------------------------------
% File     : m00006688821 : TPTP v0.0.0. Released v0.0.0.
% Domain   :
% Problem  :
% Version  :
% English  :
% Refs     :
% Source   :
% Names    :
% Status   : unknown
% Rating   : ?
% Syntax   : Number of clauses    :    5 (   0 non-Horn;   5 unit;   2 RR)
%            Number of literals   :    5 (   5 equality)
%            Maximal clause size  :    1 (   1 average)
%            Number of predicates :    1 (   0 propositional; 2-2 arity)
%            Number of functors   :    5 (   3 constant; 0-2 arity)
%            Number of variables  :    7 (   0 singleton)
%            Maximal term depth   :    6 (   2 average)
% Comments :
%          : tptp2X -f waldmeister -t rm_equality:rstfp m00006688821.p
%--------------------------------------------------------------------------
```

```
NAME            Unknown
MODE            PROOF
SORTS           ANY
SIGNATURE       a: -> ANY
                add: ANY ANY -> ANY
                b: -> ANY
                c: -> ANY
                negate: ANY -> ANY
ORDERING        KBO
                negate=1, add=1, c=1, b=1, a=1
                negate > add > c > b > a
VARIABLES       A,B,C: ANY
EQUATIONS       add(add(A,B),C) = add(A,add(B,C))
                    % associativityofaddclause43
                add(A,B) = add(B,A)
                    % commutativityofaddclause42
                add(c,c) = c
                    % idempotenceclause45
                negate(add(negate(add(A,B)),
                    negate(add(A,negate(B))))) = A
                    % robbinsaxiomclause44
CONCLUSION      add(negate(add(a,negate(b))),
                    negate(add(negate(a),negate(b)))) = b
                    % provehuntingtonsaxiomclause41
%--------------------------------------------------------------------------
```

Step 5 (Output from tptp2X – input for the external system). The linking component to TPTP2X passes the file got on the previous step to the linking component to external system.

Step 6 (Calling the external system). The linking component to external prover calls Waldmeister executable on the file got from TPTP2X:

```
% waldmeister inputfile > outputfile
```

Step 7 (getting output from the external system). Waldmeister output is taken from the `outputfile` and is put in a new notebook as it is shown on Figure 5.

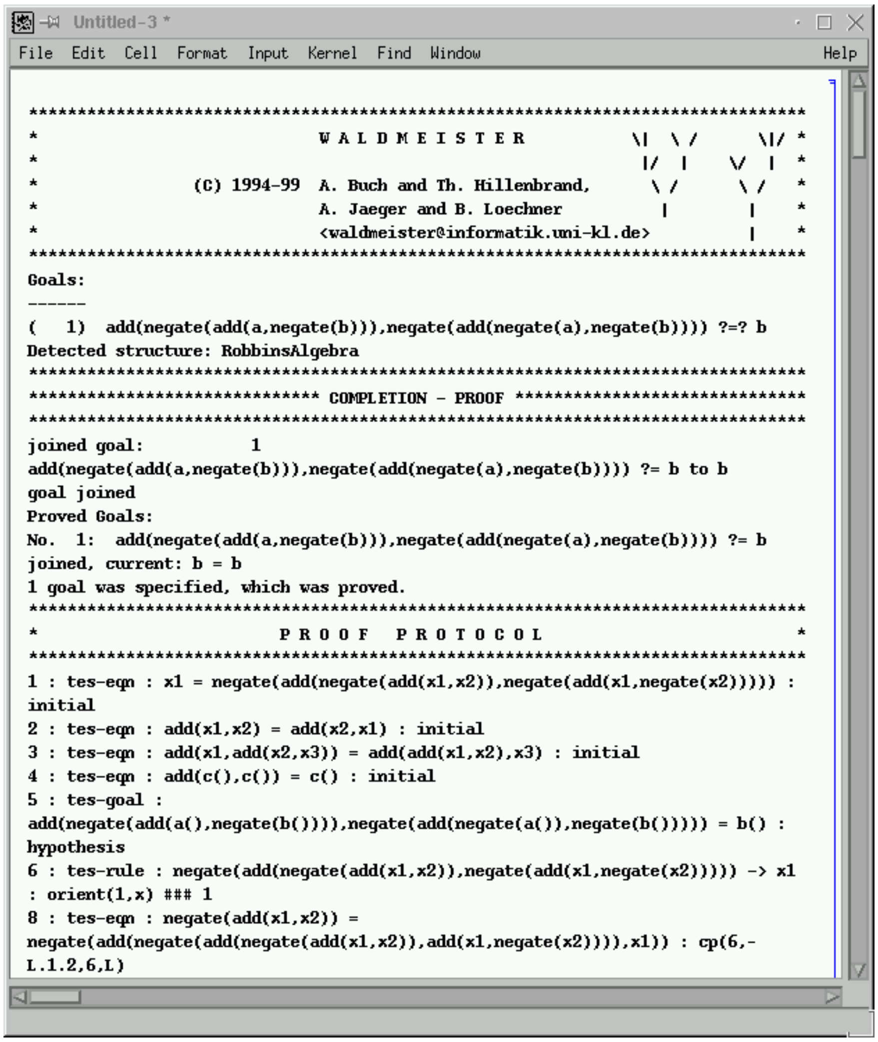

Figure 5. Notebook with Waldmeister output.

It consists of the header part, problem specification part and the proof protocol.

4. TPTP Problem Converter to Theorema Format

The TPTP (Thousands of Problems for Theorem Provers) Problem Library is a comprehensive library of the ATP test problems that are available today. A utility to convert the problems to existing ATP formats, called tptp2X, can convert the problems to almost all ATP formats. The TPTP2Theorema converter, written in Mathematica, makes the library problems available in *Theorema* format. It takes a TPTP file, converts it and puts the result in a notebook.

The converter works in an interactive mode. Calling the command `TPTP2Theorema` opens the notebook with the description of steps necessary to convert TPTP problems into *Theorema* format (Figure 6).

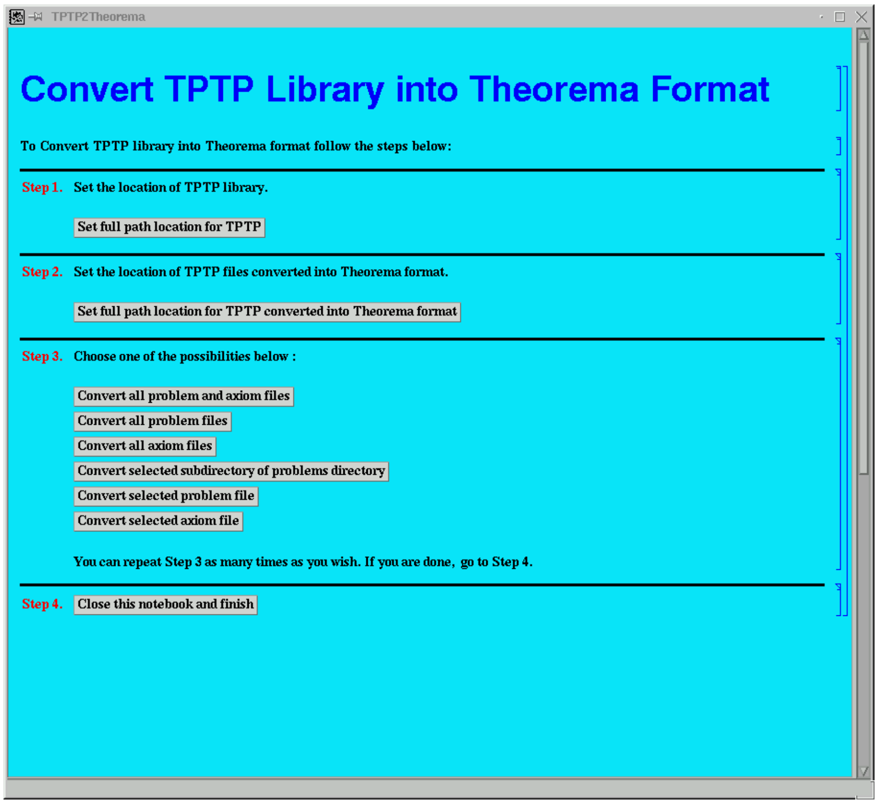

Figure 6. Notebook with TPTP2Theorema menu.

The only thing the user has to do is to follow the corresponding links on each step. They are the following:

Step 1. Setting the TPTP location – choose the directory where TPTP is located.

Step 2. Setting the location of TPTP files converted into *Theorema* format – choose the directory where TPTP files in *Theorema* format should be located.

St ep3. Choose one of the possibilities (this step can be performed several times):

- Convert all problem and axiom files – converts all files from TPTP Problems and Axioms directories and puts them into corresponding subdirectories under the location specified at Step 2.

- Convert all problem files – converts all files from TPTP Problems directory and puts them into corresponding subdirectory under the location specified at Step 2.

- Convert all axiom files – converts all files from TPTP Axioms directory and puts them into corresponding subdirectory under the location specified at Step 2.

- Convert selected subdirectory of problems directory – allows to choose a subdirectory of TPTP Problems directory (e.g. ALG), converts all files from there and puts them into corresponding subdirectory under the location specified at Step 2.

- Convert selected problem file – allows to select a TPTP Problem file (e.g. ALG001–1.p), converts and puts it into corresponding subdirectory under the location specified at Step 2.

- Convert selected axiom file – allows to select a TPTP Axiom file (e.g. ALG001–0.ax), converts and puts it into corresponding subdirectory under the location specified at Step 2.

Step 4. Close the menu notebook and finish.

The converter substitutes include instructions in the TPTP problem files with the actual formulae. The structure of the output notebook follows the original problem file, having sections for the header, theory (the axioms from the original file, included axioms and assumptions) and conjectures. Besides, the notebook contains the title part, hyperlinks to TPTP home page and documentation, subtitle with the problem name, form and domain, separate section with the conjecture formed for *Theorema* (obtained by negating the conjunction of the original conjectures), problem status explanation for *Theorema* and the 'Prove' statement. The only thing the user needs to do is just to specify prover name (s)he wants to use and set corresponding options (if any). A notebook with a TPTP problem converted to *Theorema* format is shown in Figure 7.

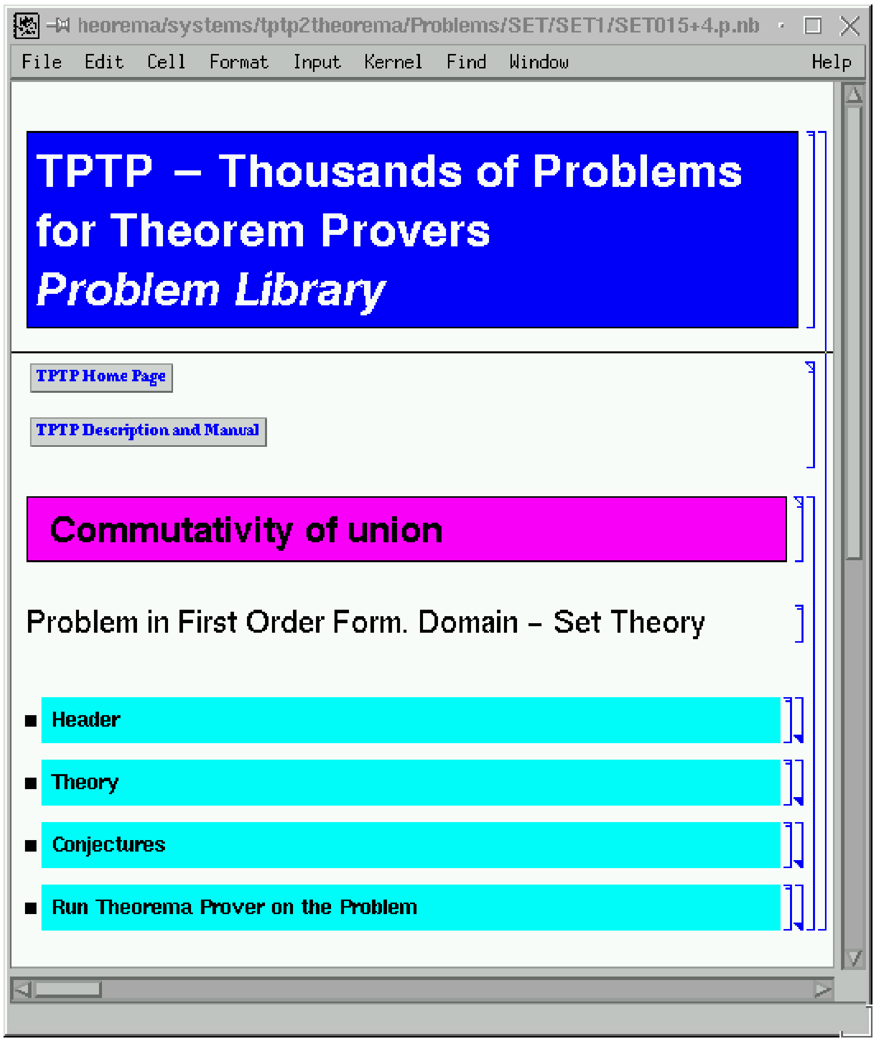

Figure 7. Notebook with a TPTP problem converted to *Theorema* format

5. Conclusion

We described the interface between *Theorema* and external automated deduction tools. The interface between *Theorema* and external provers represents a coherent base for different external theorem proving systems and *Theorema* internal provers thus giving the user a possibility to use different proving systems within a *Theorema* session in the same way as one could use the *Theorema* provers. The converter of TPTP problems to *Theorema* format makes available the TPTP problem library for *Theorema* users.

References

[Buch and Hillenbrand, 1996] A. Buch and Th. Hillenbrand. *Waldmeister: Development of a High Performance Completion−Based Theorem Prover*, SEKI−Report SR−96−01. Universitaet Kaiserslautern, Germany, 1996.

[Buchberger, 1996a] B. Buchberger. *Symbolic Computation: Computer Algebra and Logic.* In: Frontiers of Combining Systems, F. Baader and K.U. Schulz (eds.), Applied Logic Series. Kluwer Academic Publishers, 1996, pp. 193−220.

[Buchberger 1996b] B. Buchberger. *Using Mathematica for Doing Simple Mathematical Proofs.* Invited paper in: Proceedings of the 4th Mathematica Users' Conference, Tokyo, November 2, 1996, Wolfram Media Publishing, pp. 80−96.

[Buchberger et al 2000] B. Buchberger, C. Dupre, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, W.Windsteiger. *Theorema: A Progress Report.* In: M. Kerber and M. Kohlhase (eds.), Proceedings of the 8th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, St. Andrews, Scotland, August 6−7, pp. 100−115, 2000.

[Hodgson and Slaney 2000] K. Hodgson and J. Slaney. *Semantic Guidance for Saturation−based Theorem Proving*, Technical Report TR−ARP−04−2000, Automated Reasoning Project, Australian National University, Canberra, Australia, 2000.

[McCune, 1994] W. McCune. *Otter 3.0 Reference Manual and Guide.* ANL−94/6, Argonne National Laboratory, Argonne, USA, 1994.

[McCune, 1999] W. McCune. *EQP 0.9 User's Guide.* 1999, available at http://www.mcs.anl.gov/AR/eqp/Manual.txt.

[McCune, 2000] W. McCune. *MACE: Models and Counterexamples,* 2000, available at http://www.mcs.anl.gov/AR/mace/.

[McCune and Shumsky, 2000] W. McCune and O. Shumsky. *Ivy: A Preprocessor and Proof Checker for First−Order Logic.* In: Computer−Aided Reasoning: ACL2 Case Studies, M. Kaufmann, P. Manolios, and J.Moore (eds.), Kluwer Academic Publishers, 2000, pp. 265−282.

[de Nivelle, 1999] H. de Nivelle. *Bliksem 1.10 User Manual,* 1999, available at http://www.mpi−sb.mpg.de/~bliksem/manual.ps.

[Riazanov and Voronkov, 1999] A. Riazanov and A. Voronkov. *Vampire*, In: Proceedings of the 16th International Conference on Automated Deduction, H. Ganzinger (ed.), Lecture Notes in Artificial Intelligence, 1632, Springer−Verlag, 1999, pp. 292−296.

[Schulz, 2000] S. Schulz. *E 0.6x User Manual*, 2000, available at http://wwwjessen.informatik.tu−muenchen.de/~schulz/WORK/eprover.ps.

[Suttner and Sutcliffe, 1997] C. Suttner and G. Sutcliffe, *The TPTP Problem Library (TPTP v2.1.0)*, Technical report AR−97−01, Institut fuer Informatik, Technische Universitaet Muenchen, Munich, Germany, Technical Report 97/04, Department of Computer Science, James Cook University, Townsville, Australia.

[Tammet, 1997] T. Tammet. *Gandalf Version c−1.0c Reference Manual*, 1997, available at http://www.cs.chalmers.se/~tammet/gandalf/manual.ps.

[Weidenbach et al, 1999] C. Weidenbach, B. Afshordel, U. Brahm, C. Cohrs, T. Engel, E. Keen, C. Theobalt, D. Tpoic. *System Description: SPASS Version 1.0.0,* In: Proceedings of the 16th International Conference on Automated Deduction, H. Ganzinger (ed.), Lecture Notes in Artificial Intelligence, 1632, Springer−Verlag, 1999, pp. 378−382.

[Winker, 1990] S. Winker. *Robbins Algebra: Conditions That Make a Near−Boolean Algebra Boolean*, Journal of Automated Reasoning 6(4), 1990, pp. 465−489.