# Integrating TPS with ΩMEGA

Christoph Benzmüller and Volker Sorge
{chris|sorge}@cs.uni-sb.de

Fachbereich Informatik, Universität des Saarlandes
D-66141 Saarbrücken, Germany

**Abstract.** We report on the integration of TPS as an external reasoning component into the mathematical assistant system ΩMEGA. Thereby TPS can be used both as an automatic theorem prover for higher order logic as well as interactively employed from within the ΩMEGA environment. TPS proofs can be directly incorporated into ΩMEGA on a tactic level enabling their visualization and verbalization. Using an example we show how TPS proofs can be inserted into ΩMEGA's knowledge base by expanding them to calculus level using both ΩMEGA's tactic mechanism and the first order theorem prover OTTER. Furthermore we demonstrate how the facts from ΩMEGA's knowledge base can be used to build a TPS library.

## 1 Introduction

Current theorem provers, whether automatic or interactive ones, usually have strength in some specific domains while lacking reasoning power in others. Therefore there have been several attempts in recent years to combine two or more provers in order to enhance their power and facilities. On one hand these combinations have been done for the purpose of sharing databases between different systems and thereby avoiding the duplication of work by constructing analogous databases for all systems involved [FH97]. On the other hand there are attempts to integrate several existing provers into a single architecture to make use of various kinds of reasoning strategies and proof procedures in a cooperate system [GPT96]. Furthermore, in some interactive systems, external reasoning components – usually first order automatic theorem provers – are used to support the user when proving a theorem interactively, by automatically justifying simple open subgoals [Mei97].
In this paper we report on an experiment of integrating the higher order theorem proving system TPS [ABI+96] into the mathematical assistant ΩMEGA [BCF+97] for the benefit of both systems. The integration of higher order reasoning components is highly desirable for ΩMEGA as its database of mathematical theories consists mainly of higher order concepts, so that many problems formulated using this database lie naturally beyond the capabilities of the already integrated first order theorem provers SPASS [WGR96], OTTER [McC94] and PROTEIN [BF94]. TPS on the other hand gains a graphical proof display and a component for proof verbalization from its integration with ΩMEGA. Furthermore TPS can in principal be extended in order to integrate proofs passed from ΩMEGA,

such that both systems can be integrated on the same level, perhaps sharing a common knowledge base.
The integration was easified by the fact that TPS and OMEGA implement the same logic, i.e. classical higher order logic based on Church's simply typed $\lambda$-calculus. Both systems use variants of Gentzen's natural deduction calculus (ND) [Gen35] and privide all the information, e.g. full type information of terms, that is necessary for the translation from one system into the other.
We want to emphasize that the work presented in this paper is essentially an extension of the $\Omega$MEGA system in order to integrate TPS as a powerful external reasoning component.

*Remarks on* TPS TPS can be used either as fully automatic or as an interactive proof development environment based on an (extended) variant of Gentzen's natural deduction calculus (ND) [Gen35]. Even in interactive mode the automatic component can be called on subproblems. It uses the mating method (connection method) [And89] as reasoning technique and provides several built-in search strategies as well as many options to adjust these strategies interactively or even automatically. Furthermore the automatic prover has the ability of selectively expanding definitions [BA98] based on a dual instantiation strategy. This strategy provides an effective way to decide which abbreviations to instantiate when searching for a proof. The system also allows the user to interrupt the automatic proof process in order to analyze it and to influence further mating search. A very important feature of TPS for our integration is that each proof found by its automatic component gets automatically transformed into a natural deduction proof based on the work of [Mil84,Pfe87]. Furthermore TPS provides comprehensive library facilities for the maintenance of different kinds of objects, such as problems, (polymorphic) definitions, theorems, rewrite rules or even modes specifying flag-settings connected to previously proven theorems.

*Remarks on* $\Omega$MEGA The $\Omega$MEGA-system [BCF+97] for classical type theory (Church's typed $\lambda$-calculus) is designed as an interactive mathematical assistant system, aimed at supporting proof development in mainstream mathematics. It consists of a variety of tools including a proof planner [HKRS94], a graphical user interface LOUI [SHB+98], the PROVERB system [HF97] for translating proofs into natural language and a variety of external systems, such as computer algebra systems [KKS98], constraint solvers and automated theorem provers [Mei97].
The basic calculus underlying $\Omega$MEGA is similar to TPS', i.e., a variant of ND. However the set of rules in $\Omega$MEGA is smaller then the one in TPS. This stems from the necessity of keeping TPS proofs concise for displaying then in a user-friendly fashion. Therefore certain rules abstract over small subproofs (such as *RuleP* over proofs in propositional logic, cf. section 3.1). In $\Omega$MEGA however the set of basic ND-rules is just large enough to ensure completeness and all extensions to the basic ND-calculus (e.g. equality substitution) are defined as tactics. Nevertheless, proofs can be both constructed and displayed on several abstract levels by using a

three-dimensional data structure[1] for representing (partial) proofs. The structure on the one hand enables the user to freely switch back and forth between different abstract levels and on the other hand provides a means for directly integrating results of external reasoners while leaving the expansion to the calculus level to $\Omega$MEGA's tactic mechanism.
We demonstrate the integration of TPS and $\Omega$MEGA with a simple example from set theory, which we will solve for demonstration purposes partly interactive within $\Omega$MEGA while passing two subproblems to TPS[2]. In order to gain a checkable $\Omega$MEGA calculus level proof, the original TPS proof is inserted on a tactic level and expanded via several levels of abstraction and with the help of the first order theorem prover OTTER. While discussing the example we exhibit the advantageous side effects for TPS, (1) that the proof can be directly visualized and (2) its structure displayed graphically in $\Omega$MEGA's user interface LOUI, and (3) with the available different abstraction levels it can be verbalized by the PROVERB system. We show the first two effects with the help of several screen-shots and exemplify the latter by giving a short verbalization of a subproof of our theorem. In order to efficiently make use of TPS' ability to selectively expand definitions we describe in section 4 an algorithm that automatically imports definitions and recursively also imports all necessary subconcepts from the $\Omega$MEGA knowledge base to TPS. To explain the working scheme of this algorithm we use another example from set theory.

## 2 Integrating TPS with $\Omega$MEGA

The integration of TPS into $\Omega$MEGA provides two different modes for its use. One mode is to call TPS as black box system for proving a given subproblem, similar to the use of the first order theorem provers already integrated in $\Omega$MEGA. A second mode offers the possibility of employing TPS as an interactive theorem proving environment itself. While in the first mode the proof search of TPS as a black box can only be influenced by elementary flag settings adjustable as $\Omega$MEGA command parameters , e.g. flags specifying a concrete mating search procedure instead of the standard uniform search strategy, the user may take advantage of all interactive features of TPS when calling it in interactive mode. For both types of integration we currently use a file based communication between the two systems.

### 2.1 Black Box Integration

We demonstrate the black box integration of TPS into $\Omega$MEGA by using the following example: "If there exists no mapping from a set *set1* into

[1] $\Omega$MEGA's proof datastructure stores the (partial) proof on basic ND-level as well as the more abstract levels containing nodes which are justified with tactics and or methods and which correspond to certain parts of the proof on the underlying level.
[2] This problem can be solved by TPS even without any interaction.

a set *set2* then *set2* is the empty set." Using $\Omega$MEGA's theory *naive-set* this theorem can be formalized as[3]:

**assumption** $\neg\exists f_{\iota\iota}\text{.}\ \forall u_{\iota}\text{.}\ u \in \mathit{set1} \Rightarrow \exists v_{\iota}\text{.}\ v \in \mathit{set2} \wedge (f\ u) = v$

**theorem** $\mathit{set2} = \emptyset$

where the following polymorphic definitions are provided by the *naive-set* theory[4]: $\in\ \dot{=}\ \lambda e_{\beta}\text{.}\ \lambda m_{o\beta}\text{.}\ me$ and $\emptyset\ \dot{=}\ \lambda x_{\beta}\text{.}\ \bot$. Even though TPS is able to solve this problem automatically we prove this theorem for demonstration purposes partially interactive with $\Omega$MEGA and partially automatic with TPS. We begin with introducing the following lemma in $\Omega$MEGA:

**lemma** $(\neg\exists w_{\iota}\text{.}\ w \in \mathit{set2}) \Rightarrow \mathit{set2} = \emptyset$

Next we apply the implication elimination rule (modus ponens) backwards using the theorem as succedent which splits the original proof problem into two subproblems: (a) showing that $\neg\exists w_{\iota}\text{.}\ w \in \mathit{set2}$ follows from the assumption $\neg\exists f_{\iota\iota}\text{.}\ \forall u_{\iota}\text{.}\ u \in \mathit{set1} \Rightarrow \exists v_{\iota}\text{.}\ v \in \mathit{set2} \wedge (f\ u) = v$ and (b) showing that the newly introduced lemma is valid. This proof situation is visualized in figure 1.

Before applying TPS we need to eliminate in a preliminary step the defined construct $\in$ in both the assumption (node 1) and the subgoal (node 5). This is necessary as $\in$ is a definition from $\Omega$MEGA's knowledge base that is unknown to TPS. We can now call TPS thereby closing the left branch in the proof tree.

The original proof generated by TPS (see figure 3; for detailed information about the occurring TPS-justifications we refer to [AIN+97]) can be displayed within $\Omega$MEGA by applying the command *show-tps-proof* on node 7. The idea of the indirect proof is to derive a contradiction from $\exists w_{\iota}\text{.}\ w \in \mathit{set2}$ and the above assumption by showing that there indeed exists an $f_{\iota\iota}$, such that $\forall u_{\iota}\text{.}\ u \in \mathit{set1} \Rightarrow \exists v_{\iota}\text{.}\ v \in \mathit{set2} \wedge (f\ u) = v$, namely by choosing $f$ to be the constant function $\lambda u_{\iota}\text{.}\ w_{\iota}$.

We now concentrate on the introduced lemma in node 3 and first apply *implication introduction* rule. This introduces the assumption $\neg\exists w_{\iota}\text{.}\ w \in \mathit{set2}$ as a hypothesis from which we have to derive that $\mathit{set2} = \emptyset$. On this open subgoal we apply the functional extensionality principle (two functions are equal, iff they are equal with respect to all of their arguments) introducing $\forall x_{\iota}\text{.}\ (\mathit{set2}\ x) = (\emptyset\ x)$ (node 10) as new open node. The tactic *equiv2=* (see the justification in figure 2), which implements the extensionality principle on truth values (equality relation coincides with equivalence relation on truth values), can now be applied backwards to the subterm $(\mathit{set2}\ x) = (\emptyset\ x)$ introducing the open proof line $\forall x_{\iota}\text{.}\ (\mathit{set2}\ x) \equiv (\emptyset\ x)$. Note that the remaining subproblem, namely to show that the latter proof line can be derived from the assumption $\neg\exists w_{\iota}\text{.}\ w \in \mathit{set2}$, is a first order problem, provided the definition of $\emptyset$ is expanded: $\forall x_{\iota}\text{.}\ (\mathit{set2}\ x) \equiv \bot$. Thus we could also close this subproblem by calling a first order prover, but again we use TPS. The complete proof

[3] We use a TPS-like notation. Types are printed as subscripts and we write functional types as $\beta\alpha$ instead of using the more common notation $\alpha \to \beta$. The types $o$ and $\iota$ denote the sets of propositions and individuals. Dots bracket as far to the right as consistent with structure of the formula and the logical connectives.

[4] Sets are represented as characteristic functions and $\bot$ denotes *false*.

on an abstract proof level is shown in figure 2 while the original TPS proof for node 13, that can also be displayed within ΩMEGA, is presented in figure 4.

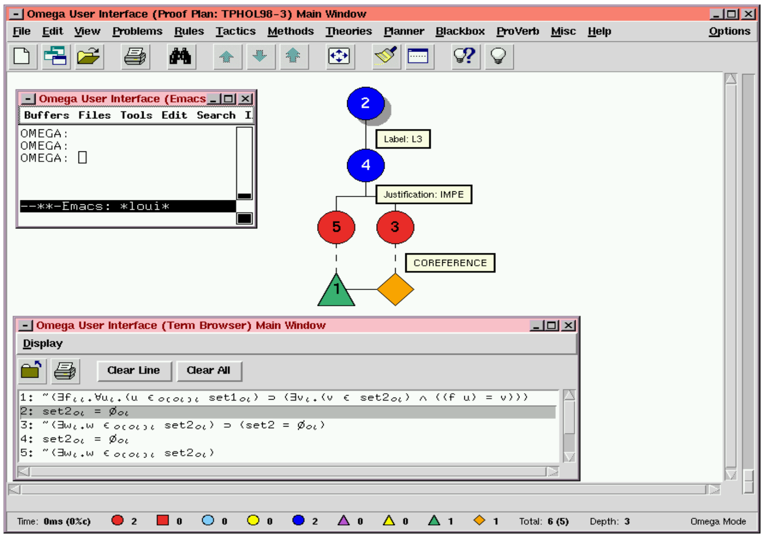


**Fig. 1.** The partial proof tree displayed in reverse order. The original proof goal (node 2) is presented at the top and the assumption (triangle-node 1) at the bottom. The formula contents of all numbered nodes are displayed in the term browser. A nodes status is symbolized via the nodes shape and color, e.g. assumptions and hypotheses are represented as green and magenta triangles, whereas all other nodes are represented as colored circles. Although this print may be black and white, we assume that the reader might be able to notice at least different shades and shapes of the nodes in the presented figures. Red circles (e.g. nodes 3 and 5) denote open subgoals and light or dark blue ones already justified nodes. Whereas the dark blue color indicates that a node is grounded, i.e. justified by a *ND-Rule* (node 2 and 4), a light blue color indicates that a node is justified by a *tactic*, which can be further expanded by ΩMEGA's tactic mechanism in order to obtain a completely ND-Rule based proof. Rhombi indicate coreferences to whole subtrees of a given partial proof in order to omit redundancy in the graphical display.

### 2.2 TPS as an Interactive Prover

TPS can also be used as an interactive theorem proving environment connected to ΩMEGA. For that TPS can be invoked from within ΩMEGA in a separate window and initialized with a specified subproblem. When a proof or a partial proof has been constructed within TPS, the user can

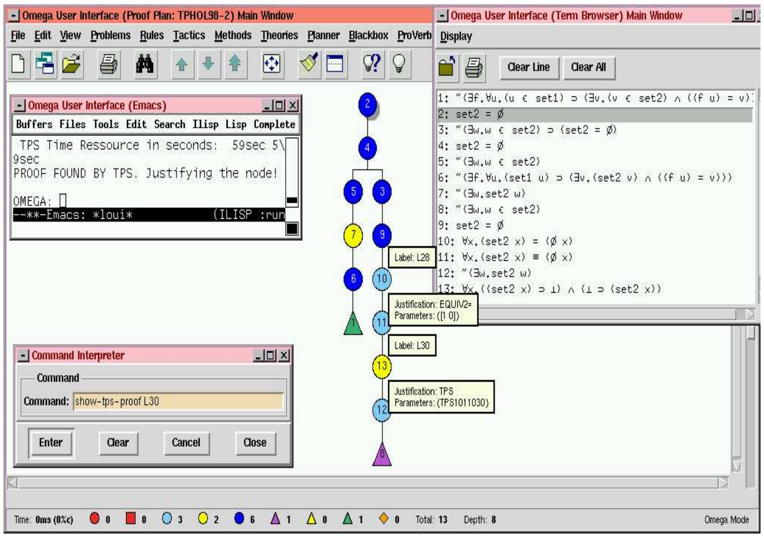


**Fig. 2.** The completed proof: Note that the subgoals in node 7 and node 13 are now justified by ΩMEGA's black box tactic *TPS*. The light blue color of this nodes indicate that they can be further expanded into proofs on a more detailed level. The paramater in the TPS-justification of node 13 refers to a file containing the proof output generated by TPS.

| | | | | |
|---|---|---|---|---|
| (2) | 2 | $\vdash$ | $\sim\exists F_{\iota\iota}.\forall U_{\iota}.SET1_{o\iota}\ U \supset \exists V_{\iota}.SET2_{o\iota}\ V \wedge F\ U = V$ | Hyp |
| (3) | 2 | $\vdash$ | $\forall F_{\iota\iota}.\sim\forall U_{\iota}.SET1_{o\iota}\ U \supset \exists V_{\iota}.SET2_{o\iota}\ V \wedge F\ U = V$ | Neg: 2 |
| (4) | 2,4 | $\vdash$ | $\exists W_{\iota}\ SET2_{o\iota}\ W$ | Hyp |
| (5) | 2,4,5 | $\vdash$ | $SET2_{o\iota}\ W_{\iota}$ | Choose: $W_{\iota}$ 4 |
| (6) | 2 | $\vdash$ | $\sim\forall U_{\iota}.SET1_{o\iota}\ U \supset \exists V_{\iota}.SET2_{o\iota}\ V \wedge [\lambda u_{\iota}\ W_{\iota}]\ U = V$ | UI: $[\lambda u_{\iota}\ W_{\iota}]$ 3 |
| (7) | 2 | $\vdash$ | $\sim\forall U_{\iota}.SET1_{o\iota}\ U \supset \exists V_{\iota}.SET2_{o\iota}\ V \wedge W_{\iota} = V$ | Lambda: 6 |
| (8) | 2 | $\vdash$ | $\exists U_{\iota}.\sim.SET1_{o\iota}\ U \supset \exists V_{\iota}.SET2_{o\iota}\ V \wedge W_{\iota} = V$ | Neg: 7 |
| (9) | 2,4,5,9 | $\vdash$ | $\sim.SET1_{o\iota}\ U_{\iota} \supset \exists V_{\iota}.SET2_{o\iota}\ V \wedge W_{\iota} = V$ | Choose: $U_{\iota}$ 8 |
| (10) | 2,4,5,9 | $\vdash$ | $SET1_{o\iota}\ U_{\iota} \wedge \sim\exists V_{\iota}.SET2_{o\iota}\ V \wedge W_{\iota} = V$ | Neg: 9 |
| (11) | 2,4,5,9 | $\vdash$ | $\sim\exists V_{\iota}.SET2_{o\iota}\ V \wedge W_{\iota} = V$ | RuleP: 10 |
| (12) | 2,4,5,9 | $\vdash$ | $\forall V_{\iota}.\sim.SET2_{o\iota}\ V \wedge W_{\iota} = V$ | Neg: 11 |
| (13) | 2,4,5,9 | $\vdash$ | $\sim.SET2_{o\iota}\ W_{\iota} \wedge W = W$ | UI: $W_{\iota}$ 12 |
| (14) | 2,4,5,9 | $\vdash$ | $SET2_{o\iota}\ W_{\iota}$ | Same as: 5 |
| (15) | 2,4,5,9 | $\vdash$ | $W_{\iota} = W$ | Assert REFL= |
| (16) | 2,4,5,9 | $\vdash$ | $SET2_{o\iota}\ W_{\iota} \wedge W = W$ | RuleP: 14 15 |
| (17) | 2,4,5,9 | $\vdash$ | $\bot$ | NegElim: 13 16 |
| (18) | 2,4,5 | $\vdash$ | $\bot$ | RuleC: 8 17 |
| (19) | 2,4 | $\vdash$ | $\bot$ | RuleC: 4 18 |
| (20) | 2 | $\vdash$ | $\sim\exists W_{\iota}\ SET2_{o\iota}\ W$ | NegIntro: 19 |

**Fig. 3.** The original TPS-proof for node 7.

| | | | | |
|---|---|---|---|---|
| (1) | 1 | $\vdash$ | $\sim\exists W_{\iota}\ SET2_{o\iota}\ W$ | Hyp |
| (2) | 2 | $\vdash$ | $SET2_{o\iota}\ X_{\iota}$ | Hyp |
| (3) | 1 | $\vdash$ | $\forall W_{\iota}.\sim SET2_{o\iota}\ W$ | Neg: 1 |
| (4) | 1 | $\vdash$ | $\sim SET2_{o\iota}\ X_{\iota}$ | UI: $X_{\iota}$ 3 |
| (5) | 1,2 | $\vdash$ | $\bot$ | RuleP: 2 4 |
| (6) | 1 | $\vdash$ | $SET2_{o\iota}\ X_{\iota} \supset \bot$ | Deduct: 5 |
| (7) | 1 | $\vdash$ | $[SET2_{o\iota}\ X_{\iota} \supset \bot] \wedge .\bot \supset SET2\ X$ | RuleP: 6 |
| (8) | 1 | $\vdash$ | $\forall X_{\iota}.[SET2_{o\iota}\ X \supset \bot] \wedge .\bot \supset SET2\ X$ | UGen: $X_{\iota}$ 7 |

**Fig. 4.** The original TPS-proof for node 13.

send it back to the ΩMEGA-system, where the results are integrated as subproof into the overall ΩMEGA proof.

As the TPS system provides plenty of very interesting interactive features for a user making it possible to solve many non-trivial problems either fully automatic or with little interaction. The ΩMEGA system will gain much from the integration of TPS as an interactive theorem proving environment of its own right.

Note that the power of TPS in proving theorems fully automatically is heavily influenced by availability of a comprehensive library providing useful information such as definitions and proof problems in connection to appropriate modes (lists of flag settings). Thus, a steady enrichment and maintenance of a TPS-library closely connected to ΩMEGA's theory database can steadily increase the power of TPS in proving problems as an automatic tool in ΩMEGA. In section 4 we present a way of providing TPS with information from ΩMEGA's knowledge base.

## 3 A Tactic-Based Proof integration

ΩMEGA's main philosophy is that all integrated systems have to generate enough protocol information, such that either a proof or proof plan, i.e. a proof on an abstract level, can be extracted from this information. Proof plans can then be expanded into a pure and checkable ND-level proof. For instance when calling a first order automated theorem prover, returned proofs are transformed into an intermediate data structure, the refutation graph, which is then translated into a high level ΩMEGA-proof plan, consisting of methods, tactics or rules. These proof plans are sound if they can be expanded successfully onto ND-rule level.

The integration of TPS is based on a much simpler proof transformation approach, which becomes possible as TPS itself transforms the automatically generated connections into a TPS-ND-level proof. This means that most but not all justifications refer to a standard ND-rule. There exist for instance justifications such as *RuleP* or *RuleC* where the first abbreviates simple derivations in propositional logic, while the latter is a slight modification of the standard exists elimination rule.

### 3.1 Integration and Expansion

The general idea is to define a new theory `TPS` in ΩMEGA which provides exactly one ΩMEGA-tactic for each possible TPS-justification. Using the tactics of this special theory each proof generated by TPS can now be mapped one to one onto an ΩMEGA proof. As a consequence each TPS-ND-proof can be visualized in its original form within ΩMEGA's graphical user interface LOUI. For instance by expanding the TPS-justified node 7 of figure 2 the original TPS-proof can be visualized in ΩMEGA. See figure 5 in comparison to the original TPS-proof in figure 3. Note that the TPS-proof is now embedded as a subproof into the whole proof. Thus line 2 is no longer a hypothesis but a derived line itself and line 20 is used to justify other proof lines.

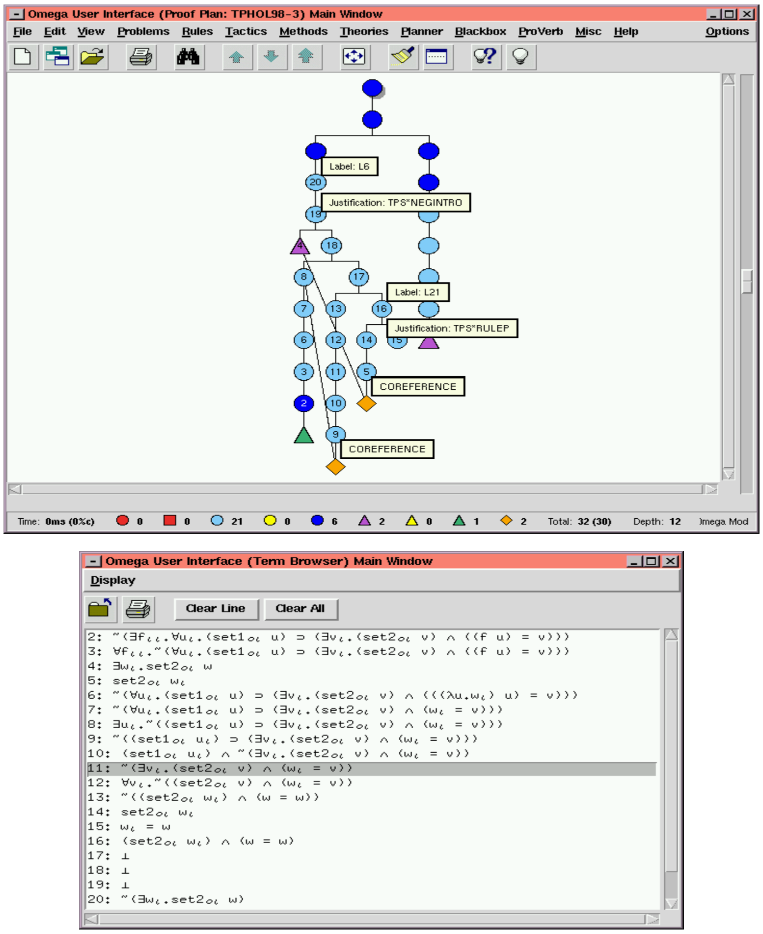


**Fig. 5.** The original TPS-proof for the first subproblem.

Each TPS-tactic of ΩMEGA's `TPS` theory contains an expansion information, which maps this TPS-tactic to a derivation build upon the rules and tactics in ΩMEGA's `base` theory. The ΩMEGA system allows to execute this transformation interactively line by line or at once for all lines. Furthermore the expansion of a line is reversible, meaning in every stage of the expansion one can get back to the original TPS proof (or even the state before the TPS proof was inserted), by unexpanding nodes.
We now discuss the different types of mappings:

**1:1 mapping** Clearly for many TPS-justifications there exist direct counterparts among ΩMEGA's ND-rules and tactics. For instance *tps*NegIntro* is mapped to the ND-rule *NotI* and *tps*Imp-Disj* is mapped to the ΩMEGA-tactic *EquivI* which itself can be further expanded on ΩMEGA's ND-rule level. Such simple mappings are the standard case as many justifications used in TPS have direct counterparts among ΩMEGA's ND-rules or tactics.

**Mappings with case distinctions** As a typical example we consider the TPS justification *Neg*, which is used as justification for an *PushNeg* application (pushing an outermost negation symbol inside a term) as well as for an *PullNeg* application (pulling a negation symbol at outermost position). Thus, depending on +the structure of the premise and conclusion line the justification *tps*Neg* gets translated in one of the ΩMEGA-tactics *PushNeg* or *Pullneg*.

**Restructuring mapping** The TPS-rule *RuleC* is a slight modification of the standard exists elimination rule, which roughly explained does not introduce the concrete instance of the existentially quantified line as a new hypotheses but instead introduces an analogous derived line justified with a special judgment. The point is that in this case, it is necessary to slightly manipulate the proof data structure while mapping *RuleC* to ΩMEGA's rule *ExtE*, i.e. the status of the instantiated line and some dependencies between proof lines have to be modified.

**External system mapping** Proof lines in TPS justified with *RuleP* abbreviate simple derivations in propositional logic which are trivial and would rather worsen the readability of the whole proof. Examples for the usage of *RuleP* are given in the two subproofs automatically proven by TPS in figures 3 and 4.
One approach to translate lines justified by *RuleP* into an ΩMEGA-proof would be to implement a simple propositional logic proof procedure as a recursive tactic (or better tactical) in ΩMEGA. While this is certainly possible it would contradict one important aspect of ΩMEGA. The aspect is to make use of other probably more specialized external reasoning systems as soon as this seems to be appropriate and thus to avoid unnecessary reimplementations. Hence instead of implementing the translation we just call another one of the integrated automated theorem provers

to expand lines justified by *RuleP*. As at the present time there is no theorem prover purely specialized on propositional logic integrated in $\Omega$MEGA. We choose to use OTTER as one of the available first order provers. Therefore the *RuleP* tactic is mapped as expansion onto a tactic specifying the call of OTTER which in turn produces a subproof when executed.

Figure 6 shows the expansion of *RuleP* via OTTER into the *AndI*-rule. In this case of a trivial derivation it seems to be an overkill to call an external reasoner. But firstly not all *RuleP* applications are that simple. One might consider line 7 in figure 4 as a more complicate situation. And on the other hand calls of OTTER are fast enough, even with respect to all necessary translations, that they do not slow down the expansion process considerably.

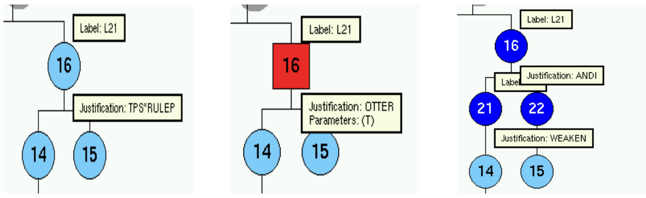


**Fig. 6.** Expansion of the *TPS*RuleP* tactic.

The described transformation approach based on $\Omega$MEGA's tactic mechanism allows to integrate arbitrary proofs generated by TPS into $\Omega$MEGA. The whole proof finally can be expanded on $\Omega$MEGA's ground level, i.e. a derivation using only rules assigned to $\Omega$MEGA's basic ND-calculus. The grounded proof for our example consists of more than 300 nodes and is shown in figure 7. This large number of single lines is due to the fact that $\Omega$MEGA's basic ND-calculus is rather small — in fact the idea is to minimize the basic calculus while upholding completeness. For example the basic calculus does not a priori include equality as this is a concept defined via Leibniz-equality, i.e. two functions are equal, iff they share the same properties. Therefore some tactics like equality substitution *=subst* expand into very large and tedious ground level derivations. Currently OMEGA has been applied to examples with atmost 1000 nodes (prooflines) at ground level, but we believe that the current system is able to handle even examples up to 10000 or more nodes on ground level.

Furthermore there is currently no advanced *cleanup* function available, which restructures the proof while eliminating most of the redundant and superfluous nodes in the proof tree. This is due to the fact that the problems arising from such deletions inside the three-dimensional proof data structure and their effects on the reuse of information from proof constructions in a planning scenario have not yet been completely solved.

### 3.2 Verbalization

Besides the possibility of graphically displaying proofs, as another feature from its integration with $\Omega$MEGA TPS gains the chance of verbalizing its proofs. The verbalization is done with the help of the PROVERB system [HF97] which is connected to $\Omega$MEGA and can be called within the graphical user interface. PROVERB was originally developed to translate proofs in first order logic into natural language and is currently extended to cover higher order logic as well. For that reason some of the verbalized higher order logic proofs still lack conciseness.
We demonstrate the use of PROVERB by giving the automatically generated natural language proof for the subproblem shown in figure 3.
**Assumptions:**
(1) $\nexists f.\ \forall u.\ set_1(u) \Rightarrow \exists v.\ set_2(v)$ and $f(u) = v$.

**Theorem:** $\nexists x.\ set_2(x)$. **Proof:** Let $\exists x.\ set_2(x)$. Let $x'$ be such $x$.

$\forall f.\ \not\forall u.\ set_1(u) \Rightarrow \exists v.\ set_2(v)$ and $f(u) = v$ because $\not\exists f.\ \forall u.\ set_1(u) \Rightarrow \exists v.\ set_2(v)$ and $f(u) = v$. We choose $\lambda\ z.x'$ for $f$. $\not\forall u.\ set_1(u) \Rightarrow \exists v.\ set_2(v)$ and $x' = v$ since $\not\forall u.\ set_1(u) \Rightarrow \exists v.\ set_2(v)$ and $(\lambda\ z.x')u = v$.
Let $set_1(u) \not\Rightarrow \exists v.\ set_2(v)$ and $x' = v$.
It isn't the case that we have $set_2(x')$ and $x' = x'$. $x' = x'$. We have a contradiction since we have $set_2(x')$ and $x' = x'$.
$\nexists x.\ set_2(x)$ since we have a contradiction. ■
Although comparable textbook proofs might be much shorter we point out that considering the size of the actual ground-level proof it is already a rather concise and abstract presentation. Moreover the verbalization models the overall proof idea to derive a contradiction by instantiating $f$ depending on $x'$ as an element of $set_2$.

## 4 Using $\Omega$MEGA's Knowledgebase

One of the features allowing TPS to automatically prove many theorems in higher order logic is the ability of selectively instantiating definitions. Indeed some theorems containing definitions can be proved with TPS using the dual instantiation strategy, whereas they cannot be automatically proven when all definitions are fully expanded [BA98].
In $\Omega$MEGA mathematical knowledge is structured into a hierarchy of theories where a theory can inherit from one or several parent theories. Each theory contains declarations of signature, axioms and definitions, where the latter can be viewed as abbreviations of more complex concepts. Moreover tactics, planning methods, linguistic knowledge and control strategies guiding the planner can be associated with each theory. Theorems are always declared within the context of a theory and can be presented concisely when using given definitions.
In order to enable TPS to prove certain subgoals containing concepts unknown to TPS it would be necessary to expand those definitions completely before sending the problem to TPS. Yet this would not only mean that the respective formulas might grow to a size intractable by TPS but

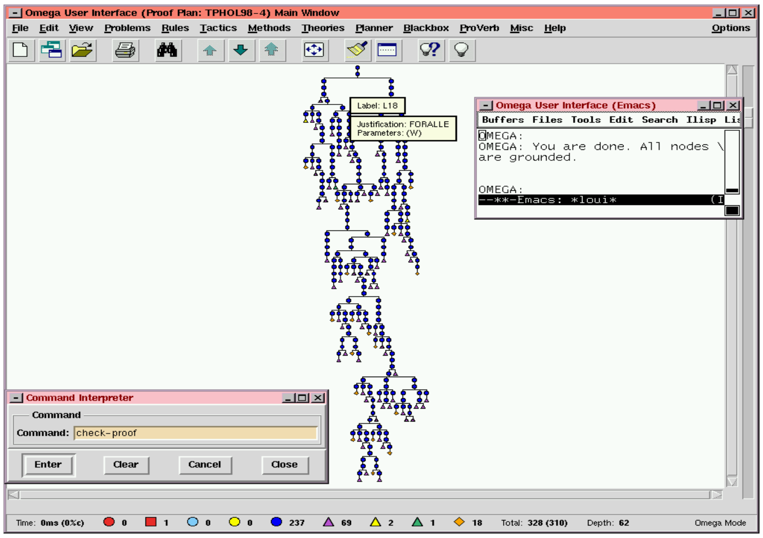


**Fig. 7.** The grounded proof tree

we would also prevent TPS from using its mechanism for selectively instantiating definitions. Thus it is necessary to transfer definitions of used concepts from $\Omega$MEGA to TPS. This can be achieved by using TPS' built-in library mechanism.

While in $\Omega$MEGA all objects, i.e. axioms, definitions, theorems, tactics, etc., a priori are associated with an existing theory, in TPS theories are created in order to group objects — thereby also creating hierarchies by specifying one theory as object of another — but a single object does not necessarily depend on a theory. Therefore we can map $\Omega$MEGA definitions onto corresponding TPS abbreviations by either

(A) transferring single concepts together with all related definitions and axioms into the TPS library, or by

(B) constructing a TPS library that mirrors both hierarchical structure and objects of the $\Omega$MEGA knowledgebase.

So far we have implemented an ad hoc version of approach (A). The underlying algorithm is stated in figure 8. However, we believe that with an extension of this algorithm goal (B), i.e. the transfer of the whole $\Omega$MEGA knowledgebase into a TPS library, can also be achieved[5] With the algorithm new definitions are stepwise expanded and inserted, together with their underlying concepts, into the library.

In order to demonstrate the working scheme of the algorithm and exemplify the translation of concepts, we consider the example theorem

$$\forall X_{o\iota}. \emptyset \in \mathfrak{P}(X). \tag{1}$$

[5] A translation of the whole $\Omega$MEGA knowledgebase into a TPS library should certainly be done only by an explicit call of the transformation process instead of an automatic translation with each passing of new theorem.

1. get all concepts used in the $\Omega$MEGA formula
2. for each concept $\mathcal{C}$ do:
   - if concept $\mathcal{C}$ already exists in the TPS library proceed with next concept
   - if it does not yet exist, then:
     - get definition of $\mathcal{C}$ from the $\Omega$MEGA knowledge base
     - extract all concepts $\mathcal{C}_1, \ldots, \mathcal{C}_n$ occurring in definition of $\mathcal{C}$
     - retrieve all axioms $\mathcal{A}_1, \ldots, \mathcal{A}_m$ revering to $\mathcal{C}$ from the $\Omega$MEGA knowledge base and insert them as formulas into the TPS library
     - insert TPS abbreviation of $\mathcal{C}$ into the TPS library specifying the dependency with respect to $\mathcal{C}_1, \ldots, \mathcal{C}_n$ and $\mathcal{A}_1, \ldots, \mathcal{A}_m$
     - apply step 2 to $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$
     - proceed with next concept

**Fig. 8.** Algorithm for transferring $\Omega$MEGA definitions into TPS libraries

(1) contains three definitions from $\Omega$MEGA's theory `naive-set`. Explicitly these are $\emptyset, \in$, and $\mathfrak{P}$, where $\emptyset$ and $\in$ are defined as in section 2 and $\mathfrak{P}$, specifying the powerset of a set $X$, can be expressed as the polymorphic $\lambda$-term:

$$\mathfrak{P} \doteq \lambda X_{o\beta} \centerdot \lambda Y_{o\beta} \centerdot Y \subseteq X$$

As both $\emptyset$ and $\in$ do not depend on any further definitions or axiomatization they can directly be translated into the TPS library. The definition of $\mathfrak{P}$ however depends on $\subseteq$ which expresses the concept that $Y$ is a subset of $X$. Therefore the abbreviation for powerset in the TPS library is defined with respect to the concept of subset, which in turn fetched from $\Omega$MEGA's knowledge base[6]. There $\subseteq$ is defined by

$$\subseteq \doteq \lambda X_{o\beta} \centerdot \lambda Y_{o\beta} \centerdot \forall Z_{\beta} \centerdot X(Z) \Rightarrow Y(Z)$$

and, as it neither contains any other abbreviations nor there exist axioms referring to it, can be translated into the TPS library.

## 5 Conclusion and Future Work

We reported about the integration of the higher order theorem prover TPS into the mathematical assistant system $\Omega$MEGA. As a result TPS can now be used to either automatically or interactively construct proofs,

[6] Both powerset and subset are already built-in abbreviations of TPS. Yet we used them here for the sake of simplifying the examples.

which then get translated and integrated into ΩMEGA's proof data structure. The tactic based proof transformation approach makes use of a special ΩMEGA theory `TPS` providing ΩMEGA-tactics for each possible TPS-justification. These tactics can subsequently be expanded onto ΩMEGAs calculus level, thereby enabling the graphical representation and the verbalization of TPS proofs on different levels of abstraction. In order to use TPS' reasoning power most efficiently we presented a way of translating and importing facts from ΩMEGAs knowledge base into TPS libraries. All those features have been presented in this paper with small examples.

We admit that the translation of *RuleP* by mapping it to a call of the first order theorem prover OTTER is somewhat an overkill. Even if it costs only a little additional time and is done fully automatically, we should at least intercept very trivial cases by mapping those immediately to appropriate tactics or rules. Furthermore we should replace OTTER by a pure propositional logic prover such as SATO [Zha97] as soon as it is integrated in ΩMEGA. Nevertheless we want to point out that this translation strategy demonstrates an interesting feature of ΩMEGA in general: All external reasoners already integrated in ΩMEGA can be used to support the integration of new systems, e.g. to close gaps when translating protocols of the new systems. Consequently as more specialized systems are being integrated with ΩMEGA the less detailed protocols may be required from the new systems.

The integration work we have done so far is essentially restricted to modifications of the ΩMEGA system and it seems to be promising to modify TPS as well, probably enabling a bidirectional communication between the two systems. As TPS also provides a tactic mechanism it seems to be plausible that an analogous translation from one of ΩMEGA's abstract proof levels into a TPS proof can be developed. Thereby external reasoners integrated to ΩMEGA as well as its internal facilities, would automatically become available to TPS. Surely TPS does not provide a three-dimensional proof data structure and hence ΩMEGA proofs cannot be mirrored in their original form within TPS, but this should not impose a serious problem for the cooperation of both systems.

Although the question about an intelligent automatic cooperation between both systems is open a combined system will at least gain additional deductive power when used in an interactive mode where the user controls the overall strategy. The authors currently work on an integration of higher order theorem provers into ΩMEGA's proof planning framework. Examples, such as proving the equivalence of two different mathematical definitions (e.g. a group definitions), demonstrate that both systems have complementary strengths: Especially the decomposition of such proof problems into subtasks can easily be handled by ΩMEGA's proof planning framework, whereas most of the particular subtasks might be solvable automatically by TPS, when using its mechanism for selectively instantiating the definitions. Currently neither ΩMEGA nor TPS is able to prove such theorems fully automatically, but their combination most likely is. Certainly the development of an advanced common theory and library mechanism will be an important prerequisite.

Two other interesting questions are: First, if TPS' automatic search process and ΩMEGA's built-in logical engine LEO [BK98], which is a resolution based higher order theorem prover specialized in an appropriate treatment of the extensionality principles, could cooperate successfully. And second, whether the combination of both higher order theorem provers could be a even more promising support for $\Omega$MEGA's proof planner.

**Acknowledgements** We thank Matthew Bishop and Peter Andrews for supporting our integration work and for providing us with important information about TPS. Furthermore we thank Holger Gebhard for his aid with the implementation and Serge Autexier for stimulating remarks.

## References

[ABI+96] Peter B. Andrews, Matthew Bishop, Sunil Issar, Dan Nesmith, Frank Pfenning, and Hongwei Xi. TPS: A Theorem Proving System for Classical Type Theory. *Journal of Automated Reasoning*, 16(3):321–353, 1996.

[AIN+97] Peter B. Andrews, Sunil Issar, Dan Nesmith, Hongwei Xi, Frank Pfenning, and Matthew Bishop. *TPS3 Facilities Guide for Programmers and Users*, 1997.

[And89] Peter B. Andrews. On Connections and Higher Order Logic. *Journal of Automated Reasoning*, 5:257–291, 1989.

[BA98] Matthew Bishop and Peter B. Andrews. Selectively Instantiating Definitions. *Proceedings of the 15th Conference on Automated Deduction*, Lindau, Germany, 1998. Springer, Berlin, Germany.

[BCF+97] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann, and V. Sorge. $\Omega$Mega: Towards a Mathematical Assistant. In W. McCune, editor, *Proceedings of the 14th Conference on Automated Deduction*, LNAI 1249, Townsville, Australia, 1997. Springer, Berlin, Germany.

[BF94] Peter Baumgartner and Ulrich Furbach. PROTEIN: A PROver with a Theory Extension INterface. In A. Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction*, LNAI 814, pages 769–773, NancyFrance, 1994. Springer, Berlin, Germany.

[BK98] Christoph Benzmüller and Michael Kohlhase. LEO, a Higher-Order Theorem Prover. *Proceedings of the 15th Conference on Automated Deduction*, Lindau, Germany, 1998. Springer, Berlin, Germany.

[FH97] A. P. Felty and D. J. Howe. Hybrid Interactive Theorem Proving Using Nuprl and HOL. In W. McCune, editor, *Proceedings of the 14th Conference on Automated Deduction*, LNAI 1249, pages 351–365, Townsville, Australia, 1997. Springer, Berlin, Germany.

[Gen35] G. Gentzen. Untersuchungen über das Logische Schließen I und II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

[GPT96] Fausto Giunchiglia, Paolo Pecchiari, and Carolyn Talcott. Reasoning Theories – Towards an Architecture for Open Mechanized Reasoning Systems. In Franz Baader and Klaus U. Schulz, editors, *Frontiers of combining systems*, *Applied logic series* 3, pages 157 – 174, 1996. Kluwer Academic Publishers.

[HF97] Xiaorong Huang and Armin Fiedler. Proof Verbalization in *PROVERB*. In *Proceedings of the First International Workshop on Proof Transformation and Presentation*, pages 35–36, Schloss Dagstuhl, Germany, 1997.

[HKRS94] X. Huang, M. Kerber, J. Richts, and A. Sehn. Planning Mathematical Proofs with Methods. *Journal of Information Processing and Cybernetics*, 30(5/6):277–291, 1994.

[KKS98] Manfred Kerber, Michael Kohlhase, and Volker Sorge. Integrating Computer Algebra into Proof Planning. *Journal of Automated Reasoning*, 1998. forthcoming.

[McC94] W. McCune. Otter 3.0 Reference Manual and Guide. Technical Report ANL-94-6, Argonne National Laboratory, Argonne, Illinois 60439, USA, 1994.

[Mei97] Andreas Meier. Übersetzung automatisch erzeugter Beweise auf Faktenebene. Master's thesis, Computer Science Department, Universität des Saarlandes, Saarbrücken, Germany, 1997.

[Mil84] Dale Miller. Expansion Tree Proofs and Their Conversion to Natural Deduction Proofs. In R. E. Shostak, editor, *Proceedings of the 7th International Conference on Automated Deduction*, LNCS 170, pages 375–303, Napa, CA, USA, 1984. Springer, Berlin, Germany.

[Pfe87] F. Pfenning. *Proof Transformations in Higher-Order Logic.* PhD thesis, Carnegie-Mellon University, Pittsburgh Pa., 1987.

[SHB+98] J. Siekmann, S. M. Hess, C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, M. Kohlhase, K. Konrad, E. Melis, A. Meier, and V. Sorge. LOUI: A Distributed Graphical User Interface for the ΩMEGAProof System. Submitted to the International Workshop on User Interfaces for Theorem Provers, 1998.

[WGR96] Christoph Weidenbach, Bernd Gaede, and Georg Rock. SPASS & FLOTTER, version 0.42. In M. A. McRobbie and J. K. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction*, LNAI 1104, New Brunswick, NJ, USA, 1996. Springer, Berlin, Germany.

[Zha97] H. Zhang. SATO: An Efficient Propositional Prover. In W. McCune, editor, *Proceedings of the 14th Conference on Automated Deduction*, LNAI 1249, pages 272–275, Townsville, Australia, 1997. Springer, Berlin, Germany.