Proving Theorems of Type Theory Automatically with TPS

Peter B. Andrews
Department of Mathematical Sciences
Carnegie Mellon University
Pittsburgh, PA, U.S.A.
andrews@cmu.edu
http://gtps.math.cmu.edu/andrews.html

Reasoning plays an important role in many activities which involve intelligence, and it may be anticipated that automated reasoning will play a significant role in many applications of artificial intelligence. The importance of developing methods of automating reasoning has been recognized since the inception of research on artificial intelligence. One fruitful approach to this problem is to use the language and methods of symbolic logic. Since a great variety of problems can be expressed in symbolic logic, progress in developing general purpose reasoning tools based on symbolic logic has the potential to contribute to progress in many realms of artificial intelligence.

Work on automated deduction using symbolic logic has been progressing steadily, but in recent years such work has been presented primarily at conferences on automated deduction (such as (Kirchner & Kirchner 1998; Ganzinger 1999; McAllester 2000; Voronkov 2002; Mayer & Pirri 2003; David Basin 2004)) rather than at conferences on artificial intelligence. While such work often focuses on proving theorems, it should be noted that procedures for automatically proving theorems can play crucial roles as inference mechanisms in more general automated reasoning tools.

We provide a demonstration of the TPS automated Theorem Proving System. This system can be used to prove theorems of type theory, which is also known as higher-order logic and which includes first-order logic. In a practical sense type theory has greater expressive power than first-order logic, and it is well suited to the formalization of various disciplines, including mathematics and fields which use mathematics. For example, inductive definitions can be expressed in a very simple and natural way in type theory. Also, in type theory one can quantify over functions, such as functions mapping states to states.

Our demonstration includes explanations of the notations used by TPS, which are based on a formulation of type theory (the typed λ-calculus) which was introduced by Alonzo Church (Church 1940) and is explained further in the text (Andrews 2002).

TPS has been developed over several decades in collaboration with Dale Miller, Frank Pfenning, Sunil Issar, Carl Klapper, Dan Nesmith, Hongwei Xi, Matthew Bishop, and Chad E. Brown. TPS produces formal proofs (in natural deduction style) in first- and higher-order logic interactively, semi-automatically, and automatically. It excels at proving theorems of higher-order logic automatically (Bishop & Andrews 1998; Bishop 1999a; 1999b; Andrews, Bishop, & Brown 2000; Brown 2002; 2004).

A general discussion of automated theorem proving in type theory may be found in (Andrews 2001). In automatic mode, TPS first searches for an *expansion proof*, which expresses in a nonredundant way the fundamental logical structure underlying various proofs of the theorem, and then transforms this into a proof in natural deduction style.

The purely interactive commands of TPS for applying rules of inference are available in a separate program called ETPS (Educational Theorem Proving System) which is used by logic students to construct formal proofs.

Both TPS and ETPS are available from the web. For more information see (Andrews *et al.* 1996), (Andrews *et al.* 2004), (Andrews & Brown 2005), and http://gtps.math.cmu.edu/tps.html.

Examples

Most of the examples to which TPS has been applied thus far have a mathematical flavor, though nothing prevents it from being applied to other types of examples. Some examples of theorems which TPS can prove automatically are:

X5305:
$\forall s_{o\alpha} . \sim \exists g_{o\alpha\alpha} \forall f_{o\alpha} . f \subseteq s \supset \exists j_{\alpha} . s j \wedge g j = f$
This expresses Cantor's Theorem that every set s has more subsets than members. It asserts that there is no function g which maps s onto the collection of all subsets of s. TPS's search for a proof leads it to the same clever diagonal argument as was used by Cantor to prove this theorem.

THM15B: $\forall f_{\iota\iota} . \exists g_{\iota\iota} [\text{ ITERATE+ } f g \wedge \exists x_{\iota} . g x = x \wedge \forall z_{\iota} . g z = z \supset z = x] \supset \exists y_{\iota} . f y = y$
This theorem asserts that if some iterate of a function f has a unique fixed point, then f has a fixed point. An *iterate* of a function f which maps a set to itself is a function which applies f repeatedly. For example, if $g(x) = f(f(f(x)))$, then g is an iterate of f. [ITERATE+ fg] means that g is an iterate of f. ITERATE+ is defined (inductively) as $[\lambda f_{\alpha\alpha} \lambda g_{\alpha\alpha} \forall p_{o(\alpha\alpha)} . p f \wedge \forall j_{\alpha\alpha} [p j \supset p . f \circ j] \supset p g]$. A *fixed point* of f is an element y such that $f(y) = y$. In the process of proving THM15B, TPS applies its general logical procedures to produce an inductive proof that $g(f(x)) =$

$f(g(x))$ when g is an iterate of f. No knowledge or heuristics about induction or iterates are built into TPS except for the inductive definition of the set of iterates. TPS uses this result to conclude that if x is the unique fixed point of g (so $g(x) = x$), then $g(f(x)) = f(g(x)) = f(x)$, so $f(x)$ is also a fixed point of g, so $f(x) = x$, and f has a fixed point.

THM136:
$\forall r_{o\alpha\alpha}$TRANSITIVE [TRANSITIVE-CLOSURE r]
This asserts that the transitive closure of a relation is transitive. TRANSITIVE is defined as
$[\lambda R_{o\alpha\alpha} \forall X_\alpha \forall Y_\alpha \forall Z_\alpha . R X Y \wedge R Y Z \supset R X Z]$,
and TRANSITIVE-CLOSURE is defined as
$[\lambda r_{o\alpha\alpha} \lambda x_\alpha \lambda y_\alpha \forall p_{o\alpha\alpha} . r \subseteq p \wedge$TRANSITIVE $p \supset p x y]$.
When searching for proofs of theorems which contain definitions (abbreviations), it is a significant problem to decide which instances of the definitions to instantiate (expand). Often, one needs to instantiate some, but not all, of them, and if one does instantiate all of them, one can cause the search space to expand in a very undesirable way. This theorem provides an example of this problem. TPS finds a proof for this theorem by using a technique called *dual instantiation* (Bishop & Andrews 1998). It involves making each instance of a definition accessible to the search procedure in both its instantiated and its uninstantiated form, and letting the search procedure decide which to use, with a bias in favor of the uninstantiated form.

THM145B: [TRANSITIVE $\leq$] $\wedge \forall s_{o\alpha} [\forall z_\alpha [s z \supset z \leq U_{\alpha(o\alpha)} s] \wedge \forall j_\alpha . \forall k_\alpha [s k \supset k \leq j] \supset U s \leq j] \supset \forall f_{\alpha\alpha} . \forall x_\alpha \forall y_\alpha [x \leq y \supset f x \leq f y] \supset \exists w_\alpha . w \leq f w \wedge f w \leq w$
This asserts that in a complete lattice (with $[U_{\alpha(o\alpha)} s_{o\alpha}]$ as $lub\ s_{o\alpha}$), every monotone function has a fixed point. TPS finds the fixed point $[U_{\alpha(o\alpha)} . \lambda u_\alpha . u \leq f_{\alpha\alpha} u]$.

THM587: IND $\wedge$ PLUS-INDEQS $0_\iota S_{\iota\iota} \supset \forall x_\iota \forall y_\iota . [x + y] + y = x + [y + y]$
IND (the principle of induction) is defined as
$\forall p_{o\iota} . p 0_\iota \wedge \forall x_\iota [p x \supset p . S_{\iota\iota} x] \supset \forall x p x$, and PLUS-INDEQS (the inductive definition of $+$) is defined as $\lambda 0_\iota \lambda S_{\iota\iota} . \forall n_\iota [n + 0 = n] \wedge \forall n \forall m_\iota . n + S m = S[n + m]$. It is natural and appropriate to try to prove this by induction, but neither induction on x nor induction on y works without some auxiliary lemmas. TPS decides to derive $\forall u_\iota . [x_\iota + y_\iota] + u = x + [y + u]$ from the hypotheses by induction on u, and then derives the desired conclusion from this.

Acknowledgements

This material is based on work supported by NSF grants MCS81-02870, DCR-8402532, CCR-8702699, CCR-9002546, CCR-9201893, CCR-9502878, CCR-9624683, CCR-9732312, and CCR-0097179.

References

Andrews, P. B., and Brown, C. E. 2005. Tps: A hybrid automatic-interactive system for developing proofs. *Journal of Applied Logic* 3. to appear.

Andrews, P. B.; Bishop, M.; Issar, S.; Nesmith, D.; Pfenning, F.; and Xi, H. 1996. TPS: A theorem proving system for classical type theory. *Journal of Automated Reasoning* 16:321–353.

Andrews, P. B.; Brown, C. E.; Pfenning, F.; Bishop, M.; Issar, S.; and Xi, H. 2004. Etps: A system to help students write formal proofs. *Journal of Automated Reasoning* 32:75–92.

Andrews, P. B.; Bishop, M.; and Brown, C. E. 2000. System description: Tps: A theorem proving system for type theory. In McAllester (2000), 164–169.

Andrews, P. B. 2001. Classical type theory. In Robinson, A., and Voronkov, A., eds., *Handbook of Automated Reasoning*, volume 2. Elsevier Science. chapter 15, 965–1007.

Andrews, P. B. 2002. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof.* Kluwer Academic Publishers, second edition.

Bishop, M., and Andrews, P. B. 1998. Selectively instantiating definitions. In Kirchner and Kirchner (1998), 365–380.

Bishop, M. 1999a. A breadth-first strategy for mating search. In Ganzinger (1999), 359–373.

Bishop, M. 1999b. *Mating Search Without Path Enumeration.* Ph.D. Dissertation, Department of Mathematical Sciences, Carnegie Mellon University. Department of Mathematical Sciences Research Report No. 99–223.

Brown, C. E. 2002. Solving for set variables in higher-order theorem proving. In Voronkov (2002), 408–422.

Brown, C. E. 2004. *Set Comprehension in Church's Type Theory.* Ph.D. Dissertation, Department of Mathematical Sciences, Carnegie Mellon University.

Church, A. 1940. A formulation of the simple theory of types. *Journal of Symbolic Logic* 5:56–68.

David Basin, M. R., ed. 2004. *Automated Reasoning, Second International Joint Conference, IJCAR 2004*, volume 3097 of *Lecture Notes in Artificial Intelligence*. Cork, Ireland: Springer-Verlag.

Ganzinger, H., ed. 1999. *Proceedings of the 16th International Conference on Automated Deduction*, volume 1632 of *Lecture Notes in Artificial Intelligence*. Trento, Italy: Springer-Verlag.

Kirchner, C., and Kirchner, H., eds. 1998. *Proceedings of the 15th International Conference on Automated Deduction*, volume 1421 of *Lecture Notes in Artificial Intelligence*. Lindau, Germany: Springer-Verlag.

Mayer, M. C., and Pirri, F., eds. 2003. *Automated Reasoning with Analytic Tableaux and Related Methods. (TABLEAUX 2003)*, volume 2796 of *Lecture Notes in Artificial Intelligence*. Rome, Italy: Springer-Verlag.

McAllester, D., ed. 2000. *Proceedings of the 17th International Conference on Automated Deduction*, volume 1831 of *Lecture Notes in Artificial Intelligence*. Pittsburgh, PA, USA: Springer-Verlag.

Voronkov, A., ed. 2002. *Proceedings of the 18th International Conference on Automated Deduction*, volume 2392 of *Lecture Notes in Artificial Intelligence*. Copenhagen, Denmark: Springer-Verlag.