The TPS Theorem Proving System

Peter B. Andrews Sunil Issar Dan Nesmith Frank Pfenning
Carnegie Mellon University
Pittsburgh, PA 15213-3890, U.S.A.

TPS is a theorem proving system for first- and higher-order logic which runs in Common Lisp and can operate in automatic, semi-automatic, and interactive modes. As its logical language TPS uses the typed λ-calculus [6], in which most theorems of mathematics can be expressed very directly.

TPS can be used to search for an *expansion proof* [10, 11] of a theorem, which represents in a nonredundant way the basic combinatorial information required to construct a proof of the theorem in any style. TPS also has facilities based on the ideas in [10, 11, 12, 13, 14] for translating back and forth between expansion proofs and natural deduction proofs.

When one is seeking an expansion proof for a theorem of higher-order logic, not all necessary substitution terms can be generated by unification of formulas already present, so certain *expansion options* [5] are applied, and then a search for a p-acceptable mating [2] is made, using Huet's higher-order unification algorithm [8] to generate all remaining substitution terms. The expansion options consist of quantifier duplications and projective and primitive substitutions (such as those which substitute $[\lambda w_{o\alpha}.P^1_{o(o\alpha)}w \wedge P^2_{o(o\alpha)}w]$, $[\lambda w_{o\alpha}.P^3_{o(o\alpha)}w \vee P^4_{o(o\alpha)}w]$, $[\lambda w_{o\alpha}.w[P^5_{\alpha(o\alpha)}w]]$, $[\lambda w_{o\alpha}\exists x_\gamma.P^6_{o\gamma(o\alpha)}wx]$, and $[\lambda w_{o\alpha}\forall x_\gamma.P^7_{o\gamma(o\alpha)}wx]$ for a variable $P_{o(o\alpha)}$). These substitutions introduce a small amount of new structure, and contain variables for which additional substitutions can be made at a later stage.

Different sets of expansion options are applied to create different expansion trees which are all subtrees of a master expansion tree. Smaller subtrees are explored before larger ones in an attempt to keep the search space manageable. The sets of expansion options are generated in a systematic and exhaustive way, except that at present the types of quantified variables in primitive substitutions (such as γ above) are chosen from a small fixed set of types which is specified interactively. We conjecture that if this restrictive (but practical) method of specifying types were replaced by a general enumerative procedure, the search procedure implemented in TPS would be complete in principle for elementary type theory (the logical system of [1]).

The second author has developed a matingsearch procedure (called *path-focused duplication* [9]) in which quantifier duplications are localized to vertical paths (thus reducing the enormous growth in the number of paths which accompanies duplications), and the duplications for each path are generated as needed to span that path. The search space grows and shrinks dynamically as different vertical paths are considered.

TPS has various features designed to make it a versatile and friendly system. Many aspects of the program's behavior can be controlled by setting flags. There are over 150 of these flags, and

This work is supported by NSF grant CCR-8702699.

TPS has a facility called Review for examining and changing the settings of flags, and for defining and reusing groups of flag settings called *modes*. There is a formula editor which facilitates constructing new formulas from others already known to TPS. There is a library facility for saving formulas, definitions, and modes. Online documentation can be assembled automatically into a facilities guide which reflects the current state of the program. A program called ETPS containing the facilities of TPS for constructing natural deduction proofs interactively and many exercises from [3] is available for use by students in logic courses, and has been used extensively at our university.

References

[1] Peter B. Andrews, Resolution in type theory, *Journal of Symbolic Logic* 36 (1971), 414–432.

[2] Peter B. Andrews, Theorem proving via general matings, *Journal of the ACM* 28 (1981), 193–214.

[3] Peter B. Andrews, *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*, Academic Press, 1986.

[4] Peter B. Andrews, Typed λ-calculus and automated mathematics, In Edgar G. K. Lopez-Escobar David W. Kueker and Carl H. Smith, editors, *Mathematical Logic and Theoretical Computer Science*, Lecture Notes in Pure and Applied Mathematics, vol. 106, Marcel Dekker, 1987, 1–14.

[5] Peter B. Andrews, On connections and higher-order logic, *Journal of Automated Reasoning* 5 (1989), 257–291.

[6] Alonzo Church, A formulation of the simple theory of types, *Journal of Symbolic Logic* 5 (1940), 56–68.

[7] Gerard P. Huet, A mechanization of type theory, In *Third International Joint Conference on Artificial Intelligence*, 1973, 139–146.

[8] Gerard P. Huet, A unification algorithm for typed λ-calculus, *Theoretical Computer Science* 1 (1975), 27–57.

[9] Sunil Issar, Path-focused duplication: A search procedure for general matings, Technical report, Carnegie Mellon University, Pittsburgh, February 1990.

[10] Dale A. Miller, Expansion tree proofs and their conversion to natural deduction proofs, In *7th International Conference on Automated Deduction*, Lecture Notes in Computer Science 170, Springer-Verlag, 1984, 375–393.

[11] Dale A. Miller, A compact representation of proofs, *Studia Logica* 46,4 (1987), 347–370.

[12] Frank Pfenning, Analytic and non-analytic proofs, In *7th International Conference on Automated Deduction*, Lecture Notes in Computer Science 170, Springer-Verlag, 1984, 394–413.

[13] Frank Pfenning, *Proof Transformations in Higher-Order Logic*, PhD thesis, Carnegie Mellon University, 1987.

[14] Frank Pfenning and Dan Nesmith, Presenting intuitive deductions via symmetric simplification, In *10th International Conference on Automated Deduction*, July 1990.