Proving Theorems Automatically, Semi-Automatically, and Interactively with TPS

Peter B. Andrews

http://gtps.math.cmu.edu/tps.html

Developers of TPS:

Peter B. Andrews
Eve Longini Cohen
Dale A. Miller, Ph.D. 1983
Frank Pfenning, Ph.D. 1987
Sunil Issar, Ph.D. 1991
Dan Nesmith
Hongwei Xi, (Ph.D. 1998)
Matthew Bishop, Ph.D. 1999
Chad E. Brown

[$\sim$ **A**] means "**A** is not true";
[**A** $\wedge$ **B**] means "**A** and **B**";
[**A** $\vee$ **B**] means "**A** or **B**";
[**A** $\supset$ **B**] means "**A** implies **B**";
[**A** $\equiv$ **B**] means "**A** if and only if **B**";

When the relative scopes of several connectives of different kinds must be determined, $\sim$ is to be given the smallest possible scope, then $\wedge$ the next smallest possible scope except for $\sim$, then $\vee$, then $\supset$, then $\equiv$.

Bracket and Parenthesis Conventions

Outermost brackets and parentheses may be omitted.

Use the convention of association to the left for brackets and parentheses.

Thus $\alpha\beta\gamma$ stands for $((\alpha\beta)\gamma)$.

A dot stands for a left bracket, whose mate is as far to the right as is possible without altering the pairing of left and right brackets already present.

Some Useful Commands in TPS

HELP

?

BEGIN-PRFW and END-PRFW

LIST-RULES

PROVE

X2113:

$$\forall y \exists w\, R\, y\, w \wedge \exists z \forall x [P\, x \supset \sim R\, z\, x] \supset \exists x. \sim P\, x$$

Four proofs of X2113:

- Interactive
- Semi-interactive using GO2
- Semi-automatic using MONSTRO
- Automatic using DIY

Church's Type Theory

Alonzo Church,
"A Formulation of the Simple Theory of Types",
Journal of Symbolic Logic 5 (1940), 56-68.

$$Y_\alpha = F_{\alpha\beta}\, X_\beta$$

$(\alpha\beta)$ is the type of functions
to objects of type α
from objects of type β.

This is sometimes written $\beta \rightarrow \alpha$.

A function of two arguments can be represented as a function of one argument whose values are functions.

$$Z_\alpha = [[G_{((\alpha\beta)\gamma)} X_\gamma] Y_\beta] = G_{\alpha\beta\gamma} X_\gamma Y_\beta$$

An entity of type $((\alpha\beta)\gamma)$ may be regarded both as

a function mapping elements of type γ to functions of type $(\alpha\beta)$

and as

a function of two arguments (of types γ and β) which has values of type α.

o is the type of truth values and statements.

We identify a set of elements of type β with the function $S_{o\beta}$ which maps the elements in the set to truth and all other objects of type β to falsehood, and refer to $S_{o\beta}$ as a set. Thus:

$S_{o\beta}\,x_\beta$ means that $S_{o\beta}\,x_\beta$ is true.

$S_{o\beta}\,x_\beta$ means that $x_\beta \in S_{o\beta}$.

$S_{o\beta} = \{x_\beta \mid S_{o\beta}\,x_\beta\}$.

Similarly, $R_{o\beta\alpha}$ is a relation between objects of type α and objects of type β.

λ-Notation

If $F(v) = v^2 + v + 5$
for all natural numbers v,
then $F = [\lambda v \ . \ v^2 + v + 5]$

In general, $[\lambda v \ A(v)]$ denotes the function whose value for any argument v is $A(v)$.

If $A(v)$ is a statement about v,
$[\lambda v \ A(v)]$ denotes $\{v \mid A(v)\}$.

If $A(u, v)$ is a statement about u and v,
$[\lambda u \lambda v \ A(u, v)]$ denotes $\{< u, v > \ \mid A(u, v)\}$.

λ-Conversion

$[\lambda v \ . \ v^2 + v + 5]\, 7 = 7^2 + 7 + 5$

$[\lambda v A(v)]\, W = A(W)$

If $A(v)$ is a statement about v,
$[\lambda v A(v)]\, W$ means
$W \in \{v | A(v)\}$, or $A(W)$.

For more information about type theory, see: Peter B. Andrews, *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*, second edition, Kluwer Academic Publishers, 2002.

or take
21-700 Mathematical Logic II
(offered every spring)

X5203: $\# f_{\alpha\beta}[x_{o\beta} \cap y_{o\beta}] \subseteq \# f\, x \cap \# f\, y$

Semi-interactive proof with GO2.

X5308:

$$\exists j_{\beta(o\beta)} \forall p_{o\beta}[\exists x_{\beta}\, p\, x \supset p.\, j\, p]$$
$$\supset .\forall x_{\alpha} \exists y_{\beta}\, r_{o\beta\alpha}\, x\, y \equiv \exists f_{\beta\alpha} \forall x\, r\, x.\, f\, x$$

Semi-interactive proof with GO2. Use ED (the editor) to construct the wffs needed to instantiate quantifiers from wffs already present in the proof. Use DIY-L to fill in the gaps automatically.

Automatic proof.

The TPS Library and Classification System

LIB

LIST-OF-LIBOBJECTS

CLASS-SCHEME

UNIXLIB

LS

CD

LEAVE

REWRITING

LIB

LIST-OF-LIBOBJECTS
TYPE $>$ THEORY

FETCH THEO2

HELP THEO2

LEAVE

LIST-RRULES

BEGIN-PRFW

PROVE SUM3

SIMPLIFY-PLAN

SIMPLIFY-PLAN

SIMPLIFY-PLAN*

END-PRFW

The Injective Cantor Theorem

There is no injective function from the power set $\mathcal{P}(U)$ of a set U into U.

Informal Proof:

Suppose h maps $\mathcal{P}(U)$ into U.

Let $D = \{ht \,|\, t \in \mathcal{P}(U) \text{ and } ht \notin t\}$.

Clearly $D \subseteq U$ so $D \in \mathcal{P}(U)$.

We show that

(1) $hD \in D$;

(2) if h is injective, then $hD \notin D$.

Therefore, there is no such injection.

Proof of (1):

Suppose $hD \notin D$. Then

$D \in \mathcal{P}(U)$ and $hD \notin D$, so

$hD \in \{ht \mid t \in \mathcal{P}(U)$ and $ht \notin t\}$.

$hD \in D$ (by the definition of D).

Contradiction. Hence $hD \in D$.

Proof of (2):

Suppose h is injective.

Suppose $hD \in D$.

$hD \in \{ht \,|\, t \in \mathcal{P}(U) \text{ and } ht \notin t\}$
(by the definition of D).

Thus $hD = ht$ for some $t \in \mathcal{P}(U)$ such that $ht \notin t$.

h is injective, so $D = t$.

$ht \notin t$, so $hD \notin D$.

This is a contradiction, so we conclude that

if h is injective, then $hD \notin D$.

D is $\{ht \mid t \in \mathcal{P}(U)$ and $ht \notin t\}$,
which depends on h.

Define IDIAG to be
$\lambda h_{\iota(o\iota)} \lambda z_\iota \exists t_{o\iota}. \sim t[h\, t] \wedge z = h\, t$.

Then [IDIAG h] represents the set D.

The Injective Cantor Theorem

x5309A: $\sim \exists h_{\iota(o\iota)}$INJECTIVE h

Semi-automatic proof using DIY-L and two lemmas:

THM143D:
$\forall h_{\iota(o\iota)}$.INJECTIVE $h \supset \sim$ IDIAG $h. h$.IDIAG h

THM144B: $\forall h_{\iota(o\iota)}$IDIAG $h. h$.IDIAG h

THM587: IND $\wedge$ PLUS-INDEQS $_{o(\iota\iota)\iota}\, 0_\iota\, S_{\iota\iota} \supset$
$\forall x_\iota \forall y_\iota . x + y + y = x + . y + y$

TPS finds an automatic inductive proof for this, though neither induction on x nor induction on y works.

THM15B: $\forall f_{\iota\iota}. \exists g_{\iota\iota}[\text{ITERATE+} f\, g$
$\wedge \exists x_\iota. g\, x = x \wedge \forall z_\iota. g\, z = z \supset z = x]$
$\supset \exists y_\iota. f\, y = y$

Informal proof of THM15B:

Let x be the unique fixed point of g.

$g\, x = x$

$f\, [g\, x] = f\, x$

$g = f \circ \ldots \circ f$ so $f \circ g = g \circ f$.

$g\, [f\, x] = f\, x$

Thus $[f\, x]$ is also a fixed point of g. Since x is the **unique** fixed point of g, $f\, x = x$

Therefore, f has a fixed point.

In the automatic proof TPS formulates, proves, and applies the lemma that $f \circ g = g \circ f$.

Some References

Peter B. Andrews. Transforming Matings into Natural Deduction Proofs. In W. Bibel and R. Kowalski, editors, *Proceedings of the 5th International Conference on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 281–292, Les Arcs, France, 1980. Springer-Verlag.

Peter B. Andrews. Theorem Proving via General Matings. *Journal of the ACM*, 28:193–214, 1981.

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

Peter B. Andrews. *Classical Type Theory*, Chapter 15 of Handbook of Automated Reasoning, edited by Alan Robinson and Andrei Voronkov, Elsevier Science, Volume 2, 965–1007, 2001.

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

Peter B. Andrews and Matthew Bishop. On Sets, Types, Fixed Points, and Checkerboards. In Pierangelo Miglioli, Ugo Moscato, Daniele Mundici, and Mario Ornaghi, editors, *Theorem Proving with Analytic Tableaux and Related Methods. 5th International Workshop. (TABLEAUX '96)*, volume 1071 of *Lecture Notes in Artificial Intelligence*, pages 1–15, Terrasini, Italy, May 1996. Springer-Verlag.

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*, 16:321–353, 1996.

Matthew Bishop and Peter B. Andrews. Selectively Instantiating Definitions. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the 15th International Conference on Automated Deduction*, volume 1421 of *Lecture Notes in Artificial Intelligence*, pages 365–380, Lindau, Germany, 1998. Springer-Verlag.

Matthew Bishop. A Breadth-First Strategy for Mating Search. In Harald Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction*, volume 1632 of *Lecture Notes in Artificial Intelligence*, pages 359–373, Trento, Italy, 1999. Springer-Verlag.

Matthew Bishop. *Mating Search Without Path Enumeration*. PhD thesis, Department of Mathematical Sciences, Carnegie Mellon University, April 1999. Department of Mathematical Sciences Research Report No. 99–223. Available at http://gtps.math.cmu.edu/tps.html.

Sunil Issar. Path-Focused Duplication: A Search Procedure for General Matings. In *AAAI–90. Proceedings of the Eighth National Conference on Artificial Intelligence*, volume 1, pages 221–226. AAAI Press/The MIT Press, 1990.

Sunil Issar. *Operational Issues in Automated Theorem Proving Using Matings*. PhD thesis, Carnegie Mellon University, 1991. 147 pp.

Dale A. Miller. *Proofs in Higher-Order Logic*. PhD thesis, Carnegie Mellon University, 1983. 81 pp.

Dale A. Miller. A Compact Representation of Proofs. *Studia Logica*, 46(4):347–370, 1987.

Frank Pfenning. *Proof Transformations in Higher-Order Logic*. PhD thesis, Carnegie Mellon University, 1987. 156 pp.

Frank Pfenning and Dan Nesmith. Presenting Intuitive Deductions via Symmetric Simplification. In M. E. Stickel, editor, *Proceedings of the 10th International Conference on Automated Deduction*, volume 449 of *Lecture Notes in Artificial Intelligence*, pages 336–350, Kaiserslautern, Germany, 1990. Springer-Verlag.

Alonzo Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1940.

Gérard P. Huet. A Unification Algorithm for Typed λ-Calculus. *Theoretical Computer Science*, 1:27–57, 1975.