Contemporary Mathematics
Volume 29, 1984

AUTOMATING HIGHER-ORDER LOGIC

Peter B. Andrews[1], Dale A. Miller[2],
Eve Longini Cohen[3], Frank Pfenning[1]

Abstract

An automated theorem-proving system called TPS for proving theorems of first or higher-order logic in automatic, semi-automatic, or interactive mode has been developed. As its logical language TPS uses Church's formulation of type theory with λ-notation, in which most theorems of mathematics can be expressed very directly. As an interactive tool TPS has many features which facilitate writing formal proofs and manipulating and displaying formulas of logic in traditional notations. In automatic mode TPS combines theorem-proving methods for first-order logic with Huet's unification algorithm for typed λ-calculus, finds acceptable general matings (which represent the essential syntactic combinatorial information implicit in proofs), and constructs proofs in natural deduction style. Among the theorems which can be proved completely automatically is $\sim \exists G_{o\alpha}\ \forall F_{o\iota}\ \exists J_{\iota}\ [\,[G\ J] = F\,]$, which expresses Cantor's Theorem that a set has more subsets than members by asserting that there is no function G from individuals to sets which has every set F of individuals in its range. The computer substitutes for $F_{o\iota}$ the formula $[\lambda W_{\iota} \sim G_{o\alpha}\ W\ W]$, which denotes the set $\{W \mid W \notin G\ W\}$ and expresses the key idea in the classical diagonal argument.

The methods presently used by TPS in automatic mode are in principle complete for first-order logic, but not for higher-order logic. A recently proved extension of Herbrand's Theorem to type theory is presented. It is anticipated that this metatheorem will provide a basis for extending the capabilities of TPS.

---

[1]Mathematics Department, Carnegie-Mellon University, Pittsburgh, Pa. 15213

[2]Department of Computer and Information Sciences, Moore School/D2, University of Pennsylvania, Philadelphia, Pa. 19104

[3]The Aerospace Corporation, Information Sciences Research Office, M-105, PO Box 92957, Los Angeles, Ca. 90009

This work is supported by NSF grant MCS81-02870.

0271-4132/84 $1.00 + $.25 per page

§1 Introduction

We have barely begun to automate higher-order logic, but we have made a start, and this paper will give some indication of what we have done so far, and what we may do in the future. After setting the stage in this section, we shall give an example of an automatic proof in §2. We shall discuss a metatheorem which may provide a foundation for further progress in §3, and prove it in §4. The ideas in §§1-2 have been discussed previously in conferences on automated deduction, and more details can be found in [3], [4], and [10].

Theorems and proofs are expressed in some language. When one is automating the theorem-proving process, it's convenient to use a language of symbolic logic, because such a language represents a nice compromise between human languages and computer languages. It's comprehensible to people trained in logic, but has a precise syntax and clear representation of logical ideas which facilitates computer manipulations.

We wanted to automate the logic of a language which is generally adequate for expressing mathematical ideas in a direct and natural way, and chose to use an elegant formulation of type theory (otherwise known as higher-order logic) due to Alonzo Church [6]. Of course, some people would prefer to use axiomatic set theory as a language for formalizing mathematics, and thus stay within the framework of first-order logic. We won't take time to debate this issue now, but let's note that mathematicians do make intuitive distinctions between different types of mathematical objects, and it's useful to have these distinctions represented explicitly in the computer. Also, in Church's type theory one doesn't need axioms for set existence, since sets are represented explicitly by λ-expressions. Another advantage of using typed λ-calculus is the existence of powerful unification algorithms for expressions of this language which were developed by Gérard Huet [8] and by Pietrzykowski and Jensen [9]. We shall see how Huet's algorithm can be used in §2.

For the convenience of the reader we provide a brief introduction to the main features of the language in [6], which we shall call $\mathcal{T}$. $\mathcal{T}$ uses λ-notation for functions, which can easily be explained by an example. If F is a function and $F(x) = x^2 + x + 3$ for all x in the domain of F, then we write $[\lambda x . x^2 + x + 3]$ as a notation for the function F itself. We denote the result of applying this function to the argument 5 by $[[\lambda x . x^2 + x + 3]\ 5]$, which we convert to $[5^2 + 5 + 3]$ by a syntactic operation called *λ-contraction*.

Expressions of T have *types*, and we often indicate the types of variables and constants by attaching subscripts called *type symbols* to them. If F is a function which maps elements of type β to elements of type α, then F is said to have type $(\alpha\beta)$. Truth values and statements have type o. If $S_{o\gamma}$ (a function which maps elements of type γ to truth values) maps an element X_γ to truth, we write $[S_{o\gamma}X_\gamma]$ to indicate that $[S_{o\gamma}X_\gamma]$ is true, and say that X_γ is *in the set* $S_{o\gamma}$ or that X_γ *has the property* $S_{o\gamma}$. Thus sets and properties are identified with functions which map elements to truth values, and have types of the form $(o\gamma)$. It can be seen that if A_o is a statement, $[\lambda X_\gamma\ A_o]$ is another notation for the set $\{X_\gamma \mid A_o\}$ of all elements X_γ for which the statement A_o is true.

We use the convention of association to the left for omitting parentheses and brackets, so that $\alpha\beta\gamma$ is regarded as an abbreviation for $((\alpha\beta)\gamma)$. Also, a dot denotes a left bracket whose mate is as far to the right as possible without changing the pairing of brackets already present.

The proofs we construct are in *natural deduction* format, and here is an example of a very simple proof in this format:

(1)	1	⊢	$\forall x_\iota\ .[P_{o\iota}\ x\ ] \wedge .Q_{o\iota}\ x$	Hyp
(2)	1	⊢	$[P_{o\iota}\ H_\iota\ ] \wedge .Q_{o\iota}\ H$	∀I: H_ι 1
(3)	1	⊢	$P_{o\iota}\ H_\iota$	RuleP: 2
(4)	1	⊢	$\exists y_\iota\ .P_{o\iota}\ y$	∃G: H_ι 3
(5)		⊢	$[\forall x_\iota\ .[P_{o\iota}\ x\ ] \wedge .Q_{o\iota}\ x\ ] \supset .\exists y_\iota\ .P\ y$	Deduct: 4

The proof is a sequence of *lines*. Each line has the form

(n) $\mathcal{H}$ ⊢ A J

where n is a number serving as a *label* for the line, $\mathcal{H}$ is a (possibly empty) sequence containing the numbers of lines which are assumed as *hypotheses* for that line, A is the statement being *asserted* in that line, and J is the *justification* for that line. J indicates what rule of inference was applied to obtain the given line, and how it was used. The rules of inference for this system of natural deduction are listed in the Appendix.

At Carnegie-Mellon we have developed an automated theorem-proving system called TPS which can be used in both interactive and automatic modes, and various combinations of these, to construct such proofs. Considerable effort has been devoted to making TPS easy to interact with.

Certain computer terminals have been equipped with extra character generators so that a wide variety of symbols can be displayed on their screens. Thus, traditional notations of logic and mathematics appear on the terminal screens as well as on printed output. TPS can handle expressions of higher-order logic as well as first-order logic, but it is logically complete in automatic mode only for first-order logic.

TPS is both a system under development and a research tool. It enables us to test our ideas, and to look efficiently at examples which reveal new problems and suggest new ideas. While we would ultimately like to build into TPS a variety of facilities useful in automating logic, our present focus is on an approach to theorem-proving which involves analyzing the essential logical structure of a theorem [4], and using the information thus obtained to construct a proof without further search [3].

§2 An Example of an Automatic Proof

In order to convey a general idea of what TPS is like, we shall show how it works on an example. We shall present the actual output produced by the computer as it proves the theorem, with certain material edited out, and with explanatory comments added in italics. (Since the program is still under development, the reader may notice that it could be made more elegant in certain respects.) Lines preceded by a * below are typed in by the human operator, and all other lines (except explanations) are produced by TPS.

TPS has a list of theorems with attached comments stored in its memory, and we start by asking it to state THM5:

```
*STATE THM5
~ .∃G_{oα} ∀F_{oι} ∃J_{ι} .[G J ] = F
(THIS IS CANTOR'S THEOREM FOR SETS)
(THE POWER SET OF A SET HAS MORE MEMBERS THAN THE SET)
```

THM5 is a rather special statement of Cantor's Theorem in which the types attached to the variables play a very significant role. Let S be the set which we wish to show is smaller than its power set, and let an object have type ι iff it is a member of S. Thus the objects of type (oι) are simply the subsets of S, and the objects of type ((oι)ι) are the functions which map members of S to subsets of S. Thus THM5 simply says that there is no function G which maps

members of S to subsets of S and which has every subset F of S in its range.

Statements of the form [X = Y] *are actually abbreviations for* $[=_{o\alpha\alpha}\ X_\alpha\ Y_\alpha]$, *where* $=_{o\alpha\alpha}$ *is defined as* $\lambda X_\alpha\ \lambda Y_\alpha\ \forall Q_{o\alpha}\ .[Q\ X] \supset .Q\ Y$. *(Thus, X and Y are equal iff every property Q of X is also a property of Y.) If we instantiate the definition of = in THM5, here is what we get:*

$$\sim\ .\exists G_{o\iota\iota}\ \forall F_{o\iota}\ \exists J_\iota\ \forall Q_{o(o\iota)}\ .[Q\ .G\ J\ ] \supset .Q\ F$$

We initiate the process of having TPS prove THM5 with the PLAN command:

```
*PLAN THM5

(RunTime 0.10091015 seconds  ConsCount 9)

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
(100)

(100)        ⊢  ~ .∃G_oιι ∀F_oι ∃J_ι .[G J ] = F                PLAN1

------------------------------------------------------------------------
```

TPS displays the proof it is constructing in a portion of the terminal screen called the proof window, and the inital contents of the proof window are shown above. The proof will be constructed from the bottom up as well as from the top down, and TPS starts off by making the theorem to be proved the last line of the proof, and giving this line the number 100. Of course, this line of the proof is really unjustified at this stage, and PLAN1 is just an indication that TPS plans to prove line 100. At the top of the proof window is the status line, which is a list of the numbers of lines which are still active (in need of further processing).

We could now say GO and TPS would find a proof completely automatically. However, we wish to force TPS to construct an indirect proof, since this will be a little easier for us to grasp intuitively. (A heuristic which would enable TPS to discover that an indirect proof is most appropriate for this theorem is discussed on page 284 of [3], but it is not yet implemented in TPS.) Therefore we instruct TPS to set up an indirect proof:

```
*P-INDIRECT
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
(99 1)
```

```
(1)   1     ⊢  ~ .~ .∃G_ου ∀F_οι ∃J_ι .[G J ] = F          Hyp
(99)   1    ⊢  ⊥                                          PLAN2
(100)       ⊢  ~ .∃G_ου ∀F_οι ∃J_ι .[G J ] = F    Indirect: 99
----------------------------------------------------------------
```

TPS has expanded the proof. Line 100 now has a justification and is no longer active, and TPS plans to prove line 99, which asserts that a contradiction (symbolized by ⊥) follows from line 1, which is the negation of THM5.

From here on the proof will proceed automatically. The first few steps involve applications of rules of inference whose appropriateness is rather obvious.

```
*GO

Evaluating  D-NEG 1

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
(99 2)

(1)   1     ⊢  ~ .~ .∃G_ου ∀F_οι ∃J_ι .[G J ] = F          Hyp
(2)   1     ⊢  ∃G_ου ∀F_οι ∃J_ι .[G J ] = F            RuleP: 1
(99)   1    ⊢  ⊥                                          PLAN2
(100)       ⊢  ~ .∃G_ου ∀F_οι ∃J_ι .[G J ] = F    Indirect: 99
----------------------------------------------------------------
Evaluating  P-CHOOSE 99 2
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
(98 3)

(1)   1     ⊢  ~ .~ .∃G_ου ∀F_οι ∃J_ι .[G J ] = F          Hyp
(2)   1     ⊢  ∃G_ου ∀F_οι ∃J_ι .[G J ] = F            RuleP: 1
(3)   3     ⊢  ∀F_οι ∃J_ι .[G_ου J ] = F        Choose: G_ου  2
(98)   1,3  ⊢  ⊥                                          PLAN2
(99)   1    ⊢  ⊥                                       RuleC: 2 98
(100)       ⊢  ~ .∃G_ου ∀F_οι ∃J_ι .[G J ] = F    Indirect: 99
----------------------------------------------------------------
The problem, (98 3), can not be reduced.
```

TPS has discovered that it can progress no further by trivial applications of rules of logic, and prepares to proceed with the MatingSearch program. It notes that the essential problem is to derive a contradiction from line 3, and (rather inelegantly) adds line 97 to the proof (as shown below) to facilitate its internal processing.

```
Evaluating  FINDPLAN 98 (3)

(RunTime 3.7734394  ConsCount 22311)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
(97 3)

(1)   1     ⊢  ~ .~ .∃G_ou ∀F_oι ∃J_ι .[G J ] = F          Hyp
(2)   1     ⊢  ∃G_ou ∀F_oι ∃J_ι .[G J ] = F             RuleP: 1
(3)   3     ⊢  ∀F_oι ∃J_ι .[G_ou J ] = F          Choose: G_ou 2
(97)  3     ⊢  ⊥                                          PLAN3
(98)  1,3   ⊢  ⊥                                 Deduct,RuleP: 97
(99)  1     ⊢  ⊥                                      RuleC: 2 98
(100)       ⊢  ~ .∃G_ou ∀F_oι ∃J_ι .[G J ] = F     Indirect: 99
----------------------------------------------------------------------
```

Next TPS processes line 3 by instantiating the definition of =, writing the ⊃ thus introduced in terms of ~ and ∨, and skolemizing to eliminate the existential quantifier. The variable J_ι is replaced by the term $[J|A_{\iota(o\iota)}\ F_{o\iota}]$, where $J|A_{\iota(o\iota)}$ is a skolem function. TPS displays the resulting formula, and the names it has attached to its literals, as follows:

```
[1]  ∀F_oι ∀Q_o(oι) .[~ .Q .G_ou .J|A_ι(oι) F ] ∨ .Q F
[1]  ∀F_oι ∀Q_o(oι) .LIT2 ∨ LIT3
```

From [1] TPS must derive a contradiction by instantiating the quantifiers appropriately.

```
Path with no potential mates: (LIT2)
No proof on this level
(RunTime 6.5491445  ConsCount 31522)
```

TPS applied its search process, which will be described below, and quickly discovered that no single instantiation of the two quantifiers can produce a contradiction.

```
Replicate outermost quantifiers.
```

$$[2] \quad [\forall F^1_{o\iota}\ \forall Q^1_{o(o\iota)}\ .[\sim\ .Q^1\ .G_{o\iota}\ .J|A_{\iota(o\iota)}\ F^1\ ] \vee\ .Q^1\ F^1\ ]$$
$$\wedge\ .\forall F^2_{o\iota}\ \forall Q^2_{o(o\iota)}\ .[\sim\ .Q^2\ .G\ .J|A\ F^2\ ] \vee\ .Q^2\ F^2$$

$$[2] \quad [\forall F^1_{o\iota}\ \forall Q^1_{o(o\iota)}\ .LIT2^1 \vee LIT3^1\ ]$$
$$\wedge\ .\forall F^2_{o\iota}\ \forall Q^2_{o(o\iota)}\ .LIT2^2 \vee LIT3^2$$

The matingsearch process is described in [4], but we shall briefly summarize the basic concepts which underlie it. A mating $\mathcal{M}$ for a wff W (such as the matrix of [2]) is a relation between occurrences of literals of W such that there is a substitution θ which makes literals with mated occurrences complementary. Thus, if A $\mathcal{M}$ B, then $\theta B = \sim \theta A$. θ is called the unifying substitution associated with $\mathcal{M}$. A vertical path through W is a sequence of occurrences of literals whose conjunction is one of the conjuncts in the disjunctive normal form of W. The vertical paths through [2] are ($LIT2^1$ $LIT2^2$), ($LIT2^1$ $LIT3^2$), ($LIT3^1$ $LIT2^2$), and ($LIT3^1$ $LIT3^2$). A mating for W is acceptable iff every vertical path contains a mated pair of literals. Thus, W becomes a contradiction when a substitution associated with an acceptable mating is applied to it.

TPS searches for an acceptable mating and its associated substitution in a systematic way. It tries various possibilities, and backtracks when it finds that the subtitutions required to mate various literal-pairs are incompatible. The processes of finding appropriate pairs of literals to mate, and searching for unifying substitutions, are carried on more or less simultaneously, and interact with each other. The search processes are controlled by various heuristics.

```
Path with no mates: (LIT2^1 LIT2^2)
   try this arc: (LIT2^1 LIT2^2)

Partial Mating 0:
```

```
   (LIT2¹)-(LIT2²)

Path with no mates: (LIT3¹ LIT2²)
   try this arc: (LIT3¹ LIT2²)

Partial Mating 0:
   (LIT2¹ LIT3¹)-(LIT2²)

Path with no mates: (LIT3¹ LIT3²)
   try this arc: (LIT3¹ LIT3²)

Partial Mating 0:
   (LIT2¹ LIT3¹)-(LIT2² LIT3²)

Mating 0 is complete.
(RunTime 9.7913925  ConsCount 44797)
```

Mating 0 is the mating which contains the three literal-pairs referred to as arcs above, plus the induced arc $(LIT2^1\ LIT3^2)$. *TPS actually found this mating without any need for backtracking, though backtracking usually does occur when dealing with formulas with a more complicated propositional structure. In essence, TPS has found that a contradiction will be obtained if the quantifiers in [2] are instantiated in such a way that [2] reduces to a formula having the form*

$$[\,D \vee D\,] \wedge [\sim D \vee \sim D].$$

Thus far TPS has actually done only a partial and superficial check that a substituion associated with Mating 0 exists. It next searches in earnest for such a substitution, using Gérard Huet's elegant unification algorithm [8]. TPS owes much of its power to this algorithm, but we omit the lengthy details of the unification process. Actually, TPS could be working on an incorrect mating, for which the unification algorithm would never terminate, so TPS alternates between generating new matings and continuing work on all of their associated unification problems. Eventually it reaches a Terminal Success Node in its search for a unifier associated with Mating 0:

```
N0-2-0-0-0-0-0-0:=    TSN
F¹oι  ⇐λW1ι .~ .Gοι W1 W1
F²oι  ⇐λW1ι .~ .Gοι W1 W1
```

$Q^1_{o(oι)}$ ⇐λW1$_{oι}$ .W1 .J|A$_{ι(oι)}$.λW2$_ι$.∼ .G$_{oιι}$ W2 W2

$Q^2_{o(oι)}$ ⇐λW1$_{oι}$.∼ .W1 .J|A$_{ι(oι)}$.λW2$_ι$.∼ .G$_{oιι}$ W2 W2

```
Unification complete
(RunTime 25.365179  ConsCount 74167)
```

Having finished its search, TPS prepares to use the information it has found to complete the natural deduction proof. First it notices that the substitutions for F^1 and F^2 are the same, which means that it would have been more appropriate to duplicate the quantifier ∀Q of formula [1] instead of ∀F. TPS makes the appropriate changes, replacing formula [2] by formula [3] below:

```
The replication has been simplified.
Old:  ( F_oι   2 )    New:  ( Q_o(oι)   2 )
```

[3]

∀F$_{oι}$. [∀Q$^1_{o(oι)}$.[∼ .Q^1 .G$_{oιι}$ .J|A$_{ι(oι)}$ F] ∨ .Q^1 F]

∧ .∀Q$^2_{o(oι)}$.[∼ .Q^2 .G .J|A F] ∨ .Q^2 F

```
The substitution has also been changed.
New:
```

F$_{oι}$ ⇐λW1$_ι$.∼ .G$_{oιι}$ W1 W1

$Q^1_{o(oι)}$ ⇐λW1$_{oι}$ .W1 .J|A$_{ι(oι)}$.λW2$_ι$.∼ .G$_{oιι}$ W2 W2

$Q^2_{o(oι)}$ ⇐λW1$_{oι}$.∼ .W1 .J|A$_{ι(oι)}$.λW2$_ι$.∼ .G$_{oιι}$ W2 W2

TPS displays the result of instantiating the quantifiers in [3] with the new substitution and λ-reducing as formula [4]:

[4]

```
   [   [∼ .G_oιι [J|A_ι(oι) .λW1_ι .∼ .G W1 W1 ]
                          .J|A .λW2_ι .∼ .G W2 W2 ]
     ∨ .∼ .G [J|A .λW2 .∼ .G W2 W2 ]
                          .J|A .λW2 .∼ .G W2 W2 ]
 ∧ .   [∼ .∼ .G [J|A .λW1 .∼ .G W1 W1 ]
                          .J|A .λW2 .∼ .G W2 W2 ]
     ∨ .∼ .∼ .G [J|A .λW2 .∼ .G W2 W2 ]
                          .J|A .λW2 .∼ .G W2 W2
```

Note that [4] is indeed a contradiction having the form anticipated above.

TPS now eliminates the skolem terms from the substitution terms and transforms the information it has obtained into a plan for proving line 97, which it calls PLAN3. Thus, PLAN3 is no longer an empty label indicating that TPS plans to prove line 97; it is now the name of a data structure which contains all the information necessary to carry out the proof. PLAN3 is exhibited by displaying the expanded form of the formula to be proved, and the substitution which (as will be seen in §3) reduces it to a tautology.

PLAN3 is:

$$\sim .\forall F_{o\iota}\ \exists J_{\iota}\ .\quad [\forall Q^1_{o(o\iota)}\ .[Q^1\ .G_{o\iota\iota}\ J\ ]_{ATM2^1} \supset .Q^1\ F\ ]_{ATM3^1}$$
$$\wedge\ .\forall Q^2_{o(o\iota)}\ .[Q^2\ .G\ J\ ]_{ATM2^2} \supset .Q^2\ F_{ATM3^2}$$

The substitution is:

$$F_{o\iota} \Leftarrow \lambda W1_{\iota}\ .\sim .G_{o\iota\iota}\ W1\ W1 \qquad Q^1_{o(o\iota)} \Leftarrow \lambda W1_{o\iota}\ .W1\ J_{\iota}$$

$$Q^2_{o(o\iota)} \Leftarrow \lambda W1_{o\iota}\ .\sim .W1\ J_{\iota}$$

Now TPS continues the construction of the natural deduction proof. The substitution terms in PLAN3 are used to instantiate the quantifiers in the proof below.

Evaluating D-ALL 3 $\lambda W1_{\iota}\ .\sim .G_{o\iota\iota}\ W1\ W1$ 97

++

(97 4)

(1)	1	⊢	$\sim .\sim .\exists G_{o\iota\iota}\ \forall F_{o\iota}\ \exists J_{\iota}\ .[G\ J\ ] = F$	Hyp
(2)	1	⊢	$\exists G_{o\iota\iota}\ \forall F_{o\iota}\ \exists J_{\iota}\ .[G\ J\ ] = F$	RuleP: 1
(3)	3	⊢	$\forall F_{o\iota}\ \exists J_{\iota}\ .[G_{o\iota\iota}\ J\ ] = F$	Choose: $G_{o\iota\iota}$ 2
(4)	3	⊢	$\exists J_{\iota}\ .[G_{o\iota\iota}\ J\ ] = .\lambda W1_{\iota}\ .\sim .G\ W1\ W1$ ∀I: $[\lambda W1_{\iota}\ .\sim .G_{o\iota\iota}\ W1\ W1\ ]\ 3$	
(97)	3	⊢	⊥	PLAN3
(98)	1,3	⊢	⊥	Deduct,RuleP: 97
(99)	1	⊢	⊥	RuleC: 2 98
(100)		⊢	$\sim .\exists G_{o\iota\iota}\ \forall F_{o\iota}\ \exists J_{\iota}\ .[G\ J\ ] = F$	Indirect: 99

```
Evaluating  P-CHOOSE 97 4

Evaluating  D-DEF1 5

Evaluating  D-ALL 6 λW1_oι .W1 J_ι  96

Evaluating  D-ALL 6 λW1_oι .~ .W1 J_ι  96
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
(96 8 7)

(1)   1     ⊢  ~ .~ .∃G_oιι ∀F_oι ∃J_ι .[G J ] = F          Hyp
(2)   1     ⊢  ∃G_oιι ∀F_oι ∃J_ι .[G J ] = F            RuleP: 1
(3)   3     ⊢  ∀F_oι ∃J_ι .[G_oιι J ] = F          Choose: G_oιι  2
(4)   3     ⊢  ∃J_ι .[G_oιι J ] = .λW1_ι .~ .G W1 W1
                                   ∀I: [λW1_ι .~ .G_oιι W1 W1 ] 3
(5)   5     ⊢  [G_oιι J_ι ] = .λW1_ι .~ .G W1 W1
                                              Choose: J_ι  4
(6)   5     ⊢     ∀Q_o(oι) .[Q .G_oιι J_ι ]
                ⊃ .Q .λW1_ι .~ .G W1 W1                Def: 5
(7)   5     ⊢  [G_oιι J_ι J ] ⊃ .~ .G J J
                                        ∀I: [λW1_oι .W1 J_ι ] 6
(8)   6     ⊢  [~ .G_oιι J_ι J ] ⊃ .~ .~ .G J J
                                    ∀I: [λW1_oι .~ .W1 J_ι ] 6
(96)  3,5   ⊢  ⊥                                       PLAN3
(97)  3     ⊢  ⊥                                  RuleC: 4 96
(98)  1,3   ⊢  ⊥                            Deduct,RuleP: 97
(99)  1     ⊢  ⊥                                  RuleC: 2 98
(100)       ⊢  ~ .∃G_oιι ∀F_oι ∃J_ι .[G J ] = F    Indirect: 99
----------------------------------------------------------------
```

After each application of a rule of inference, TPS checks to see whether the line it is planning to prove follows by RuleP from the other active lines. TPS now notices that line 96 follows from lines 7 and 8 by this rule.

```
Evaluating  RULEP 96 (8 7)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
(1)   1     ⊢  ∼ .∼ .∃G_οιι ∀F_οι ∃J_ι .[G J ] = F                  Hyp
(2)   1     ⊢  ∃G_οιι ∀F_οι ∃J_ι .[G J ] = F                    RuleP: 1
(3)   3     ⊢  ∀F_οι ∃J_ι .[G_οιι J ] = F                Choose: G_οιι  2
(4)   3     ⊢  ∃J_ι .[G_οιι J ] = .λW1_ι .∼ .G W1 W1
                                          ∀I: [λW1_ι .∼ .G_οιι W1 W1 ] 3
(5)   5     ⊢  [G_οιι J_ι ] = .λW1_ι .∼ .G W1 W1
                                                  Choose: J_ι  4
(6)   5     ⊢     ∀Q_ο(οι) .[Q .G_οιι J_ι ]
                  ⊃ .Q .λW1_ι .∼ .G W1 W1                      Def: 5
(7)   5     ⊢  [G_οιι J_ι J ] ⊃ .∼ .G J J
                                       ∀I: [λW1_οι .W1 J_ι ] 6
(8)   5     ⊢  [∼ .G_οιι J_ι J ] ⊃ .∼ .∼ .G J J
                                    ∀I: [λW1_οι .∼ .W1 J_ι ] 6
(96)  3,5   ⊢  ⊥                                          RuleP: 7 8
(97)  3     ⊢  ⊥                                        RuleC: 4 96
(98)  1,3   ⊢  ⊥                                   Deduct,RuleP: 97
(99)  1     ⊢  ⊥                                        RuleC: 2 98
(100)       ⊢  ∼ .∃G_οιι ∀F_οι ∃J_ι .[G J ] = F     Indirect: 99
----------------------------------------------------------------------
There are no plan lines.

(RunTime 31.780359  ConsCount 103666)
```

Since every line of the proof now has a justification, the proof is complete. Note that it is simply a formalized presentation of the traditional diagonal proof of Cantor's Theorem. It was obtained in less than 32 seconds by a purely syntactic analysis of the theorem.

§3 A Metatheorem for Extending the Capabilities of TPS

TPS is able to prove certain theorems of higher-order logic, such as Cantor's Theorem, by combining higher-order unification with a theorem-proving method which was really designed for first-order logic. We're a long way from having a reasonable theorem-proving procedure for higher-order logic which is logically complete even in principle, but in this section we shall present a metatheorem which provides a conceptual framework in which it may be possible to develop such a procedure.

It may be recalled that in his thesis Herbrand discussed properties A, B, and C of wffs, and asserted for each of the properties that a wff of first-order logic is provable iff it has the property. In its present formulation, the metatheorem below is a generalization to higher-order logic of Herbrand's Theorem using Property A, in which skolem functions are not used. (The metatheorem as formulated below was proved by Andrews. Another formulation due to Miller will be presented in [11].)

In order to state the metatheorem precisely enough so that it can be proved in §4, we must next give some technical definitions. However, the reader who wishes to temporarily skip these can get a general idea of what the metatheorem says by looking at the example later in this section.

In §1 we used $\mathcal{T}$ as a name for a language of type theory, but we shall now use $\mathcal{T}$ in a more precise way as the name of the logical system described in §2 of [1]; $\mathcal{T}$ consists of the system originally presented by Church in [6], minus axioms of extensionality, descriptions, choice, and infinity. Thus $\mathcal{T}$ simply embodies the logic of propositional connectives, quantifiers, and λ-conversion in the context of type theory, and (as in [2]) we shall refer to $\mathcal{T}$ as *elementary type theory*.

The primitive logical constants of $\mathcal{T}$ are $\sim_{oo}$ (negation), $\vee_{ooo}$ (disjunction), and $\Pi_{o(o\alpha)}$. $\Pi_{o(o\alpha)}[\lambda x_\alpha C_o]$ may be abbreviated as $\forall x_\alpha C_o$, and $\Pi_{o(o\alpha)}\, A_{o\alpha}$ is provably equivalent to $\forall x_\alpha[A_{o\alpha} x_\alpha]$ if x_α is not free in $A_{o\alpha}$. $\wedge$, $\supset$, $\equiv$, and $\exists$ are defined in familiar ways.

The *well-formed formulas (wffs)* of $\mathcal{T}$ are defined in [6] and in [1]. We shall use *occwfp* as an abbreviation for the phrase *occurrence of a well-formed part*, and *occwfps* for *occurrences of well-formed parts*. We say that an occwfp A of a wff_o (wff of type o) W_o is *very accessible* in W_o iff A is only in the scope of propositional connectives in W_o; for greater precision this notion is defined inductively as follows:

1. W_o is very accessible in W_o.
2. If $\sim B$ is very accessible in W_o, so is B.
3. If $[B \vee C]$ is very accessible in W_o, so are B and C.

An occwfp A of W_o is *negative* in W_o iff A is in the scope of an odd number of occurrences of negation signs in W_o, and is *positive* in W_o otherwise. An occwfp of W_o having the form $\Pi_{o(o\alpha)}A_{o\alpha}$ is *essentially universal* iff it is positive in W_o, and *essentially existential* iff it is negative in W_o. (The essentially universal [existential] occwfps of W_o correspond to the universal [existential] quantifiers in prenex normal forms of W_o.) An occwfp A of a wff W_o is *critical in* W_o iff A is very accessible and essentially existential in W_o.

Let W be a wff_o which has a very accessible occwfp M having the form $\Pi_{o(o\alpha)}A_{o\alpha}$. Let U be the result of replacing the given occurrence of M in W by an occurrence of a wff N which we discuss below.

1. If M is positive in W and N is $A_{o\alpha}x_\alpha$, where x_α does not occur free in W, we say that U is obtained from W by *universal quantifier deletion* (∀-deletion).
2. If M is negative in W and N is $[\Pi_{o(o\alpha)}A_{o\alpha} \wedge \Pi_{o(o\alpha)}A_{o\alpha}]$, we say that U is obtained from W by *existential duplication* (∃-*duplication*).
3. If M is negative in W and N is $A_{o\alpha}B_\alpha$, where B_α is any wff_α, we say that U is obtained from W by *existential instantiation* (∃-*instantiation*).

We shall say that we apply λ-*reduction* to a wff when we apply λ-contraction (rule 2.6.2 of [1]) to one of its parts of the form $[[\lambda x_\alpha B_\beta]A_\alpha]$, after making any necessary alphabetic changes of bound variables (rule 2.6.1 of [1]). A wff is in λ-*normal form* if it has no parts of the form $[[\lambda x_\alpha B_\beta]A_\alpha]$.

We call ∀-deletion, ∃-duplication, ∃-instantiation, and λ-reduction the four *basic operations*. A *development* of a sentence W of $\mathcal{T}$ is any wff_o obtained from it by any sequence of applications of the basic operations.

It can be seen that if W_o contains a very accessible positive occwfp $\forall x_\alpha C_o$, where x_α does not occur free in W_o, then this occurrence of $\forall x_\alpha C_o$ can be replaced by C_o by ∀-deletion and λ-reduction. Indeed, $\forall x_\alpha C_o$ is an abbreviation for $\Pi_{o(o\alpha)}[\lambda x_\alpha C_o]$, which can be replaced by $[\lambda x_\alpha C_o]x_\alpha$ by ∀-deletion, and λ-reduced to C_o. Similarly, a very accessible positive occwfp

$\exists x_\alpha C_o$ is an abbreviation for $\sim \Pi_{o(o\alpha)}[\lambda x_\alpha \sim C_o]$, which can be $\exists$-instantiated to $\sim [\lambda x_\alpha \sim C_o]B_\alpha$, which λ-reduces (modulo a double negation which we often omit writing) to the result of instantiating the quantifier in $\exists x_\alpha C_o$ with the wff B_α. Thus, when we abbreviate wffs so that quantifiers always occur as $\forall$ or $\exists$, and $\Pi_{o(o\alpha)}$ does not explicitly appear, we see that developments of wffs are obtained by deleting essentially universal quantifiers, duplicating or instantiating essentially existential quantifiers, and performing λ-reductions.

We next illustrate these concepts by developing THM5. In abbreviated form, THM5 is:

$$\sim .\exists G_{o\iota} \forall F_{o\iota} \exists J_\iota .[G\ J] = F$$

We start by instantiating the definition of $=$ (which is really just an application of λ-reduction), and obtain:

$$\sim .\exists G_{o\iota} \forall F_{o\iota} \exists J_\iota \forall Q_{o(o\iota)} .[Q\ .G\ J] \supset .Q\ F$$

Delete $\exists G$:

$$\sim .\forall F_{o\iota} \exists J_\iota \forall Q_{o(o\iota)} .[Q\ .G_{o\iota}\ J] \supset .Q\ F$$

Instantiate $\forall F$ with $\lambda W_\iota .\sim .G_{o\iota}\ W\ W$:

$$\sim .\exists J_\iota \forall Q_{o(o\iota)} .[Q\ .G_{o\iota}\ J] \supset .Q\ .\lambda W_\iota .\sim .G\ W\ W$$

Delete $\exists J$:

$$\sim .\forall Q_{o(o\iota)} .[Q\ .G_{o\iota}\ J_\iota] \supset .Q\ .\lambda W_\iota .\sim .G\ W\ W$$

Duplicate $\forall Q$:

$$\sim .\ [\forall Q_{o(o\iota)} .[Q\ .G_{o\iota}\ J_\iota] \supset .Q\ .\lambda W_\iota .\sim .G\ W\ W]$$
$$\wedge .\forall Q\ .[Q\ .G\ J] \supset .Q\ .\lambda W\ .\sim .G\ W\ W$$

Instantiate the first $\forall Q$ with $\lambda U_{o\iota} .U\ J_\iota$:

$$\sim .\ [[[\lambda U_{o\iota} .U\ J_\iota].G_{o\iota}\ J] \supset .[\lambda U\ .U\ J].\lambda W_\iota .\sim .G\ W\ W]$$
$$\wedge .\forall Q_{o(o\iota)} .[Q\ .G\ J] \supset .Q\ .\lambda W\ .\sim .G\ W\ W$$

λ-reduce:

$$\sim . \quad [[G_{o\iota\iota}\ J_\iota\ J\ ] \supset .[\lambda U_{o\iota}\ .U\ J\ ].\lambda W_\iota\ .\sim\ .G\ W\ W\ ]$$
$$\wedge\ .\forall Q_{o(o\iota)}\ .[Q\ .G\ J\ ] \supset .Q\ .\lambda W\ .\sim\ .G\ W\ W$$

$$\sim . \quad [[G_{o\iota\iota}\ J_\iota\ J\ ] \supset .[\lambda W_\iota\ .\sim\ .G\ W\ W\ ]\ J\ ]$$
$$\wedge\ .\forall Q_{o(o\iota)}\ .[Q\ .G\ J\ ] \supset .Q\ .\lambda W\ .\sim\ .G\ W\ W$$

$$\sim . \quad [[G_{o\iota\iota}\ J_\iota\ J\ ] \supset .\sim\ .G\ J\ J\ ]$$
$$\wedge\ .\forall Q_{o(o\iota)}\ .[Q\ .G\ J\ ] \supset .Q\ .\lambda W_\iota\ .\sim\ .G\ W\ W$$

Instantiate the remaining ∀Q with $\lambda U_{o\iota}\ .\sim\ .U\ J_\iota$:

$$\sim . \quad [[G_{o\iota\iota}\ J_\iota\ J\ ] \supset .\sim\ .G\ J\ J\ ]$$
$$\wedge . \quad [[\lambda U_{o\iota}\ .\sim\ .U\ J\ ].G\ J\ ]$$
$$\supset .[\lambda U\ .\sim\ .U\ J\ ].\lambda W_\iota\ .\sim\ .G\ W\ W$$

λ-reduce:

$$\sim . \quad [[G_{o\iota\iota}\ J_\iota\ J\ ] \supset .\sim\ .G\ J\ J\ ]$$
$$\wedge\ .[\sim\ .G\ J\ J\ ] \supset .[\lambda U_{o\iota}\ .\sim\ .U\ J\ ].\lambda W_\iota\ .\sim\ .G\ W\ W$$

$$\sim . \quad [[G_{o\iota\iota}\ J_\iota\ J\ ] \supset .\sim\ .G\ J\ J\ ]$$
$$\wedge\ .[\sim\ .G\ J\ J\ ] \supset .\sim\ .[\lambda W_\iota\ .\sim\ .G\ W\ W\ ]\ J$$

$$\sim . \quad [[G_{o\iota\iota}\ J_\iota\ J\ ] \supset .\sim\ .G\ J\ J\ ]$$
$$\wedge\ .[\sim\ .G\ J\ J\ ] \supset .\sim\ .\sim\ .G\ J\ J$$

Note that we have obtained a development of THM5 which is tautologous (a substitution instance of a tautology of propositional calculus). Our main metatheorem asserts that this can be done for every theorem of T.

Metatheorem. A sentence of T is provable in T iff it has a tautologous development.

It can be seen from the results in [11] that once one knows how to obtain a tautologous development of a sentence, one can construct a natural-

deduction proof of it without further search. Thus, the metatheorem above shows that we can focus our research in higher-order theorem proving on the problem of finding tautologous developments. Of course, this is a partial extension to higher-order logic of the basic approach in [4].

Although a tautologous development of THM5 can be found by purely automatic methods, significant new ideas must be found in order to expand the set of sentences for which tautologous developments can be found automatically. Nevertheless, it seems reasonable to hope that the search for tautologous developments provides a context in which further progress on automating higher-order logic can be made.

§4 Proof of the Metatheorem

In this section we prove the metatheorem stated in §3. We shall write $\vdash W$ to indicate that W is a theorem of $\mathcal{T}$, and $\models W$ to indicate that W is tautologous.

By examining the axioms and rules of inference of $\mathcal{T}$, one easily sees that analogues of all theorems and derived rules of inference of first-order logic are provable in $\mathcal{T}$, and we shall use this fact freely. In particular, if $\models W$, then $\vdash W$. Also, it is easy to establish the following:

Lemma (Substitutivity of Implication). Let M and N be $wffs_o$ such that $\vdash M \supset N$, let W and U be $wffs_o$ such that M has an occurrence in W as a very accessible occwfp, and U is obtained from W by replacing the designated occurrence of M in W by an occurrence of N. Then $\vdash W \supset U$ if the designated occurrence of M is positive in W, and $\vdash U \supset W$ if it is negative.

It is easy to see that if a sentence W has a tautologous development, then $\vdash W$. Indeed, suppose $W^0, W^1, \ldots, W^n$ is a sequence of $wffs_o$ such that W^0 is W, W^n is tautologous, and W^{i+1} is obtained from W^i by a basic operation for each $i < n$. Since $\vdash W^n$, it suffices to show for each $i < n$ that if $\vdash W^{i+1}$, then $\vdash W^i$. Since $\vdash \Pi_{o(o\alpha)}A_{o\alpha} \supset [\Pi_{o(o\alpha)}A_{o\alpha} \wedge \Pi_{o(o\alpha)}A_{o\alpha}]$ and $\vdash \Pi_{o(o\alpha)}A_{o\alpha} \supset A_{o\alpha}B_{\alpha}$ and operations of λ-conversion are reversible, it is easy to see with the aid of the Lemma that if W^{i+1} is obtained from W^i by $\exists$-duplication, $\exists$-instantiation, or λ-reduction, then $\vdash W^{i+1} \supset W^i$, which is more than sufficient. Also, if W^{i+1} is obtained from W^i by $\forall$-deletion, then W^i can be inferred from W^{i+1} by universal generalization and anti-prenex rules for pushing in quantifiers. (Here we use the condition on the variable in the

∀-deletion rule). This completes the proof of the theorem in the trivial direction, and shows that finding tautologous developments is a sound method for establishing theorems of $\mathcal{T}$.

Next we must show that the method is complete. Here we shall assume familiarity with §4 of [1] (the cut-elimination theorem for $\mathcal{T}$, which is based on the work of Takahashi [16]). It is interesting to note that just as there is a close relation between Gentzen's cut-elimination theorem and Herbrand's theorem for first-order logic, there is a close relation between cut-elimination for $\mathcal{T}$ and our generalization of Herbrand's theorem to $\mathcal{T}$.

We start by introducing some definitions. A wff_o is *open* iff it is in λ-normal form and contains no positive very accessible occwfps of the form $\Pi_{o(o\alpha)}A_{o\alpha}$. We *open* a wff_o W by applying the operations of ∀-deletion and λ-reduction to it repeatedly until it is open. This process is called *opening* W, and the resulting wff is called an *open form* of W. It can be seen (as a minor extension of the Church-Rosser Theorem for typed λ-calculus) that every wff_o W has an open form (i.e., the process terminates), which is unique modulo alphabetic changes of free and bound variables.

If W is a wff_o and U is obtained from W by applying ∃-instantiation (once) to W and opening the resulting wff, then we say that U is obtained from W by *major ∃-instantiation*. The sequence $W^0,\ldots, W^i,\ldots, W^n$ (where $n \geq 0$) is a *development sequence (d-seq)* for a wff_o W iff W^0 is an open form of W, and for each $i < n$, W^{i+1} is obtained from W^i by ∃-duplication or by major ∃-instantiation. Clearly, each of the wffs W^i in a d-seq for W is an open development of W. If $W^0,\ldots, W^n$ is a d-seq for W and $\models W^n$, we say that the d-seq $W^0,\ldots, W^n$ *verifies* W and is a *verification of* W, and write $\square W\{W^0,\ldots,W^n\}$. If some d-seq verifies W we say that W is *verified* and write $\square W$.

Let $W^0,\ldots,W^i,\ldots, W^n$ be any development sequence, and let R be any very accessible occwfp of W^i (where $0 \leq i \leq n$). For each $k \geq i$, we define the *descendant* R^k of R in W^k by induction on k:

1. If $k = i$, then R^k is R.
2. Suppose $k > i$, so W^{k+1} is obtained from W^k by replacing some critical occwfp M of W^k by an occwfp N. M has the form $\Pi_{o(o\alpha)}A_{o\alpha}$, so M is either a subformula of R^k or does not overlap R^k. In the former case R^{k+1} is the subformula of W^{k+1} obtained

when M is replaced by N in R^k. In the latter case R^{k+1} is the ocurrence of R^k in W^{k+1} which corresponds to the occurrence of R^k in W^k.

Now we are ready to start the completeness proof. We shall show that if $\vdash W$, then $\square W$. Suppose that $\vdash W$; then by Theorem 4.10 of [1], W has a proof $P^1,...,P^k,...,P^m$ in the cut-free system $\mathcal{G}$ of [1]. We show by induction on k that $\square P^k$ for each k ($k = 1,..., m$).

Case 0. P^k is an axiom $A \vee \sim A$ of $\mathcal{G}$, where A is atomic.

Clearly $\square P^k\{A^0 \vee \sim A^0\}$, where A^0 is a λ-normal form of A.

For the sake of brevity, in all the cases below we shall assume that P^k is inferred from P^i [or from P^i and P^j in Case 5] by the indicated rule of inference of $\mathcal{G}$, where $i < k$ [and $j < k$]. By inductive hypothesis we are given that $\square P^i\{\mathcal{I}\}$ (where $\mathcal{I}$ is the d-seq $W^0,W^1,...,W^n$), and we must show that $\square P^k$.

Case 1. P^k is inferred by λ-conversion.

Then $\square P^k\{\mathcal{I}\}$.

Case 2. P^k is inferred by disjunction rules.

Clearly one can apply the same disjunction rules as were used to obtain P^k from P^i to each wff in $\mathcal{I}$ to obtain a verification of P^k.

Case 3. P^k is $P^i \vee A$.

Let A^0 be an open form of A. $\square P^k\{W^0 \vee A^0, W^1 \vee A^0,..., W^n \vee A^0\}$.

Case 4. P^i is $M \vee A$ and P^k is $M \vee \sim\sim A$.

W^0 has the form $M^0 \vee A^0$. For each p, let U^p be obtained from W^p by replacing the descendant D of A^0 in W^p by the wff $\sim\sim D$. It is easy to see that $\square P^k\{U^0,...,U^n\}$.

Case 5. P^i is $M \vee \sim A$ and P^j is $M \vee \sim B$ and P^k is $M \vee \sim[A \vee B]$.

Since M has an essentially unique open form (modulo renaming of variables), we may assume that we are given verifications $\mathcal{I} = (W^0,..., W^n)$ for $M \vee \sim A$ and $\mathcal{J} = (U^0,..., U^q)$ for $M \vee \sim B$, where W^0 is $M^0 \vee \sim A^0$, U^0 is $M^0 \vee \sim B^0$, and M^0 is an open form of M. Also, we may assume that the only variables which occur free both in wffs of $\mathcal{I}$ and in wffs of $\mathcal{J}$ are those which occur free both in $M^0 \vee \sim A$ and in $M^0 \vee \sim B$. Clearly W^n has the form $M^a \vee \sim A^1$ and U^q has the form $M^b \vee \sim B^1$, where M^a, M^b, $\sim A^1$, and $\sim B^1$ are descended from M^0, M^0, $\sim A^0$, and $\sim B^0$, respectively. (See Figure 4-1.)

We now describe a verification of $M \vee \sim[A \vee B]$. Its initial wff X^0 is $M^0 \vee \sim[A^0 \vee B^0]$. Next, for each critical occwlp R of M^0, perform an

Figure 4-1: Notations for Case 5

$$P^i = [M \vee \sim A] \qquad P^j = [M \vee \sim B] \qquad P^k = [M \vee \sim[A \vee B]]$$

$$W^0 = [M^0 \vee \sim A^0] \qquad U^0 = [M^0 \vee \sim B^0] \qquad X^0 = [M^0 \vee \sim[A^0 \vee B^0]]$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$W^n = [M^a \vee \sim A^1] \qquad U^q = [M^b \vee \sim B^1] \qquad X^r = [M^c \vee \sim[A^0 \vee B^0]]$$

$$\vdots$$

$$X^{r+n} = [M^d \vee \sim[A^1 \vee B^0]]$$

$$\vdots$$

$$X^{r+n+q} = [M^e \vee \sim[A^1 \vee B^1]]$$

∃-duplication; the two copies of R thus produced will be called the *left* and the *right* copies. This produces a development $X^r = [M^c \vee \sim[A^0 \vee B^0]]$ of $M \vee \sim[A \vee B]$, where $\models[M^c \equiv M^0]$. Continue from here by imitating the operations in the verification I; whenever an operation used to construct I was applied to an occwfp of a descendant of a critical occwfp R of M^0, perform that operation within the descendant of the left copy of R in M^c. Whenever an operation in I was applied to an occwfp in a descendant of $\sim A^0$ (in W^0), perform the same operation to the corresponding occwfp in the descendant of the copy of $\sim A^0$ in X^r. One thus obtains a development X^{r+n} of $M \vee \sim[A \vee B]$. Now continue by imitating the operations in J, but use the occwfps in descendants of right copies of occwfps of M^0. One thus obtains a development $X^{r+n+q} = [M^e \vee \sim[A^1 \vee B^1]]$ of $M \vee \sim[A \vee B]$. Note that all this is possible without changing any of the variables which were introduced in I and J.

Lemma. Let C be any very accessible occwfp of M^0, and let C^a, C^b, and C^e be the descendants of C in M^a, M^b, and M^e, respectively. Then $\models[C^a \vee C^b] \supset C^e$ if C is positive in M^0, and $\models C^e \supset [C^a \wedge C^b]$ if C is negative in M^0.

The lemma is proved by induction on the construction of C.

Case a. C is atomic.

Then C^e, C^a, and C^b are all the same as C, so $\models[C^a \vee C^b] \supset C^e$ and $\models C^e \supset [C^a \wedge C^b]$.

Case b. C is critical in M^0.

Then C^e is $[C^a \wedge C^b]$, so $\models C^e \supset [C^a \wedge C^b]$. Since C is negative in M^0, this is

the desired conclusion.

Case c. C has the form $\sim D$.

Then C^e is $\sim D^e$, C^a is $\sim D^a$, and C^b is $\sim D^b$. If C is positive in M^o, then D is negative in M^o, so $\models D^e \supset [D^a \wedge D^b]$ by inductive hypothesis, so $\models [C^a \vee C^b] \supset C^e$. Similarly, if C is negative in M^o, then $\models [D^a \vee D^b] \supset D^e$, so $\models C^e \supset [C^a \wedge C^b]$.

Case d. C has the form $[D \vee E]$.

Then C^i is $[D^i \vee E^i]$ for i = a, b, e. If C is positive in M^o, then D and E are too, so by inductive hypothesis $\models [D^a \vee D^b] \supset D^e$ and $\models [E^a \vee E^b] \supset E^e$, so $\models [C^a \vee C^b] \supset C^e$. If C is negative in M^o, then $\models D^e \supset [D^a \wedge D^b]$ and $\models E^e \supset [E^a \wedge E^b]$, so $\models C^e \supset [[D^a \wedge D^b] \vee [E^a \wedge E^b]$, so $\models C^e \supset [[D^a \vee E^a] \wedge [D^b \vee E^b]]$, so $\models C^e \supset [C^a \wedge C^b]$.

This completes the proof of the lemma, since every very accessible occwfp of the open wff M^o must fall under one of these cases.

We now apply the Lemma to see that $\models M^a \vee M^b \supset M^e$. Since $\models W^n$ and $\models U^q$ we know that $\models M^a \vee \sim A^1$ and $\models M^b \vee \sim B^1$, so $\models M^e \vee \sim [A^1 \vee B^1]$. Thus $\models X^{r+n+q}$, so $\square[M \vee \sim[A \vee B]]\{X^o, \ldots, X^{r+n+q}\}$. This concludes the treatment of Case 5.

Case 6. P^i is $M \vee \sim \Pi_{o(o\alpha)} A_{o\alpha} \vee \sim A_{o\alpha} B_\alpha$ and P^k is $M \vee \sim \Pi_{o(o\alpha)} A_{o\alpha}$.

By the argument in Case 4 we see that $\square\, M \vee \sim\sim[\sim \Pi_{o(o\alpha)} A_{o\alpha} \vee \sim A_{o\alpha} B_\alpha]$, so this wff has a verification $U^o, \ldots, U^n$, where U^o has the form $M^o \vee \sim\sim[\sim \Pi_{o(o\alpha)} A^o{}_{o\alpha} \vee \sim C_o]$, and $\sim C_o$ is an open form of $\sim A_{o\alpha} B_\alpha$. Since $A^o{}_{o\alpha}$ is a λ-normal form of $A_{o\alpha}$, and $\sim C_o$ is an open form of $\sim A^o{}_{o\alpha} B_\alpha$, it is easy to see that

$\square\, P^k\{M^o \vee \sim \Pi_{o(o\alpha)} A^o{}_{o\alpha},$
$M^o \vee \sim[\Pi_{o(o\alpha)} A^o{}_{o\alpha} \wedge \Pi_{o(o\alpha)} A^o{}_{o\alpha}]$
(which is $M^o \vee \sim\sim[\sim \Pi_{o(o\alpha)} A^o{}_{o\alpha} \vee \sim \Pi_{o(o\alpha)} A^o{}_{o\alpha}]$),
$M^o \vee \sim\sim[\sim \Pi_{o(o\alpha)} A^o{}_{o\alpha} \vee \sim C_o]$,
$U^1, \ldots, U^n\}$.

Case 7. P^i is $M \vee A_{o\alpha} x_\alpha$ and P^k is $M \vee \Pi_{o(o\alpha)} A_{o\alpha}$, and x_α is not free in M or $A_{o\alpha}$.

Clearly $\square\, P^k\{I\}$, since any open form of P^i is also an open form of P^k.

Since we have now considered all the rules of inference of $\mathcal{G}$, the proof of the metatheorem is complete.

I. Appendix: Rules of Inference

$\mathcal{H}$ denotes a (possibly empty) set of wffs_o, and $\mathcal{H}$,A denotes $\mathcal{H} \cup \{A\}$.

Hypothesis Rule (Hyp): Infer $\mathcal{H}$, $A \vdash A$.

Deduction Rule (Deduct): From $\mathcal{H}$, $A \vdash B$ infer $\mathcal{H} \vdash A \supset B$.

Rule of Propositional Calculus (RuleP): From $\mathcal{H}_1 \vdash A_1$, . . . and $\mathcal{H}_n \vdash A_n$ infer $\mathcal{H}_1 \cup \ldots \cup \mathcal{H}_n \vdash B$, provided that $[[A_1 \wedge \ldots \wedge A_n] \supset B]$ is tautologous. (For the case that $n = 0$, this rule takes the form: infer $\mathcal{H} \vdash B$, provided that B is tautologous.)

Negation-Quantifier Rule (RuleQ): From $\mathcal{H} \vdash A$ infer $\mathcal{H} \vdash B$, where A is $\sim\forall xC$, $\sim\exists xC$, $\forall x\sim C$, or $\exists x\sim C$, and B is $\exists x\sim C$, $\forall x\sim C$, $\sim\exists xC$, or $\sim\forall xC$, respectively.

Rule of Indirect Proof (Indirect): From $\mathcal{H}$, $\sim A \vdash \perp$ infer $\mathcal{H} \vdash A$.

Rule of Cases (Cases): From $\mathcal{H} \vdash A \vee B$ and $\mathcal{H}$, $A \vdash C$ and $\mathcal{H}$, $B \vdash C$ infer $\mathcal{H} \vdash C$.

Universal Generalization ($\forall$G): From $\mathcal{H} \vdash A$ infer $\mathcal{H} \vdash \forall xA$, provided that x is not free in any member of $\mathcal{H}$.

Existential Generalization ($\exists$G): Let A(x) be a wff and let t be a term which is free for x in A(x). (t may occur in A(x).) From $\mathcal{H} \vdash A(t)$ infer $\mathcal{H} \vdash \exists xA(x)$.

Universal Instantiation ($\forall$I): From $\mathcal{H} \vdash \forall xA(x)$ infer $\mathcal{H} \vdash A(t)$, provided that t is a term free for x in A(x).

When one has inferred a wff of the form $\exists xA$, one often "existentially instantiates" to obtain the wff A, which asserts that x (a free variable of A) is an entity of the sort whose existence is asserted by $\exists xA$. We shall simply regard A as an additional hypothesis, which is eventually to be eliminated by Rule C below. (The name Rule C was introduced by Rosser [15], but our formulation of the rule differs from his.)

Rule C: From $\mathcal{H} \vdash \exists xA$ and $\mathcal{H}$, $A \vdash B$ infer $\mathcal{H} \vdash B$, when x is not free in B or in any member of $\mathcal{H}$.

Rules of Alphabetic Change of Bound Variable ($\alpha\beta$) and λ-conversion (λ). See [6] or [1] for the formal statements of these rules.

Rule of Definition (Def): Eliminate or introduce a definition.

Bibliography

1. Peter B. Andrews, *Resolution in Type Theory*, Journal of Symbolic Logic 36 (1971), 414-432.

2. Peter B. Andrews, *Provability in Elementary Type Theory*, Zeitschrift fur Mathematische Logic und Grundlagen der Mathematik 20 (1974), 411-418.

3. Peter B. Andrews, "Transforming Matings into Natural Deduction Proofs," in *5th Conference on Automated Deduction, Les Arcs, France*, edited by W. Bibel and R. Kowalski, Lecture Notes in Computer Science 87, Springer-Verlag, 1980, 281-292.

4. Peter B. Andrews, *Theorem Proving via General Matings*, Journal of the Association for Computing Machinery 28 (1981), 193-214.

5. W. W. Bledsoe. A Maximal Method for Set Variables in Automatic Theorem-proving. Machine Intelligence 9, 1979, pp. 53-100.

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

7. Gérard P. Huet. A Mechanization of Type Theory. Proceedings of the Third International Joint Conference on Artificial Intelligence, IJCAI, 1973, pp. 139-146.

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

9. D. C. Jensen and T. Pietrzykowski, *Mechanizing ω-Order Type Theory Through Unification*, Theoretical Computer Science 3 (1976), 123-171.

10. Dale A. Miller, Eve Longini Cohen, Peter B. Andrews, "A Look at TPS," in *6th Conference on Automated Deduction, New York*, edited by Donald W. Loveland, Lecture Notes in Computer Science 138, Springer-Verlag, June 1982, 50-69.

11. Dale A. Miller. *Proofs in Higher-Order Logic*, Ph.D. Th., Carnegie-Mellon University, 1983. (to appear)

12. Tomasz Pietrzykowski, *A Complete Mechanization of Second-Order Type Theory*, Journal of the Association for Computing Machinery 20 (1973), 333-364.

13. T. Pietrzykowski and D. C. Jensen. A complete mechanization of (ω)-order type theory. Proceedings of the ACM Annual Conference, Volume I, 1972, pp. 82-92.

14. J. A. Robinson. Mechanizing Higher-Order Logic. In *Machine Intelligence 4*, Edinburgh University Press, 1969, pp. 151-170.

15. J. Barkley Rosser, *Logic for Mathematicians*, McGraw-Hill, 1953.

16. Moto-o-Takahashi, *A proof of cut-elimination theorem in simple type-theory*, Journal of the Mathematical Society of Japan 19 (1967), 399-410.