Typing and Proof by Cases
in Program Verification

by

W. W. Bledsoe and Mabry Tyson

May 1975

ATP 15

* The work reported here was supported by NSF Grant #DCR74-12886.

Typing and Proof by Cases
in Program Verification

W.W. Bledsoe and Mabry Tyson

ABSTRACT

Special procedures have been added to an automatic prover to facilitate its handling of inequalities and proof by cases. A data base, called TYPELIST, is used which maintains upper and lower bounds of variables occuring in the proof of a theorem. These procedures have been coded and used to (interactively) prove several theorems arising in automatic program verification.

Introduction

We describe here procedures that have been added to an automatic theorem prover [1] to make it more effective in proving verification conditions (theorems) that arise in the field of program verification. These procedures, which handle inequalities and equalities, and proof by cases, are based upon a pointer system used by Bundy [2], SRI [3,4], and others to handle inequalities, and upon the interval types used in [5]. The present description follows somewhat the discussion in [6].

In order to follow this presentation the reader should have some understanding of the prover described in [1]. However we feel that many workers in this field are already generally familiar with our prover and can read this paper directly, referring to [1] only when the need arises. Tables I and II from [1] are included here in Appendix 1, for convenience, but the reader is referred to Section 2 of [1] for a fuller understanding.

These methods can also be used in Resolution based provers and other Gentzen type systems. Section 5 gives a brief description of this for resolution.

1. Types

Typing information can be a powerful asset in automatic theorem proving. For example, knowing that j and k are non-negative integers and that $j < k$ lets us deduce that $j * k \geq 0$, $j \leq k - 1$, etc. Often, we have other "typing" information. For example, we may know (from a given hypothesis) that j lies in some interval, $a \leq j \leq b$. In our system, we have decided to include such information as part of the type of j. Thus j has the type: "non-negative integer in the interval $a \leq j \leq b$". We express this fact by the notation $\{j: a\ b\}$.

In what follows, certain variables i, j, k,... occur in inequalities and can assume only non-negative integer values. These will be "typed" as indicated above. Such variables often arise as program variables in computer programs. (Actually these variables are all universally quantified in the theorem being proved and are converted to skolem constants by the skolemization process, but that need not concern us here. Refer to Appendix 1 of [1] and Section 1, of [1].)

Upper and lower bounds are computed and maintained for these typed variables. When a new inequality is encountered, as a hypothesis, the bounds for these variables are updated appropriately. This interval information is kept in a knowledge base (which we call the TYPELIST), which represents the "state of the world for these variables at that particular time, and serves as an additional hypothesis to the theorem or subgoal being considered. For example, a hypothesis

$$(a \leq j \leq b)$$

is stored in TYPELIST as

$$\{j:\ a\ b\}$$

which means that j is in the closed interval $[a,b]$[1]. If a contradiction such as $\{j:\ k\ k-1\}$ occurs in TYPELIST, this represents a false hypothesis and successfully terminates the proof. Also if an entry $\{j:\ N\ \infty\}$ is already in TYPELIST, any new hypothesis such as $(j \leq N+1)$ causes the entry to be updated to $\{j:\ N\ \ N+1\}$, which means that j can take only the value N or the value $N+1$.

An entry of the form $\{j:\ N+1\ \ N+1\}$ which occurs in TYPELIST is treated as the equality $(j = N+1)$.

Initially all typed variables j are given the type $\{j:\ 0\ \ \infty\}$.

A subroutine SET-TYPE is used to convert information in the hypothesis of a theorem to TYPELIST entries. It is called at the beginning of the proof and at each point in the proof when new expressions are added to the hypothesis of the theorem being proved. For example, if the theorem being proved is

<u>Ex. 1</u>.

(1) $$(P(1) \wedge 1 \leq j \wedge j \leq n \wedge j \leq 1 \longrightarrow P(j))$$

the original value of TYPELIST is

$$(\{j:\ 0\ \infty\}\{n:\ 0\ \infty\})\ ,$$

[1]Except in the case when b is $+\infty$; then the interval is $[a,\infty)$.

but then SET-TYPE is called on the hypothesis of (1) which changes TYPELIST to

$$(\{j: 1\ 1\}\{n: j\ \infty\})$$

and converts (1) to

(2) $$(j = 1 \wedge P(1) \longrightarrow P(j)) \quad .$$

Notice that the program detected that j was equal to 1 from the entry {j: 1 1}. The prover will now substitute 1 for j in (2) to obtain

$$(P(1) \longrightarrow P(1))$$

which it recognizes as true.

Other examples are now given.

<u>Ex. 2</u>.

(3) $$(1 \leq j \wedge P(1) \longrightarrow (j \leq k \wedge k \leq 1 \longrightarrow P(k))) \quad .$$

An initial call to SET-TYPE, on the hypothesis of (3), changes TYPELIST to $(\{j: 1\ \infty\}\{k: 0\ \infty\})$ and converts (3) to

(4) $$(P(1) \longrightarrow (j \leq k \wedge k \leq 1 \longrightarrow P(k))) \quad .$$

Now Rule 7 of IMPLY (see [1], Table I), converts (4) to

(5) $$(P(1) \wedge j \leq k \wedge k \leq 1 \longrightarrow P(k))$$

at which time SET-TYPE is again called, which uses $j \leq k$ and $k \leq 1$ to change TYPELIST to $(\{j: 1\ 1\}\{k: 1\ 1\})$, and converts (5) to

$$(j = 1 \wedge k = 1 \wedge P(1) \longrightarrow P(k)) \quad .$$

The prover, as before, converts this to

$$(P(1) \longrightarrow P(1))$$

which it recognizes as true.

Ex. 3.

$$(2 \leq j \wedge j \leq 1 \longrightarrow P(j))$$

SET-TYPE changes TYPELIST to ({j: 2 1}). The program detects the contradictions in TYPELIST (i.e., $2 \leq 1$) and successfully concludes the proof.

Whenever an inequality $(a \leq b)$ occurs in the conclusion of the theorem being proved, the prover updates TYPELIST with the negation of $(a \leq b)$, and looks for a contradiction. Thus, for the example

Ex. 4.

(6) $$(j \leq 1 \wedge k \leq j \wedge P \longrightarrow k \leq 3) \quad ,$$

TYPELIST is given the value ({j: k 1}{k: 0 j}) and (6) is converted to

$$(P \longrightarrow k \leq 3) \quad .$$

The prover now uses $(k \not\leq 3)$, which is first converted to $(4 \leq k)$[2], to update TYPELIST, getting ({j: k 1}{k: 4 j}), which contains the contradiction

$$(4 \leq k \leq j \leq 1) \quad .$$

[2]Since k is an integer. See [7, p. 27].

The Prover detects such contradictions by computing absolute upper and lower bounds, sup and inf, for j and k. For this case

$$\sup j = 1 , \quad \inf j = 4$$
$$\sup k = 1 , \quad \inf k = 4 \quad .$$

Since $4 > 1$ we have a contradiction. The prover uses the routines SUP and INF to evaluate these bounds. In [7] we carefully define the algorithms SUP and INF and prove that they have the required properties.

Formula (6), (without the P), is an example of a formula in Presburger Arithmetic. These often arise from computer programs and are discussed in [7] and by Cooper in [8].

Ex. 5. $(2 \le j \le 4 \wedge k \le j \wedge k \le 7 \rightarrow C)$. Here we use the symbols 'max' and 'min' in typing j and k. TYPELIST is given the value [{j: max(2,k) 4}{k: 0 min(j,7)}].

2. TYPELIST in PROVER

In Section 2 of [1] we describe IMPLY and HOA, the main algorithms of Prover, and give Tables I and II which define them, and list several examples of their use. Tables I and II are reproduced in Appendix 1 of this paper for convenience. The reader is referred to the Section 2 of [1] for a fuller understanding.

IMPLY has five arguments

(TYPELIST, H, C, TL, LT) ,

but in Section 2 of [1] we deal with only H, C, and TL, the hypothesis, conclusion, and theorem label of the theorem or subgoal being proved. For convenience to the reader we represent, in this paper, a call to IMPLY(TYPELIST, H, C, TL, LT) by the notation

(TL) $\qquad (H \Rightarrow C)$.

As mentioned earlier TYPELIST represents an additional hypothesis, so we will augment this notation as follows:

(TL) $\qquad ([\text{TYPELIST}] \wedge H \Rightarrow C)$.

Thus Ex. 2., after it is partially converted, is represented by

(1) $\qquad ([\{j\colon 1\ 1\}\{k\colon 1\ 1\}] \wedge P(1) \Rightarrow P(k))$.

We will now describe some changes and additions to the Rules of IMPLY and HOA (Tables I and II, of [1]) which have been made to facilitate the use of TYPELIST. Before doing so we first describe the algorithm SET-TYPE, which was mentioned earlier.

SET-TYPE (A)

This algorithm updates TYPELIST by using inequalities and equalities in conjunctive positions of A, and returns a value A', which is the remainder of A not used in updating TYPELIST.

For example, if TYPELIST = [{j: 0 k}{k: j 7}] then a call

$$\text{SET-TYPE}(k \leq 5 \wedge P(j))$$

updates TYPELIST to

[{j: 0 k}{k: j 5}]

and returns the value P(j).

IMPLY RULE CHANGES

	IF	ACTION	RETURN
7.	$C \equiv (A \twoheadrightarrow B)$		IMPLY $(H \wedge A, B)$
	is changed to		
7.	$C \equiv (A \twoheadrightarrow B)$	Put A': = SET-TYPE(A)	
7.1	TY' has a contradiction		"T"
7.2	ELSE		IMPLY(TY', $H \wedge A'$, B)

Where TY' is the updated value of TYPELIST after the action of SET-TYPE(A).

Rule 11 and 14 are added to IMPLY

	IF	ACTION	RETURN
11.	$C \equiv (a \leq b)$	Put A': = SET-TYPE($\sim(a \leq b)$) Let TY' be the updated TYPELIST	
11.1	TY' has a contradiction		"T"
11.2	TY' $\equiv$ TYPELIST	Go to 12 (with TYPELIST and C as they were)	
11.3	TY' $\not\equiv$ TYPELIST		(T TY')
14.1	$C \equiv (a = b)$	Put $C' \equiv (a \leq b \wedge b \leq a)$	IMPLY(H, C')
14.2	$C \equiv (a \neq b)$	Put $C' \equiv (a < b \vee b < a)$	IMPLY(H, C')

Later in this description we will further change these tables, but the reader need not be concerned with that at this time. We will summarize all of these changes in Tables I-T, II-T, of Section 3.

Ex. 5. $(Q \longrightarrow (j \leq 1 \wedge k \leq j \wedge P \longrightarrow k \leq 3))$

(1) $([\{j: 0\ \infty\}\{k: 0\ \infty\}]$

$\Rightarrow (Q \longrightarrow (j \leq 1 \wedge k \leq j \wedge P \longrightarrow k \leq 3)))$

Note that each of j and k is given the original type $[0\ \infty)$, when the theorem is given to Prover.

(1) $([\{j: 0\ \infty\}\{k: 0\ \infty\}] \wedge Q$

$\Rightarrow (j \leq 1 \wedge k \leq j \wedge P \longrightarrow k \leq 3))$ I 7

In this case SET-TYPE(Q) left TYPELIST unchanged and returned the value Q.

TYPELIST H C

(1) $([\{j: k\ 1\}\{k: 0\ j\}] \wedge (Q \wedge P) \longrightarrow k \leq 3)$ I 7

Here SET-TYPE $(j \leq 1 \wedge k \leq j \wedge P)$ has updated TYPELIST to the new value shown, and returned P, which was conjoined to Q.

Now the new Rule I-11, employes SET-TYPE$(\sim(k \leq 3))$ = SET-TYPE$(4 \leq k)$ to update TYPELIST to TY' = $[\{j: k\ 1\}\{k: 4\ j\}]$, and Rule 11.1 detects the contradiction

$$4 \leq j \leq 1$$

in TY' and terminates the proof successfully.

As mentioned in Section 1, we detect the contradiction in

$$TY' = [\{j: k\ 1\}\{k: 4\ j\}]$$

(or any other list of inequalities) by computing

$$\sup_{TY'}(j) \quad \text{and} \quad \inf_{TY'}(j) \quad .$$

In this case

$$\sup_{TY'}(j) = 1, \qquad \inf_{TY'}(j) = 4 \quad ,$$

and since $4 > 1$ we have a contradiction. These are computed by the algorithms SUP and INF (See [7], especially Section 3). In this example the values of sup and inf are rather obvious; for more involved examples see Section 5 of [7].

We have decided to give each variable j just <u>one</u> interval {j: a b} in TYPELIST. So if we are proving a goal of the form

$$((j \leq 1 \vee j \geq 5) \wedge H \Longrightarrow C) \quad ,$$

where there is a <u>disjunction</u> of inequalities in the hypothesis, then we use <u>two</u> TYPELIST's expressed in the form

$$(([\{j: 0\ 1\}\{k: \quad \}\cdots] \vee [\{j: 5\ \infty\}\{k: \quad \}\cdots])$$
$$\wedge H \Longrightarrow C).$$

To handle such examples we add Rule 2 to IMPLY to split such goals into two subgoals.

2.	TYPELIST ≡ TY' ∨ TY"	Put θ: = IMPLY(TY',H,C)	
2.1	θ ≡ NIL		NIL
2.2	θ ≢ NIL	Put λ: = IMPLY(TY",H,C)	
2.3	λ ≡ NIL		NIL
2.4	λ ≢ NIL		σ ∘ λ

<u>Ex. 7</u>. $(k \leq 3 \longrightarrow k \leq 1 \vee 2 \leq k \leq 3)$.

(1) $(\{k:\ 0\ \infty\} \Rightarrow (k \leq 3 \longrightarrow k \leq 1 \vee 2 \leq k \leq 3))$

(1) $(\{k:\ 0\ 3\} \Rightarrow k \leq 1 \vee 2 \leq k \leq 3)$ I 7

$(\{k:\ 0\ 3\} \wedge \sim(2 \leq k \leq 3) \Rightarrow k \leq 1)$ H 4.2

$(\{k:\ 0\ 3\} \wedge (k \leq 1 \vee 4 \leq k) \Rightarrow k \leq 1)$

$((\{k:\ 0\ 1\} \vee \{k:\ 4\ 3\}) \Rightarrow k \leq 1)$

(1 1) $(\{k:\ 0\ 1\} \Rightarrow k \leq 1)$ I 2

Rule 10' uses $\sim(k \leq 1)$ to update TYPELIST to $\{k:\ 2\ 1\}$ and Rule 10.2 detects the contradiction. "T"

(1 2) $(\{k:\ 4\ 3\} \Rightarrow k \leq 1)$

Proved since $\{k:\ 4\ \ 3\}$ is a contradiction.

3. Cases

Many of the theorems (verification conditions) from program validation require a proof by cases, in that the theorem must be proved separately for two different ranges of values for some variable. Ex. 7 is such a case, but there the proof was straightforward because the two cases,

$$k \leq 1 \quad \text{and} \quad 2 \leq k \leq 3$$

were stated explicitly in the theorem.

On the other hand, consider the following equivalent form of Ex. 7.

<u>Ex. 8</u>. $((k \leq 3 \wedge (k \leq 1 \longrightarrow C) \wedge (2 \leq k \leq 3 \longrightarrow C) \longrightarrow C)$.

(1) $(\{k\!: 0\ 3\} \wedge (k \leq 1 \longrightarrow C) \wedge (2 \leq k \leq 3 \longrightarrow C) \Rightarrow C)$ I 7

Backchaining (Rule H 7) off of the hypothesis $(k \leq 1 \longrightarrow C)$ we obtain the subgoal

(1 H) $(\{k\!: 0\ 3\} \wedge (k \leq 1 \longrightarrow C) \wedge (2 \leq k \leq 3 \longrightarrow C) \longrightarrow k \leq 1)$

which is false. Similarly if we backchain off of the hypothesis $(2 \leq k \leq 3 \longrightarrow C)$ we fail again.

If the prover could somehow be made to know that it should consider the two cases

$$k \leq 1 \quad \text{and} \quad 2 \leq k \leq 3$$

as it did in Ex. 7 the proof would proceed routinely.

We could, of course, require that prover backchain off of both of these hypotheses and thereby set up the provable subgoal

$$(k \leq 1 \vee 2 \leq k \leq 3) \quad ,$$

but such a rule is not only unnatural, it is combinatorially explosive. What's more, a similar problem arises in many other theorems, such as

Ex. 9.
$$\begin{aligned} &(1 \leq n) \\ &\wedge \forall m\ (2 \leq n \wedge 1 \leq m \wedge m \leq 1 \rightarrow A[m] \leq A[2]) \\ &\wedge \forall k\ (k+1 \leq n \wedge 2 \leq k \rightarrow A[k] \leq A[k+1]) \\ &\rightarrow \forall\ K(K+1 \leq n \wedge 1 \leq K \rightarrow A[K] \leq A[K+1]) \end{aligned}$$

and Example 10 below, which are more complicated than Exercise 8 and which will not submit to such an attack.

The procedure we employ to prove Ex. 8 and all others like it, forces the prover into a proof by cases in a natural way. This is effected by further changes and additions to Tables 1 and 2. These are shown (for the most part) in Tables I-T and II-T below. These changes are justified by the results in Appendix 2.

These changes require that IMPLY and HOA now return a pair

$$(\theta \quad TY') \quad ,$$

where θ is the same substitution we got before, and TY' is a new value of TYPELIST which can be used in subsequent calls to IMPLY. This outputted value TY' represents the part of the theorem that has not been proved. Thus if $(\theta \quad TY')$ is returned from a call IMPLY (TYPELIST, H, C), it means that $(TYPELIST \wedge H \rightarrow C)$ is valid except for the case TY', or that

$$(\sim TY' \wedge TYPELIST \wedge H \rightarrow C)$$

is valid. See Appendix 2.

Table I-T

TYPELIST VERSION

IMPLY RULE CHANGES*

	IF	ACTION	RETURN
2.	TYPELIST ≡ (TY' ∨ TY")	Put Z: = IMPLY(TY', H, C)	
2.1	Z ≡ NIL		NIL
2.2	Z ≡ (θ TY1)	Put Z2: = IMPLY(TY", H, C)	
2.3	Z2 ≡ NIL		NIL
2.4	Z2 ≡ (θ2 TY2)		(θ ∘ θ2 (TY1 ∨ TY2))
3.	H ≡ (A ∨ B)	Put Z: = IMPLY(TYPELIST, A, C)	
3.1	Z ≡ NIL		NIL
3.2	Z ≡ (θ TY1)	Put Z2: = IMPLY(TYPELIST, Bθ, C)	
3.3	Z2 ≡ NIL		NIL
3.4	Z2 ≡ (θ2 TY2)		(θ ∘ θ2 (TY1 ∨ TY2))
4.	C ≡ (A ∧ B)	Put Z: = IMPLY(TYPELIST, H, A)	
4.1	Z ≡ NIL		NIL
4.2	Z ≡ (θ TY1)	Put Z2: = IMPLY(TYPELIST, H, Bθ)	
4.3	Z2 ≡ NIL		NIL
4.4	Z2 ≡ (θ2 TY2)		(θ ∘ θ2 (TY1 ∨ TY2))
7.	C ≡ (A ⟶ B)	Put A': = SET-TYPE(A). TY' is the updated TYPELIST	
7.1	TY' has a contradiction		(T NIL)
7.2	ELSE		IMPLY(TY', H∧A', B)

*IMPLY has arguments (TYPELIST, H, C, TL, LT). H is the hypothesis and C the conclusion. We are ignoring TL and LT here.

Table I-T (Continued)

11.	$C \equiv (a \leq b)$	Put A': = SET-TYPE($\sim$ $(a \leq b)$) TY' is the updated TYPELIST	
	11.1 TY' has a contradiction		(T NIL)
	11.2 TY' $\equiv$ TYPELIST	Go to 12	
	11.3 TY' $\not\equiv$ TYPELIST[3]		(T TY')

[3] If TY' has an equality entry of the form {k: t t} then k is replaced by t in H, C, and TY'.

Table II-T

TYPELIST VERSION

HOA RULE CHANGES*

	IF	ACTION	RETURN
4.	$C \equiv A \vee D$	Put Z: = HOA(B ∧ ~D, A)	
4.1	$Z \equiv$ NIL		HOA(B ∧ ~ A, D)
4.2	$Z \equiv (\theta$ TY1)	Go to 4.3.	
4.3	TY1 $\equiv$ NIL		(θ NIL)
4.4	TY1 $\not\equiv$ NIL	Put Z2: = IMPLY(TY1, B ∧ ~ A, D)	
4.5	Z2 $\equiv$ NIL		(θ TY1)
4.6	Z2 $\equiv$ (θ2 TY2)		($\theta \circ \theta$2 TY2)
6.	$B \equiv A \wedge D$	Put Z: = HOA(A, C)	
6.1	$Z \equiv$ NIL		HOA(D, C)
6.2	$Z \equiv (\theta$ TY1)	Go to 6.3.	
6.3	TY1 $\equiv$ NIL		(θ NIL)
6.4	TY1 $\not\equiv$ NIL	Put Z2: = IMPLY(TY1, D, C)	
6.5	Z2 $\equiv$ NIL		(θ TY1)[4]
6.6	Z2 $\equiv$ (θ2 TY2)		($\theta \circ \theta$2 TY2)
7.	$B \equiv (A \longrightarrow D)$	Put θ: = ANDS(D, C)	
7.1	$\theta \equiv$ NIL	GO TO 7E	
7.2	$\theta \not\equiv$ NIL	Put Z2: = IMPLY(TYPELIST, H, Aθ)	
7.3	Z2 $\equiv$ NIL		NIL
7.4	Z2 $\equiv$ (θ2 TY2)		($\theta \circ \theta$2 TY2)

[4] In case Z2 ≡ NIL it repeats Rule 6 (once) with $D \wedge A$ instead of $A \wedge D$. If on this second time Z2 = NIL then (θ TY1) is returned.

*HOA has arguments (B,C,HL). B is the hypothesis and C the conclusion. We are ignoring HL here.

Table II-T (Continued)

7E. $B \equiv (A \longrightarrow a = b)$	Put Z: = HOA(a = b, C)	
7E.1 $Z \equiv$ NIL	Go to 7LE	
7E.2 $Z \equiv (\theta$ TY1)	Put Z2: = IMPLY(TYPELIST, H, Aθ)	
7E.3 Z2 $\equiv$ NIL		NIL
7E.4 Z2 $\equiv (\theta 2$ TY2)		$(\theta \circ \theta 2$ (TY1 $\vee$ TY2))
7LE. $B \equiv (A \quad a \leq b)$	Put A': = SET-TYPE(a $\leq$ b) Let TY' be the updated TYPELIST	
7LE.1 TY' $\equiv$ TYPELIST	Go to 8	
7LE.2 TY' $\not\equiv$ TYPELIST	Put Z: = IMPLY(TY',H,C)	
7LE.3 $Z \equiv$ NIL		NIL
7LE.4 $Z \equiv (\theta$ TY1)	Put Z2: = IMPLY(TYPELIST,H, Aθ)	
7LE.5 Z2 $\equiv$ NIL		NIL
7LE.6 Z2 $\equiv (\theta 2$ TY2)		$(\theta \circ \theta 2$ (TY1 $\vee$ TY2))

The other rules of IMPLY and HOA should be changed similarly, always changing an output

$$\theta$$

to

$$(\theta \text{ NIL}) \quad .$$

These changes are best explained by the use of examples.

In the following proofs, the theorem label (X h1) is used to indicate that the first hypothesis is being used to try to prove the subgoal (X). Similarly for (X h2), etc. Also the label (X h2 H) is used to indicate that, after backchaining on the second hypothesis (see Rule H7), it is now trying to prove the hypothesis of the second hypothesis, etc.

<u>Ex. 8</u>. $(k \leq 3 \wedge \underset{\alpha}{(k \leq 1 \longrightarrow C)} \wedge \underset{\beta}{(2 \leq k \leq 3 \longrightarrow C)} \longrightarrow C)$

(1) $(\{k{:}\ 0\ 3\} \wedge (k \leq 1 \longrightarrow C) \wedge (2 \leq k \wedge k \leq 3 \longrightarrow C) \Rightarrow C)$ I 7

(1 h1) $(\{k{:}\ 0\ 3\} \wedge (k \leq 1 \longrightarrow C) \Rightarrow C)$ H 6

(1 h1 H) $(\{k{:}\ 0\ 3\} \wedge \alpha \wedge \beta \Rightarrow k \leq 1)$ H 7, 7.2

SET-TYPE$(\sim(k \leq 1))$, $2 \leq k$ I 11

TY' = $\{k{:}\ 2\ 3\}$, has no contradiction.

Returns (T $\{k{:}\ 2\ 3\}$) for (1 h1 H) I 11.3

and for (1 h1) H 7.4

(1 h2) $(\{k{:}\ 2\ 3\} \wedge \beta \Rightarrow C)$ H 6.4

(1 h2 H) $(\{k{:}\ 2\ 3\} \wedge \alpha \wedge \beta \Rightarrow 2 \leq k \wedge k \leq 3)$ H 7, 7.2

(1 h2 H1) $(\{k{:}\ 2\ 3\} \wedge \alpha \wedge \beta \Rightarrow 2 \leq k)$ I 4

SET-TYPE$(\sim(2 \leq k))$, $k \leq 1$ I 11

TY' = $\{k{:}\ 2\ 1\}$, has a contradiction

Returns (T NIL) I 11.1

(1 h2 H2) $(\{k{:}\ 2\ 3\} \wedge \alpha \wedge \beta \Rightarrow k \leq 3)$ I 4.2

SET-TYPE$(\sim(k \leq 3))$, $4 \leq k$

TY' = $\{k{:}\ 4\ 3\}$, has a contradiction.

Returns (T NIL) I 11.1

Returns (T NIL) for (1 h2 H) I 4.2

Returns (T NIL) for (1 h2) H 7.4

Returns (T NIL) for (1) H 6.6

Thus the theorem is true.

<u>Ex. 9.</u> $(1 \leq n)$

$$\wedge \forall m(2 \leq n \wedge 1 \leq m \wedge m \leq 1 \longrightarrow A[m] \leq A[2])$$

$$\wedge \forall k(k \leq n \wedge 2 \leq k \longrightarrow A[k] \leq A[k+1])$$

$$\longrightarrow \forall K(K \leq n \wedge 1 \leq K \longrightarrow A[K] \leq A[K+1])$$

(1) $$(1 \leq n \wedge \overbrace{(2 \leq n \wedge 1 \leq m \wedge m \leq 1 \longrightarrow A[m] \leq A[2])}^{\alpha}$$

$$\wedge \overbrace{(k \leq n \wedge 2 \leq k \longrightarrow A[k] \leq A[k+1])}^{\beta}$$

$$\longrightarrow (K \leq n \wedge 1 \leq K \longrightarrow \overbrace{A[K] \leq A[K+1]}^{\gamma}))$$

n and K are skolem constants

(1) $(\overbrace{[\{K\colon 1\ n\}\ \ \{n\colon K\ \ \infty\}]}^{TY} \wedge \alpha \wedge \beta \Rightarrow A[K] \leq A[K+1])$ I 7

(1 h1) $(\alpha \Rightarrow \gamma)$ Returns NIL H 6

(1 h2) $(\beta \Rightarrow \gamma)$ H 6.1

$(A[k] \leq A[k+1] \longrightarrow A[K] \leq A[K+1])$, $\{K/k\}$ H 7

(1 h2 H) $(TY \wedge \alpha \wedge \beta \Rightarrow K \leq n \wedge 2 \leq K)$ H 7.2

(1 h2 H1) $(TY \wedge \alpha \wedge \beta \Rightarrow K \leq n)$ I 4

SET-TYPE$(\sim(K \leq n))$, $n \leq K - 1$ I 11

$TY' = [\{K\colon n+1\ n\}\ \{n\colon K\ K-1\}]$,

has a contradiction, so returns (T NIL) I 11.1

(1 h2 H2) $(TY \wedge \alpha \wedge \beta \Rightarrow 2 \leq K)$ I 4.2

SET-TYPE $(\sim(2 \leq K))$, $K \leq 1$ I 11

$TY'' = [\{K\colon 1\ \min(1,n)\}\{n\colon K\ \infty\}]$

$TY'' = [\{K\colon 1\ 1\}\{n\colon K\ \infty\}]$

Here min(1,n) is converted automatically to 1, because it deduces that

$$n \geq K \geq 1 \quad .$$

TY'' has no contradiction but the program detects {K: 1 1} in TY'' and therefore replaces K by 1 in H, C, and TY'', (and in γ for (1 h1) below). Thus $(A[K] \leq A[K+1])$ becomes $(A[1] \leq A[2])$ and TY'' becomes

$$TY''' = [\{K: 1\ 1\}\{n: 1\ \infty\}] \quad .$$

It then returns (T TY''') for (1 h2 H2). I 11.3

It then returns (T TY''') for (1 h2 H) . H 7.4

It then returns (K/k TY''') for (1 h2) . I 4.4

(1 h1) $(TY''' \wedge \alpha \Rightarrow A[1] \leq A[2])$ H 6.4 and Footnote 4

$(A[m] \leq A[2] \Rightarrow A[1] \leq A[2])$, 1/m H 7

(1 h1 H) $(TY''' \wedge \alpha \wedge \beta \Rightarrow 2 \leq n \wedge 1 \leq 1 \wedge 1 \leq 1)$ H 7.2

(1 h1 H1) $(TY''' \wedge \alpha \wedge \beta \Rightarrow 2 \leq n)$ I 4

SET-TYPE$(\sim(2 \leq n))$, $n \leq 1$ I 11

TY'' = [{K: 11}{n: 1 1}]

Replaces n by 1 throughout and I 11.1

Returns (T NIL) for (1 h1 H1)

(1 h1 H2) $(TY''' \wedge \alpha \wedge \beta \Rightarrow 1 \leq 1 \wedge 1 \leq 1)$

Returns (T NIL) by REDUCE

Returns (T NIL) for (1 h1 H) H 7.4

Returns (1/m NIL) for (1 h1) H 4.4.3

Returns ((K/k 1/m)NIL) H 6.6

Thus the theorem is true.