Correctness of a Lucid Interpreter Based on Linked Forest Manipulation Systems

by

Mansour Farah
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

Research Report CS-78-07

June 1978

Correctness of a Lucid Interpreter
Based on Linked Forest Manipulation Systems

Mansour Farah

Department of Mathematics
Université de Moncton
Moncton, N. B., Canada

Abstract

The non-procedural programming language, Lucid, is described formally using a model based on linked forest manipulation systems. In this model the semantics is defined computationally by an abstract interpreter which is essentially non-deterministic and involves parallelism. This computational semantics is proven to be totally correct with respect to the denotational semantics of Lucid.

1. Introduction

The notion of Linked-Forest Manipulation System has been introduced in [4, 5] as a powerful tool in computational semantics and in particular for the formal description of programming languages. The basic objects which are manipulated in such a system are linked trees, i.e. rooted multilabelled trees with pointers. In describing a programming language, the syntax and semantics can be in two parts. Essentially, the syntax part defines (in a constructive way) a mapping from any syntactically correct program to the corresponding linked tree. The semantics part defines some transformations on such a linked tree leading to a final linked tree on which the results of the computations are shown.

The formal descriptions of several conventional programming languages using linked-forest manipulation systems (l.f.m.s.) have been given, e.g. [6, 8]. These descriptions are at the same time precise and readable.

In this paper the computational description of Lucid, a non-procedural language, is given and shown to be totally correct with respect to its denotational semantics. This language differs from the more conventional programming languages in that it is not sequential and has operators which require parallel computations. A goal oriented demand driven interpretation scheme is at the basis of the semantics given here. A similar scheme has been used in [3] to give a deterministic operational semantics for the same language. However, in this description non-determinism and parallelism are handled very elegantly by the l.f.m.s. that defines the semantics of Lucid.

The tools necessary for proving the equivalence of two models, one of which is based on l.f.m.s., had to be developped before being able to state and prove the correctness of the computational semantics of Lucid that is given here. Moreover, these tools, and more specifically the notion of transformation on subtrees, would allow the expression of properties of l.f.m.s. in general.

In the next section we give the computational description of Lucid. The basic tools for expressing properties of an l.f.m.s. are introduced in section 4. The following two sections deal respectively with the partial correctness and the consistency of this computational semantics, thus showing the total correctness of the interpretation scheme for Lucid.

2. Computational Description of Lucid

2.1. Syntax Description

Table I gives the syntax rules for Lucid programs. ID is a set of identifiers including the identifiers INPUT and OUTPUT. The constants are elements of the domain $D = \mathbb{Z} \cup \{T, F\}$, i.e. consist of integers and booleans. Terms are formed from constants, variables, unary operators, binary operators and the if-then-else operator. The set of unary operators is UNOP = {first, next, latest, latest^{-1} , − , ¬}, that of the binary operators is BIOP = {asa, fby, +, *, −, /, ↑, >, ≥, <, ≤, eq, ∧, ∨} . The priorities of the operators which is implied by the context-free productions is such that the unary operators have higher priority than the binary operators, which in turn have higher priority than the if-then-else operator.

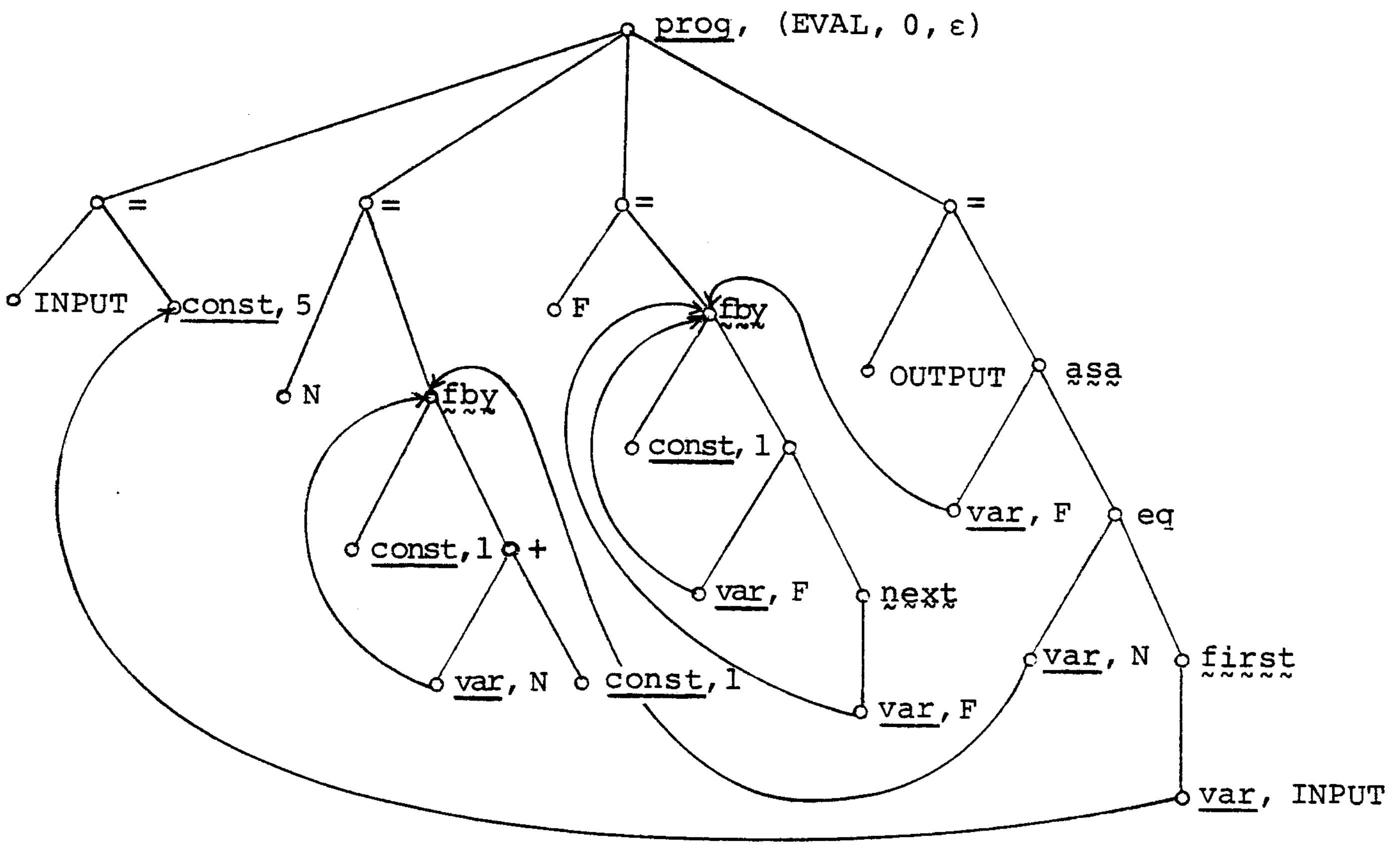
An assertion consists of an identifier, representing a variable, followed by the "=" sign, followed by a term. A program consists of a list of assertions followed by a natural number which indicates the instance of variable OUTPUT to be computed. The l.f.m.s. associated with the non-terminal <program> checks for multiple definitions of a variable, and for the definition of the special variable OUTPUT. Then it links every point of usage of a variable with its defining expression. Finally it puts back the names of the variables at their usage points. This latter action is only necessary to be able to carry out the proof of correctness.

Example

The following Lucid program computes the factorial of a positive integer given as input (in this case it is 5).

```
INPUT = 5;
N = 1 fby N + 1;
F = 1 fby F * next F ;
OUTPUT = F  asa  N eq first INPUT ;
0
```

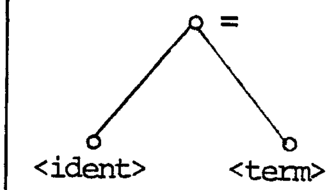
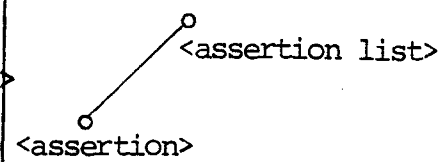
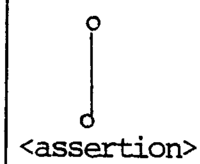
According to the syntax description of Lucid given in Table I, the linked tree corresponding to this factorial program is as follows,

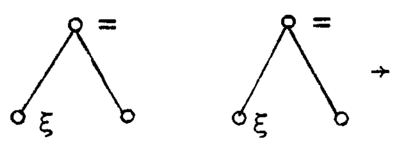
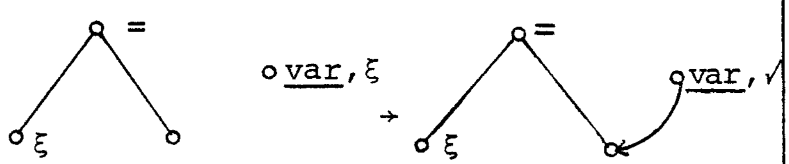
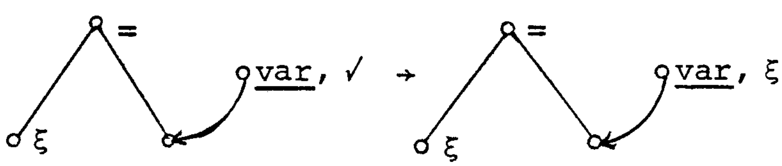


The pointers are constructed in the l.f.m.s. associated with the non-terminal <program> .

Table I - Syntax of Lucid

1	<ident> → ξ $\xi \in$ ID	o ξ
2	<const> → χ $\chi \in \mathbb{D}$	o const, χ
3	<prim> → <const>	o <const>
4	<prim> → <ident>	o var <ident>
5	<prim> → α <prim> $\alpha \in$ UNOP	o α <prim>
6	<prim> → (<term>)	o <term>
7	<term> → <prim>	o <prim>
8	<term> → <prim> β <term> $\beta \in$ BIOP	o β <prim> <term>
9	<term>→if<term>then<prim>else<term>	o if <term> <prim> <term>

10	<assertion> → <ident> = <term>	
11	<assertion list>→<assertion>;<assertion list>	
12	<assertion list> → <assertion>	
13	<program> → <assertion list> ; i $i \in \mathbb{N}$	o <u>prog</u>,(EVAL,i,ε) <assertion list>

START		ERROR	L1
L1		L2	ERROR
L2		L2	L3
L3		L3	STOP

2.2. Semantics Description

The l.f.m.s. describing the semantics of Lucid programs is given in Table II. Of special interest in this description are a subset of the set of labels called control labels, and the set of basic functions. The subset of control labels is denoted CL and is defined as

$$CL = \left(\{EVAL, WAIT\} \times \mathbb{N}^* \times \mathbb{N}^*\right) \cup \left(\{VAL\} \times \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{D}\right)$$

where $\mathbb{N}^*$ denotes the set of all strings of non-negative integers including the empty string denoted by ε. The labels of this set control the evaluations in the program. A label of the form (EVAL, t, s), where $t, s \in \mathbb{N}^*$, can be interpreted as a request to evaluate the instance, indicated by t, of a certain term, while s is used as a stack of indices which is necessary for parallel computations. A label of the form (WAIT, t, s) is used only with variables to point out that the instance, indicated by t, is being evaluated. Finally, a label of the form (VAL, t, s, m), where $m \in \mathbb{Z} \cup \{T, F\}$, indicates that the value of the instance corresponding to t of a certain term is m.

The set of basic functions is given below. The function spef (for special functions) and feps (for reverse special functions) manipulate time parameter t and stack s according to the special operator involved. This special operator can be any one in the set SPOP = {first, next, latest, latest^{-1}}. The functions inc and dec manipulate the time parameter only. These functions are used for the semantics of the Lucid special operators. The functions uop (for unary operator) and bop (for binary operator) are used for the semantics of the arithmetic and logical operators. The set of binary arithmetic and logical operators is

$$\text{ALOP} = \{+, -, *, /, \uparrow, \vee, \wedge, \text{eq}, \text{ne}, \leq, <, >, \geq\}$$

The definition of these special functions is as follows.

$$\text{spef:}\quad \text{SPOP} \times \mathbb{N}^{+} \times \mathbb{N}^{*} \longrightarrow \mathbb{N}^{*} \times \mathbb{N}^{*}$$

$$(\sigma,\ t_0 t_1 \ldots t_n,\ s_0 \ldots s_m) \mapsto \begin{cases} (0 t_1 \ldots t_n,\ t_0 s_0 \ldots s_m) & \text{if } \sigma = \underline{\text{first}} \\ (t_0{+}1\ t_1 \ldots t_n,\ s_0 \ldots s_m) & \text{if } \sigma = \underline{\text{next}} \\ (t_1 \ldots t_n,\ t_0 s_0 \ldots s_n) & \text{if } \sigma = \underline{\text{latest}} \\ (0 t_0 \ldots t_n,\ s_0 \ldots s_m) & \text{if } \sigma = \underline{\text{latest}}^{-1} \end{cases}$$

$$\text{feps:}\quad \text{SPOP} \times \mathbb{N}^{*} \times \mathbb{N}^{*} \longrightarrow \mathbb{N}^{+} \times \mathbb{N}^{*}$$

$$(\sigma,\ t_0 t_1 \ldots t_n,\ s_0 s_1 \ldots s_m) \mapsto \begin{cases} (s_0 t_1 \ldots t_n,\ s_1 \ldots s_m) & \text{if } \sigma = \underline{\text{first}} \\ (t_0{-}1\ t_1 \ldots t_n,\ s_0 \ldots s_m) & \text{if } \sigma = \underline{\text{next}} \\ (s_0 t_0 \ldots t_n,\ s_1 \ldots s_m) & \text{if } \sigma = \underline{\text{latest}} \\ (t_1 \ldots t_n,\ s_0 s_1 \ldots s_m) & \text{if } \sigma = \underline{\text{latest}}^{-1} \end{cases}$$

$$\text{inc:}\quad \mathbb{N}^{+} \rightarrow \mathbb{N}^{+}$$

$$t_0 t_1 \ldots t_n \mapsto t_0{+}1\ t_1 \ldots t_n$$

$$\text{dec:}\quad \mathbb{N}^{+} \rightarrow \mathbb{N}^{+}$$

$$t_0 t_1 \ldots t_n \mapsto t_0{-}1\ t_1 \ldots t_n$$

$$\text{uop}: \{\neg, -\} \times \mathbb{D} \longrightarrow \mathbb{D}$$

$$(-, n) \longmapsto -n \quad \text{if} \quad n \in \mathbb{N}$$

$$(\neg, n) \longmapsto \neg n \quad \text{if} \quad n \in \{T, F\}$$

$$\text{bop}: \text{ALOP} \times \mathbb{D} \times \mathbb{D} \longrightarrow \mathbb{D}$$

$$(\rho, n, m) \longmapsto n\rho m$$

Note that functions uop and bop are partial functions. They can be changed into total functions by adding the special element Λ to the domain and letting

$$\text{uop } (\eta, n) = \Lambda \quad \text{if} \quad \eta n \text{ is not defined}$$

$$\text{bop } (\rho, n, m) = \Lambda \quad \text{if} \quad n\rho m \text{ is not defined.}$$

Intuitively Λ indicates an error like division by 0 or wrong type in a term.

The label parameters and their domains are listed below.

$\sigma \in \text{SPOP}$

$t \in \mathbb{N}^*$, $t' \in \{0\} \times \mathbb{N}^*$, $t'' \in (\mathbb{N}-\{0\}) \times \mathbb{N}^*$

$s \in \mathbb{N}^*$

$\eta \in \{\neg, -\}$

$\rho \in \{+, -, *, /, \uparrow, \vee, \wedge, \text{eq}, \text{ne}, \leq, <, >, \geq\}$

$\rho' \in \{+, -, *, /, \uparrow, \text{eq}, \text{ne}, \leq, <, >, \geq\}$

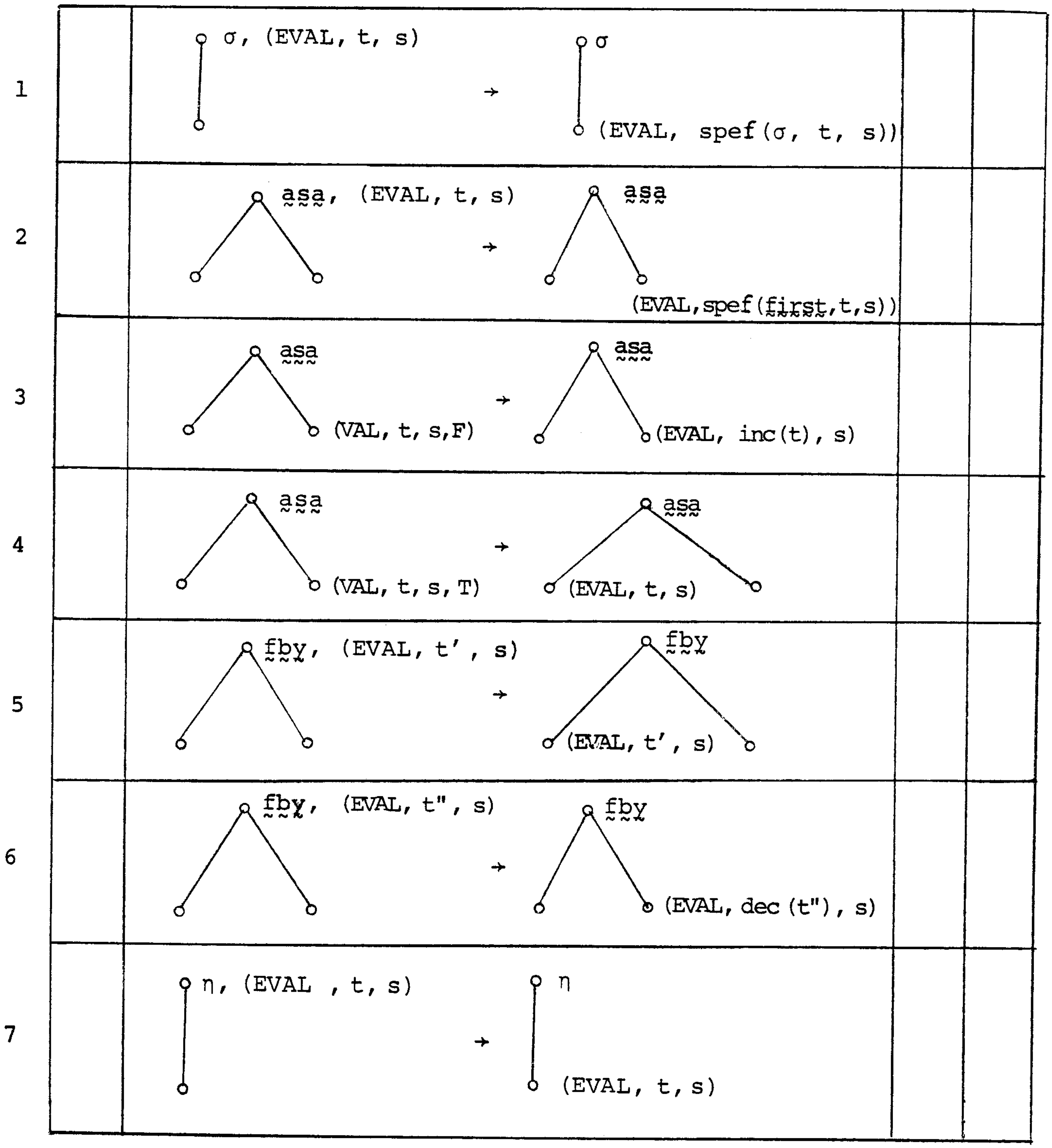
$m, n \in \mathbb{D}$

$i \in \mathbb{N}$

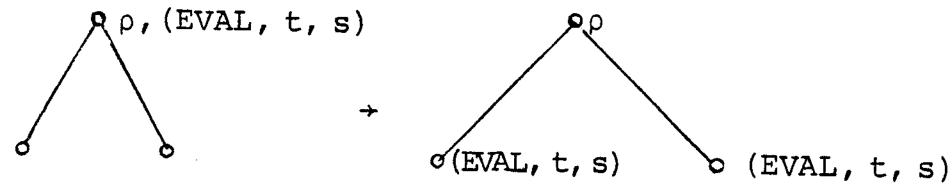
There are no tree parameters since the productions are essentially structure preserving.

The productions of the l.f.m.s. describe the semantics of Lucid in a computational way. The goal of the computations is the value of $OUTPUT_i$ in a given program. Production 29 starts the evaluation of $OUTPUT_i$ by requesting its value. When this value is obtained the computations come to a halt as indicated by production 30. Productions 1 through 28 define the evaluation of any possible term. Since all these productions are labelled by the same blank label, as well as their success and failure fields, the computations are essentially non-deterministic. Moreover the evaluations are performed in parallel as implied by production 8 and 17-23. This parallelism is essential for the operators $\wedge$ and $\vee$ but not so for all the other arithmetic and logical operators.

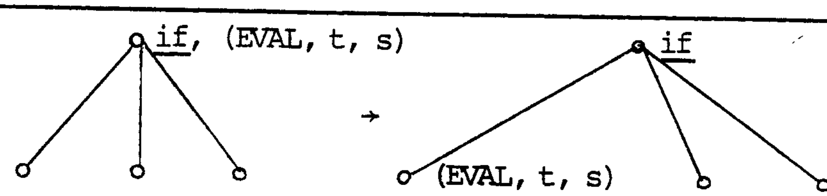
Table II — Semantics of Lucid



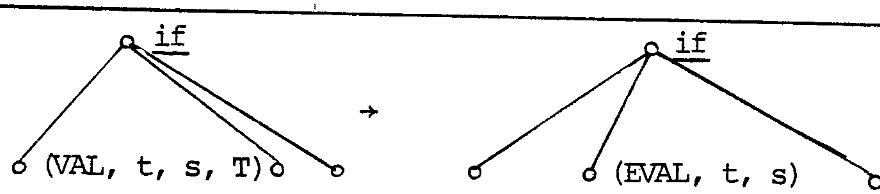
8



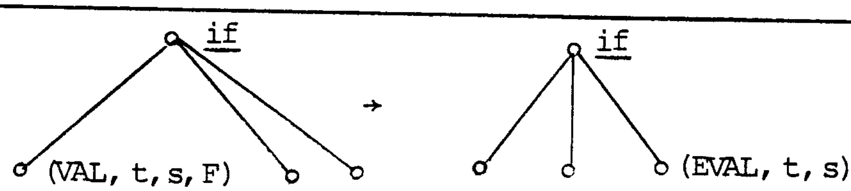
9



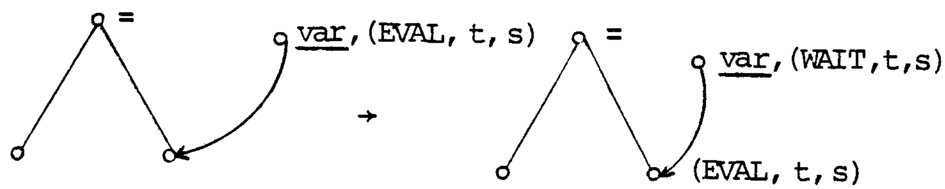
10



11



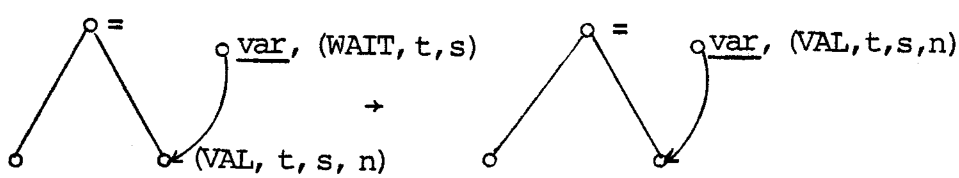
12



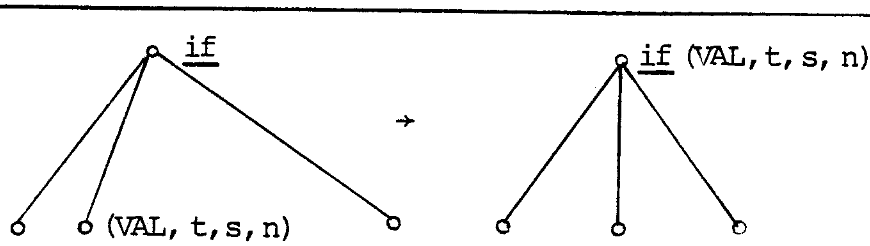
13

o const, n, (EVAL, t, s) → o const, n, (VAL, t, s, n)

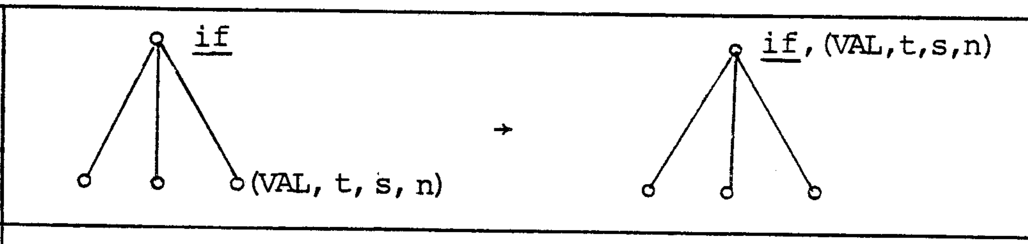
14



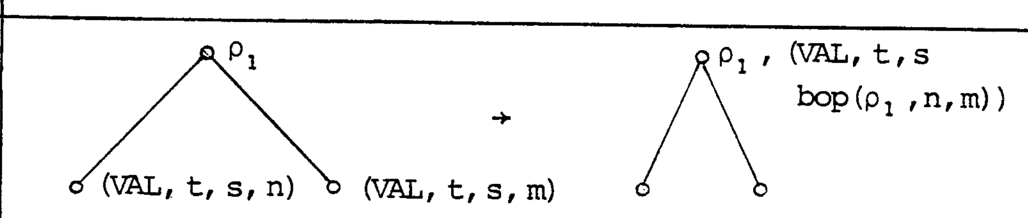
15



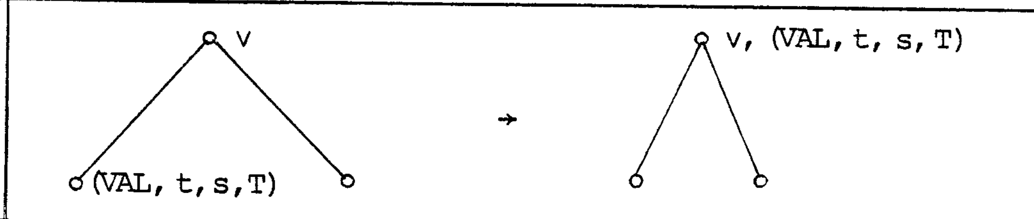
16



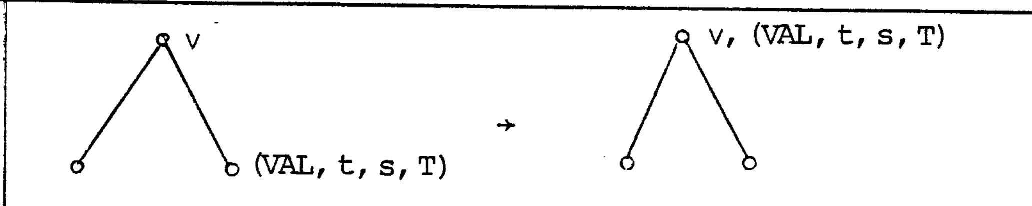
17



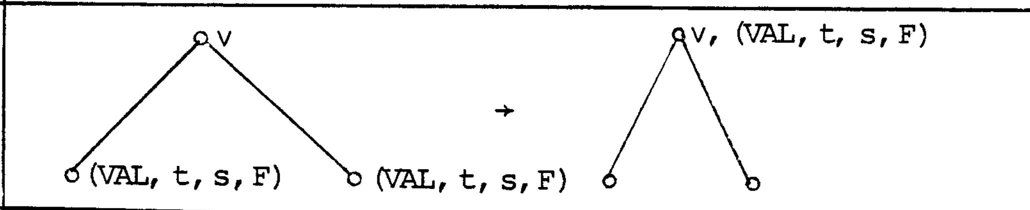
18



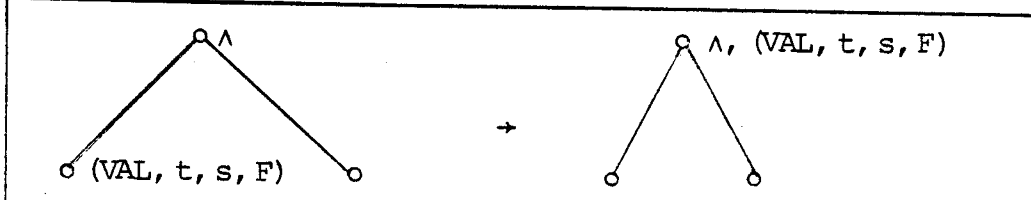
19



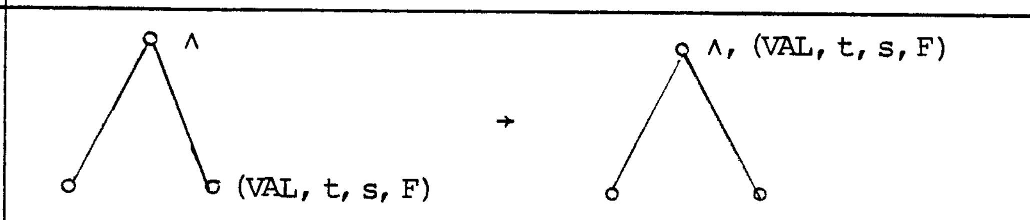
20



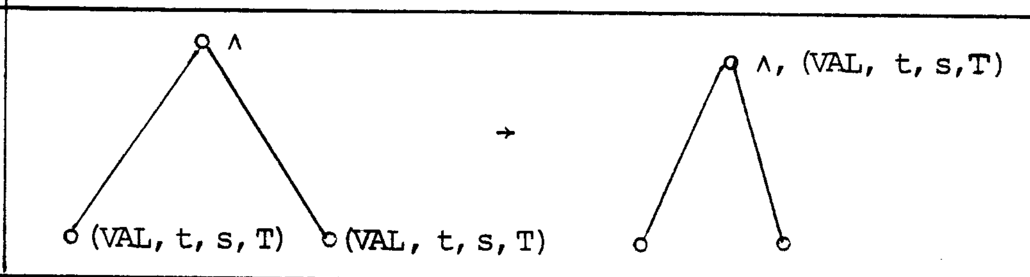
21

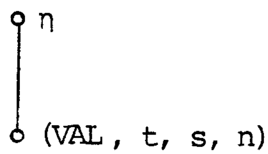
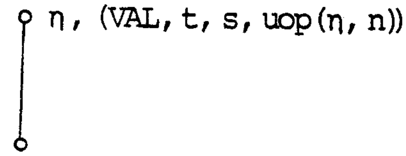
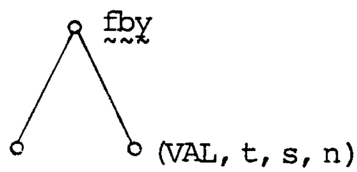
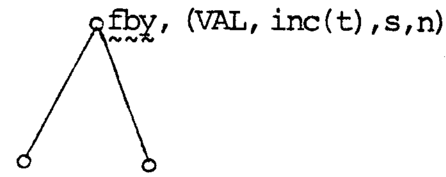
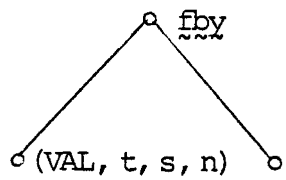
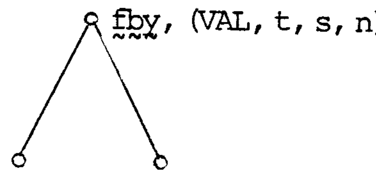
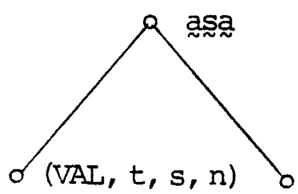
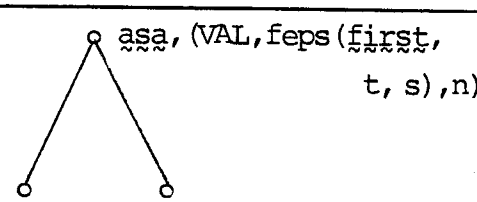
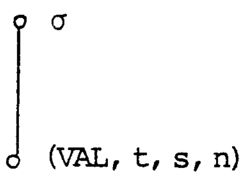
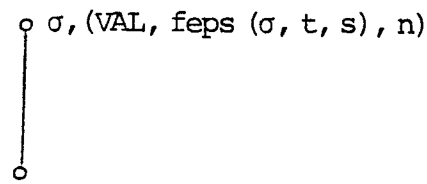
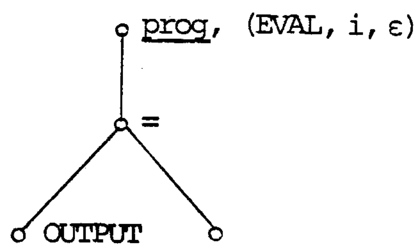
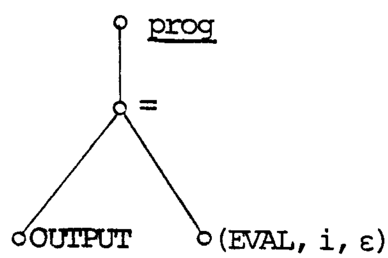
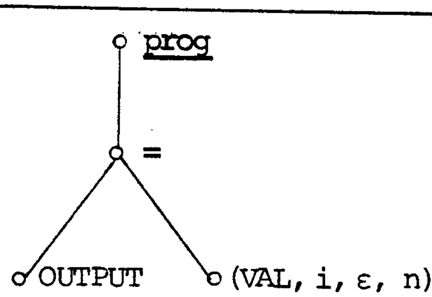
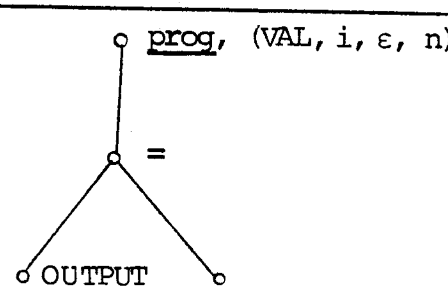


22



23



24			→			
25			→			
26			→			
27			→			
28			→			
29	START		→			
30			→		STOP	

3. Brief overview of the denotational semantics of Lucid

The complete semantics for Lucid can be found in [1], but we will give a short review of the main points. The domain of values is augmented by adding the undefined element denoted $\perp$, and a flat complete partial order is defined on this domain. Thus $\perp \sqsubseteq x$ for any x of the domain while any two elements which are different from $\perp$ do not compare in this relation.

A program is written as $\overline{X} = \tau(\overline{X})$ where $\overline{X} = \langle x_1, \ldots, x_v \rangle$ and $\tau(\overline{X}) = \langle \tau_1(\overline{X}), \ldots, \tau_v(\overline{X}) \rangle$. The operators which may be conbined to form a functional τ_i are the constant functions, the arithmetic and logical operators as well as the Lucid special operators mentioned in the previous section. One additional class of operators consists of the projection functions denoted p_j for $j = 1, \ldots, v$. They are defined by $p_j(\overline{X}) = x_j$; and clearly are not needed when using the infix notation for the operators as in the previous section.

The semantics of the special Lucid operators is as follows. Let t be an infinite sequence of non-negative integers i.e. $t = t_0 t_1 \ldots$ and let x and y be elements of the domain.

$$(\text{first } x)_{t_0 t_1 \ldots} = x_{0 t_1 \ldots}$$

$$(\text{next } x)_{t_0 t_1 \ldots} = x_{t_0+1\ t_1 \ldots}$$

$$(x \text{ fby } y)_{t_0 t_1 \ldots} = \begin{cases} x_{0 t_1 \ldots} & \text{if } t_0 = 0 \\ y_{t_0-1\ t_1 \ldots} & \text{otherwise} \end{cases}$$

$$(\underset{\sim}{\text{latest}}\ x)_{t_0 t_1 \ldots} = x_{t_1 \ldots}$$

$$(\underset{\sim}{\text{latest}}^{-1}\ x)_{t_0 t_1 \ldots} = x_{0 t_0 t_1 \ldots}$$

$$(x\ \underset{\sim}{\text{asa}}\ y)_{t_0 t_1 \ldots} = \begin{cases} x_{s t_1 \ldots} & \text{if } \exists\ s:\ \forall\ r < s,\ y_{r t_1 \ldots} = F \\ & \text{and } y_{s t_1 \ldots} = T \\ \bot & \text{otherwise} \end{cases}$$

The arithmetic and logical unary, binary and trinary operators are pointwise operators with respect to the time parameter t, e.g. $(x + y)_t = x_t + y_t$.

The semantics of a program P is defined as being the minimal solution for P which is shown to exist because all the operators are continuous. Moreover, it can be computed as the upper bound of $\tau^i(\bar{\bot})$, where $\bar{\bot} = <\bot, \ldots, \bot>$ and τ^i denotes the composition of τ with itself i times. As it is shown in [7] this upper bound is in fact the limit of the sequence

$$\tau^0(\bar{\bot}) = \bar{\bot},\quad \tau(\bar{\bot}),\quad \tau^2(\bar{\bot}), \ldots$$

because that sequence is increasing, i.e.

$$\tau^i(\bar{\bot}) \sqsubseteq \tau^{i+1}(\bar{\bot}) \quad \text{for all} \quad i \in N.$$

Therefore $\exists j: (\tau^j(\bar{\bot}))_t = m \neq \bot$ iff $(\bigsqcup_i \tau^i(\bar{\bot}))_t = m$.

<u>Notation</u>: For any $t \in \mathbb{N}^*$ we write $x_t = m$ to mean that for any infinite sequence of non-negative integers t' we have $x_{tt'} = m$.

4. How to express certain properties of l.f.m.s.

Since we will be dealing with programs and their linked tree representations we need a notation which expresses the relation between the two representations. We will denote by [P] the tree representation of program P as produced by the syntax description of Lucid given in Table I. Moreover, for any term σ (or more formally $\sigma(\overline{X})$) we denote by $[\sigma]$ the tree (without pointers) corresponding to σ in the mapping defined by the syntax description.

Let CL denote the set of control labels in the semantics description. Two linked trees e_1 and e_2 are almost identical, written $e_1 \simeq e_2$, if the removal of all control labels from both trees makes them isomorphic.

Definition 4.1. Let e and f be such that $e \Longrightarrow f$ (i.e. f derives from e by some production) and suppose that label $\ell \in CL$ is at node x in e. We say that (x, ℓ) is essential for $e \Longrightarrow f$ if the tree g obtained from e by removing label ℓ from node x is such that $f \not\Longrightarrow g$ (i.e. there is no production such that $f \Longrightarrow g$). $\square$

For any linked tree e we denote by $A(e)$ the set of pairs (x, ℓ) such that x is a node in e and ℓ is a label at node x in e.

Definition 4.2. Let e and f be linked trees such that $e = e^0 \xrightarrow{p_1} e^1 \xrightarrow{p_2} \cdots \xrightarrow{p_k} e^k = f$ and $e \simeq f$. Also let

$B_1 \subseteq A(e)$ and $B_2 \subseteq A(f)$. We say that B_1 <u>produces</u> B_2 in the above derivation if there are linked trees $f^0 \simeq e^0$, $f^1 \simeq e^1, \ldots f^k \simeq e^k$ such that

$$f^0 \xRightarrow{p_1} f^1 \xRightarrow{p_2} \cdots \xRightarrow{p_k} f^k$$

with $\quad A(f^0) = B_1$

$\qquad A(f^i) \subseteq A(e^i) \quad$ for $\quad 1 \le i \le k$

and $\quad B_2 \subseteq A(f^k)$. $\square$

Intuitively, if B_1 produces B_2 and $(x, \ell) \in B_2$ then control label ℓ at node x in f is generated in the derivation by the control labels of e which are at the nodes indicated by the pairs (node, label) of B_1, and only by these control labels.

The notion of transformation on subtrees in a given derivation is needed to be able to express different properties of terms in a certain evaluation. The following definition makes this notion precise.

<u>Definition 4.3</u>. Let E and F be linked trees such that $E \simeq F$. Also let e be a subtree of E, and f a subtree of F such that $e \simeq f$. For any $\ell_1, \ell_2 \in CL$ we say that (ℓ_1, e) <u>is transformed into</u> (ℓ_2, f) and write

$$\overset{o\ell_1}{e} \models^* \overset{o\ell_2}{f}$$

if $\quad E = E^0 \xRightarrow{p_1} E^1 \xRightarrow{p_2} \cdots \xRightarrow{p_k} E^k = F$

and relatively to the above derivation the following two conditions are satisfied:

(i) if r is the root of e in E and s the root of f in F then $\{(r, \ell_1)\}$ produces $\{(s, \ell_2)\}$.

(ii) for all i and any pair $(x, \ell) \in A(E^i)$ which is essential for $E^i \Longrightarrow E^{i+1}$,

$$\{(r, \ell_1)\} \text{ produces } \{(x, \ell)\}. \qquad \square$$

5. Partial correctness of the Lucid interpreter

In this section we will show that whenever an instance σ_t of term σ is defined in the minimal solution of a program P and its value is m, then the evaluation of the term σ (in tree form) by the interpreter leads to the same value m. This is stated more formally as follows.

Theorem 5.1.

For any term $\sigma(\bar{X})$ in P, e such that $e \simeq [\sigma(\bar{X})]$, and any $t, s \in \mathbb{N}^*$, if

$$\exists i, \ (\sigma(\tau^i(\bar{I})))_t = m \quad \text{and} \quad m \neq \bot$$

then

$$\underset{e}{o}(\text{EVAL}, t, s) \models^* \underset{e'}{o}(\text{VAL}, t, s, m)$$

for some $e' \simeq [\sigma(X)]$. $\square$

Before proving this theorem we will prove a lemma which simplifies the proof of the theorem. Also we say that term $\sigma(\overline{X})$, or simply σ, has property $\pi_1(k)$ if it satisfies the property of Theorem 5.1 with $i = k$.

Lemma 5.1.

If σ_1, σ_2 and σ_3 are terms having property $\pi_1(i)$ then so is any term σ formed from one or more σ_j, $(j = 1, 2, 3)$ using any Lucid special function or arithmetic or logical operator.

Proof.

Several cases have to be considered.

(a) Let $f \in \text{SPEC}' = \{\text{first}, \text{next}, \text{latest}, \text{latest}^{-1}\}$, and consider $f\sigma_1$. Suppose that for some $t, (f\,\sigma_1\,\tau^i(\bar{I}))_t = m$ and $m \neq \perp$. If $(t', s') = \text{spef}(f, t, s)$ it is easy to check from the definitions of f and spef that $(f\,\sigma_1\tau^i(\bar{I}))_t = (\sigma_1\,\tau^i(\bar{I}))_{t'}$.

Also $[f\,\sigma_1] = \begin{array}{c} \circ\; f \\ | \\ \circ \\ [\sigma_1] \end{array}$, and

if $e_1, e_1' \simeq [\sigma_1]$ then for any s,

$$\begin{array}{c} \circ\; f, (\text{EVAL}, t, s) \\ | \\ \circ \\ e_1 \end{array} \overset{1}{\Longrightarrow} \begin{array}{l} \circ\; f \\ | \\ \circ\; (\text{EVAL, spef}\,(f, t, s)) \\ e_1 \end{array}$$

$$(\text{by hypothesis on } \sigma_1) \models\!\!=^{*} \begin{array}{l} \circ\; f \\ | \\ \circ\; (\text{VAL, spef}(f, t, s), m) \\ e_1' \end{array}$$

$$\overset{28}{\Longrightarrow}\quad \begin{array}{l} \circ\ f,\ (VAL,\ feps\,(f, spef(f,t,s)), m \\ | \\ \circ \\ e_1' \end{array}$$

However, $feps(f, spef(f, t, s)) = (t, s)$ for any $f \in SPOP$.

Thus for any $e \simeq [f\sigma_1]$,

$$\underset{e}{\circ}\ (EVAL,\ t,\ s) \quad \models\!\!=\!\!=^{*} \quad \underset{e'}{\circ}\ (VAL,\ t,\ s,\ m)$$

for some $e' \simeq [f\sigma_1]$.

(b) Consider σ_1 asa σ_2 and suppose that for some t, $t = t_0 t_1 \dots t_k$, $((\sigma_1 \text{ asa } \sigma_2)\ \tau^i(\bar{I}))_t = m$ and $m \neq \bot$.

From the definition of asa follows that there exists t_0' such that for $t' = t_0' t_1 \dots t_n$, $(\sigma_2)_{t'} = T$ and for all $t_0'' \leq t_0' - 1$, $(\sigma_2)_{t_0'' t_1 \dots t_k} = F$. Also $((\sigma_1 \text{ asa } \sigma_2)\ \tau^i(\bar{I}))_t = (\sigma_1\ \tau^i(\bar{I}))_{t'}$.

On the other hand

$$[\sigma_1 \text{ asa } \sigma_2] = \begin{array}{ccc} & \circ\ \text{asa} & \\ / & & \backslash \\ \circ & & \circ \\ [\sigma_1] & & [\sigma_2] \end{array}$$

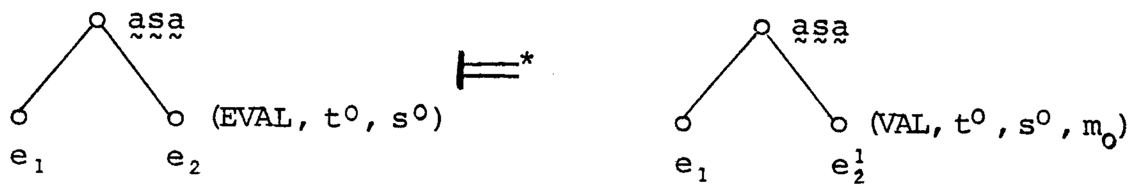
So, if $e_1 \simeq [\sigma_1]$ and $e_2 \simeq [\sigma_2]$, then

$$\begin{array}{ccc} & \circ\ \text{asa},\ (EVAL,\ t,\ s) & \\ / & & \backslash \\ \circ & & \circ \\ e_1 & & e_2 \end{array} \quad \overset{2}{\Longrightarrow} \quad \begin{array}{ccc} & \circ\ \text{asa} & \\ / & & \backslash \\ \circ & & \circ\ (EVAL,\ t^0,\ s^0) \\ e_1 & & e_2 \end{array}$$

where $(t^0, s^0) = spef\ (\text{first},\ t,\ s)$

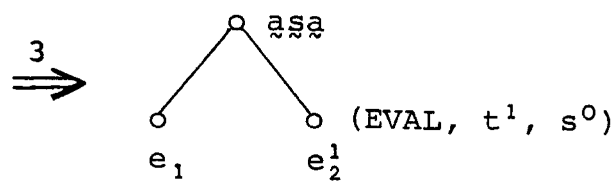
$$= (0t_1 \dots t_k,\ t_0 s_0 \dots s_j).$$

But $(\sigma_2\ \tau^i(\bar{I}))_{0t_1 \ldots t_k}$ is either F or T, i.e. different from $\perp$. Using the hypothesis on σ_2, we have that

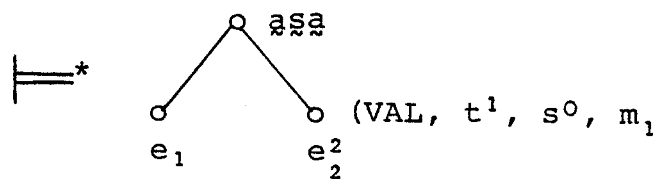


where $m_0 = (\sigma_2\ \tau^i(\bar{I}))_{0t_1 \ldots t_k}$.

If $m_0 = F$ then

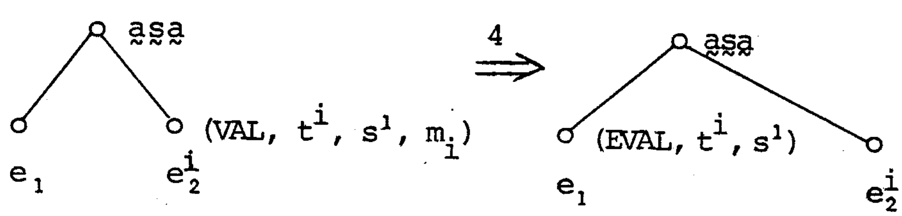


where $t^1 = 1t_1 \ldots t_k$. Also we have that



where $m_1 = (\sigma_2\ \tau^i(\bar{I}))_{1t_1 \ldots t_k}$.

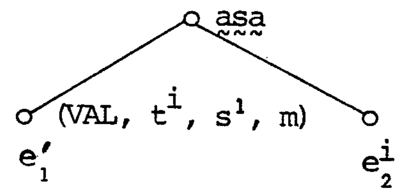
Thus, it can easily be shown that production 3 would have to be used t_0' times, because $m_0, \ldots m_{t_0'-1}$ are all equal to F, and $m_{t_0'} = T$. Let $i = t_0'$, we would then have



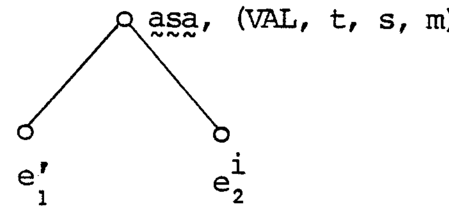
By hypothesis on σ_1, and since $(\sigma_1\ \tau^i(\bar{I}))_{it_1 \ldots t_n} = m$

and $feps(\text{first}, it_1 \ldots t_k\ ,\ \ t_0 s_0 \ldots s_j) = (t_0 t_1 \ldots t_k\ , s_0 \ldots s_j)$

$\models^*$



$\overset{27}{\Longrightarrow}$



Consequently, for any $e \simeq [\sigma_1\ \text{asa}\ \sigma_2]$,

$$\underset{e}{\circ}\ (\text{EVAL}, t, s) \models^* \underset{e'}{\circ}\ (\text{VAL}, t, s, m)$$

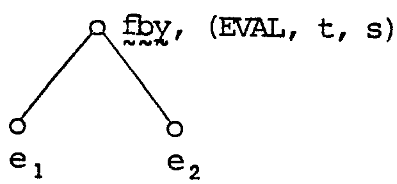
(c) Consider $\sigma_1\ \text{fby}\ \sigma_2$ and suppose that for some $t = t_0 t_1 \ldots t_k$, $((\sigma_1\ \text{fby}\ \sigma_2)\ \tau^i(\bar{I}))_t = m$ and $m \neq \bot$.

Two cases are to be considered.

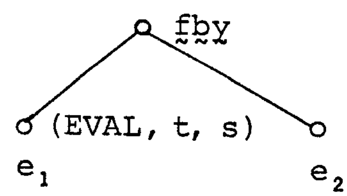
- If $t_0 = 0$ then by definition of fby

$$((\sigma_1\ \text{fby}\ \sigma_2)\ \tau^i(\bar{I}))_t = (\sigma_1\ \tau^i(\bar{I}))_t$$

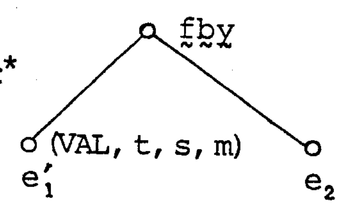
In this case, for any $e_1 \simeq [\sigma_1]$ and $e_2 \simeq [\sigma_2]$ we have,

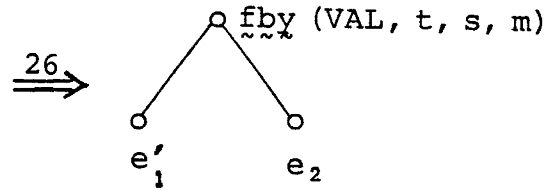


$\overset{5}{\Longrightarrow}$



(by hypothesis on σ_1) $\models^*$



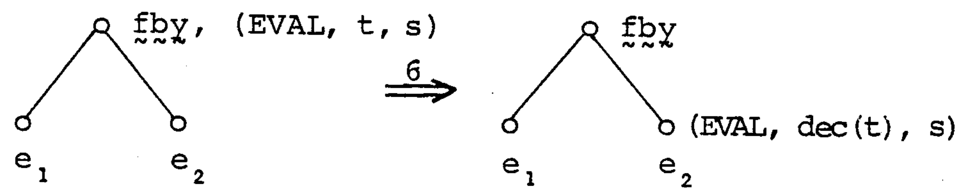


- If $t_0 \neq 0$ then by definition of fby

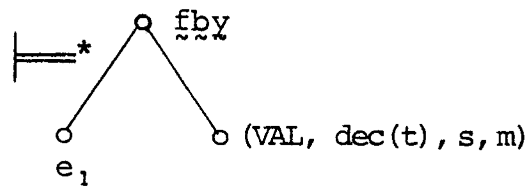
$$((\sigma_1 \text{ fby } \sigma_2)\tau^i(\bar{I}))_{t_0 t_1 \ldots t_k} = (\sigma_2\ \tau^i(\bar{I}))_{t_0 - 1\, t_1 \ldots t_k}$$

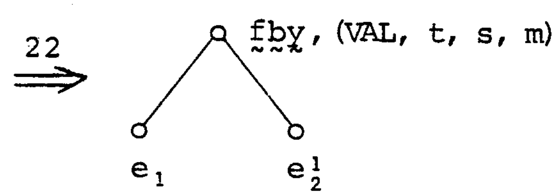
$$= (\sigma_2\ \tau^i(\bar{I}))_{\mathrm{dec}(t)}$$

In this case, for any $e_1 \simeq [\sigma_1]$ and $e_2 \simeq [\sigma_2]$, since $\mathrm{inc}(\mathrm{dec}(t)) = t$, we have



(by hypothesis on σ_2)





Thus in both cases for any $e \simeq [\sigma_1 \text{ fby } \sigma_2]$

$$\underset{e}{\circ}\ (\text{EVAL}, t, s) \models^* \underset{e'}{\circ}\ (\text{VAL}, t, s, m)$$

(d) Consider $g\,\sigma_1$ where $g \in \{-, \neg\}$.

Suppose that for some t, $(g\,\sigma_1\ \tau^i(\bar{I}))_t = m$ and $m \neq \bot$.

Then $(g\,\sigma_1\,\tau^i(\bar{I}))_t = m$, i.e. $(\sigma_1\;\tau^i(\bar{I}))_t = m_1 \;(\neq \perp)$

and $g(m_1) = m$.

But $[g\,\sigma_1] = \begin{array}{c} \circ\; g \\ | \\ \circ \\ [\sigma_1] \end{array}$.

So, for any $e_1 \simeq [\sigma_1]$ we have

$$\begin{array}{l} \circ\; g, (\mathrm{EVAL},\, t,\, s) \\ | \\ \circ \\ e_1 \end{array} \quad \overset{7}{\Longrightarrow} \quad \begin{array}{l} \circ\; g \\ | \\ \circ \quad (\mathrm{EVAL},\, t,\, s) \\ e_1 \end{array}$$

(by hypothesis on σ_1)

$$\models^{*} \quad \begin{array}{l} \circ\; g \\ | \\ \circ\; (\mathrm{VAL},\, t,\, s,\, m_1) \\ e_1' \end{array}$$

$$\overset{24}{\Longrightarrow} \quad \begin{array}{l} \circ\; g, (\mathrm{VAL},\, t,\, s,\, \mathrm{uop}(g,\, m_1)) \\ | \\ \circ \\ e_1' \end{array}$$

By definition of uop, $\mathrm{uop}(g,\, m_1) = g(m_1)$.

Thus for any $e \simeq [g\,\sigma_1]$, there exists $e' \simeq [g\,\sigma_1]$

such that: $\underset{e}{\circ}\; (\mathrm{EVAL},\, t,\, s) \;\models^{*}\; \underset{e'}{\circ}\; (\mathrm{VAL},\, t,\, s,\, m)$

(e) Consider $\sigma_1\; h\; \sigma_2$ where $h \in \mathrm{ALOP} - \{\vee, \wedge\}$.

Suppose that for some $t, ((\sigma_1\; h\; \sigma_2)\; \tau^i(\bar{I}))_t = m$

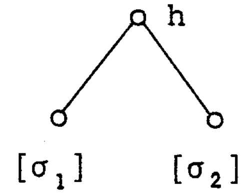
and $m \neq \perp$. It follows that

$$(\sigma_1\;\tau^i(\bar{I}))_t\; h\; (\sigma_2\,\tau^i(\bar{I}))_t = m \quad \text{and} \quad m \neq \perp$$

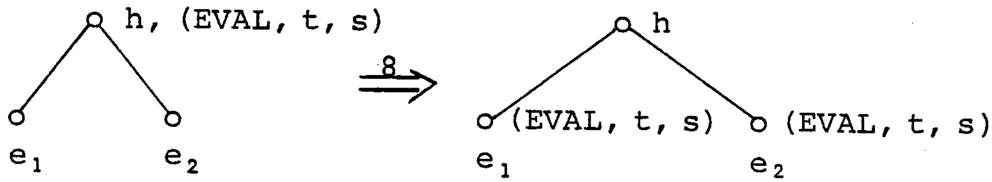
and

$$(\sigma_1\;\tau^i(\bar{I}))_t = m_1 \neq \perp, \quad (\sigma_2\,\tau^i(\bar{I}))_t = m_2,\; m_1\, h\, m_2 = m.$$

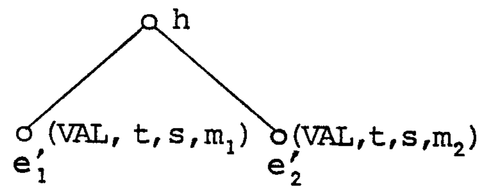
Since $\sigma_1 \; h \; \sigma_2 =$

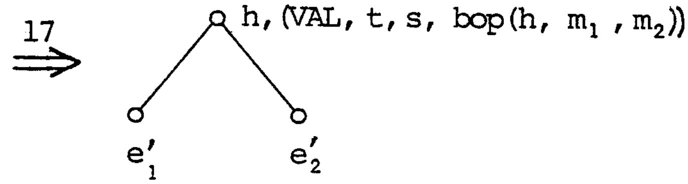


for any $e_1 \simeq [\sigma_1]$, $e_2 \simeq [\sigma_2]$ we have



(by hypothesis on σ_1 and σ_2) $\models^*$





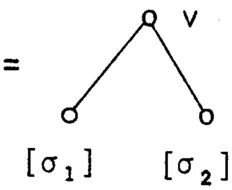
However, $bop(h, m_1 , m_2) = m_1 \; h \; m_2$ by definition of the basic function bop.

Thus, for any $e \simeq [\sigma_1 \; h \; \sigma_2]$

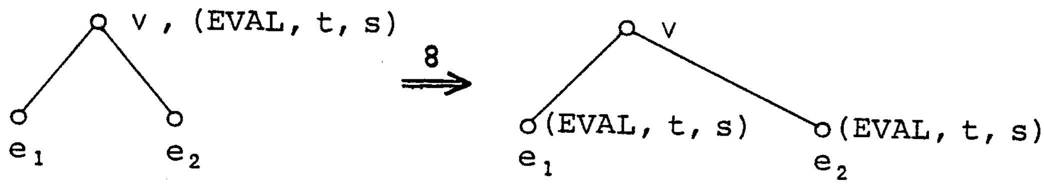
$$\underset{e}{o}\,(EVAL, t, s) \models^* \underset{e'}{o}\,(VAL, t, s, m)$$

for some $e' \simeq [\sigma_1 \; h \; \sigma_2]$.

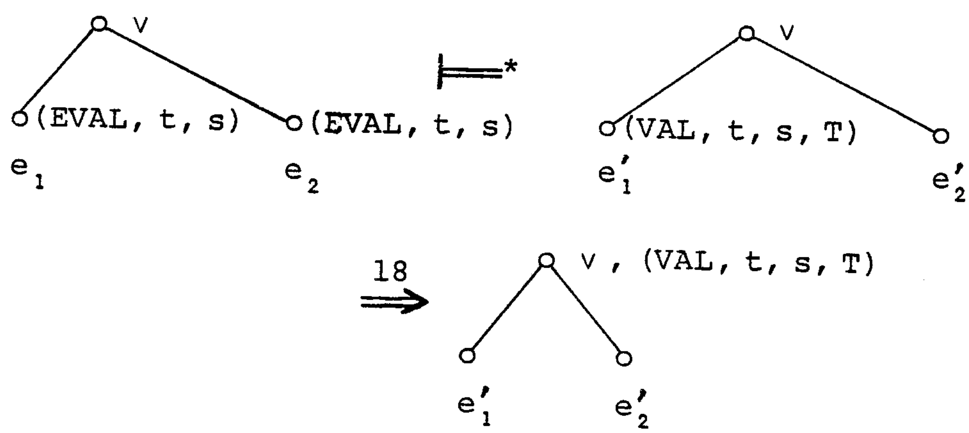
(f) Consider $\sigma_1 \vee \sigma_2$, and suppose that for some t, $((\sigma_1 \vee \sigma_2) \; \tau^i(\bar{\bot}))_t = m$ and $m \neq \bot$. This means that $(\sigma_1 \; \tau^i(\bar{\bot}))_t \vee (\sigma_2 \; \tau^i(\bar{\bot}))_t = m$ and $m \neq \bot$. By definition of the $\vee$ operator if $m = T$ then at least one of $(\sigma_1 \; \tau^i(\bar{\bot}))_t$ and $(\sigma_2 \; \tau^i(\bar{\bot}))_t$ should be T, but if $m = F$ then both $(\sigma_1 \; \tau^i(\bar{I}))_t$ and $(\sigma_2 \; \tau^i(\bar{I}))_t$ should be equal to F.

Since $[\sigma_1 \vee \sigma_2] =$ 

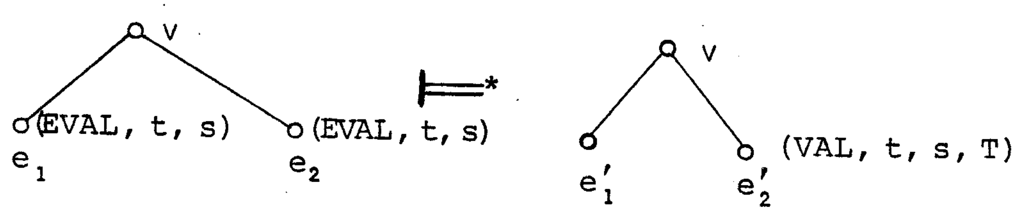
, for any $e_1 \simeq [\sigma_1]$ and $e_2 \simeq [\sigma_2]$, we have

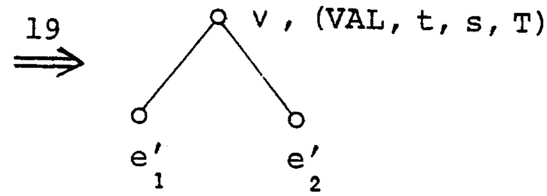


- If $m = T$ and $(\sigma_1 \; \tau^i(\bar{I}))_t = T$ then by hypothesis on σ_1,

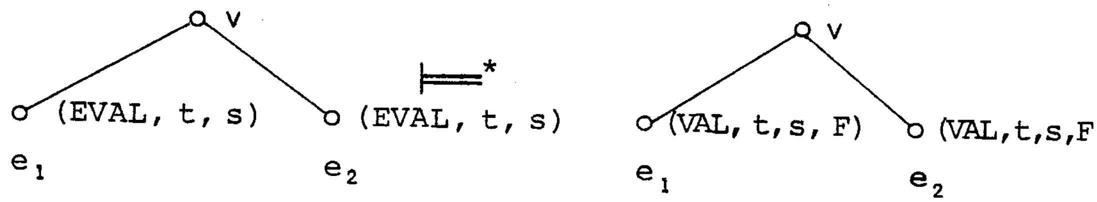


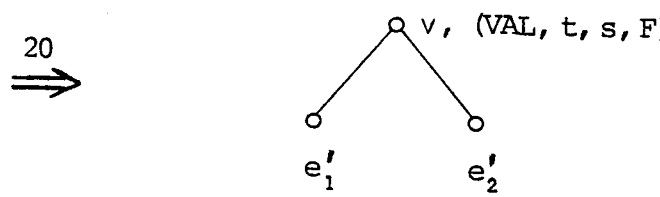
- If $m = T$ and $(\sigma_2 \; \tau^i(\bar{I}))_t = T$ then by hypothesis on σ_2,





- If $m = F$ then by hypothesis on σ_1 and σ_2





Thus in all possible cases and for any $e \simeq [\sigma_1 \vee \sigma_2]$

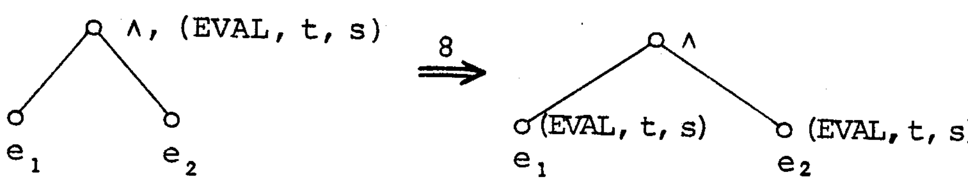
$$\underset{e}{\circ}\,(\mathrm{EVAL}, t, s) \quad \models^{*} \quad \underset{e'}{\circ}\,(\mathrm{VAL}, t, s, m)$$

for some $e' \simeq [\sigma_1 \vee \sigma_2]$.

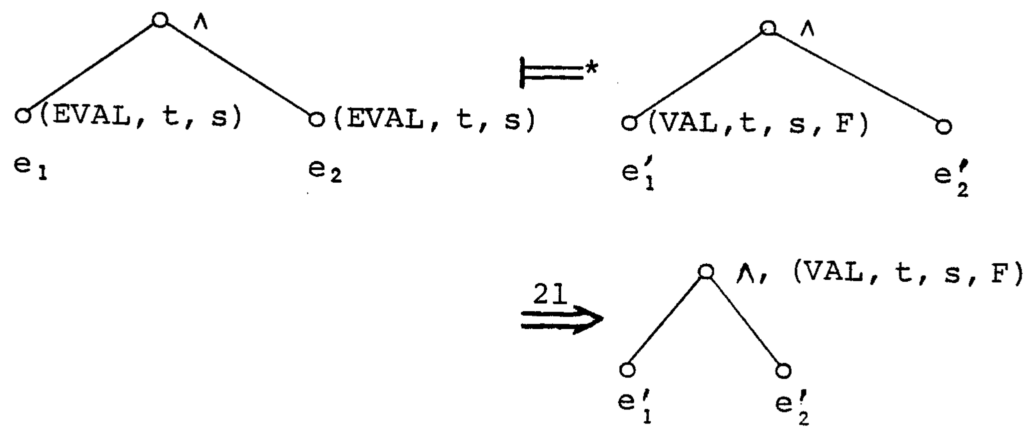
(g) Consider $\sigma_1 \wedge \sigma_2$.

Suppose that for some t, $((\sigma_1 \wedge \sigma_2)\ \tau^i(\bar{I}))_t = m$ and $m \neq \perp$. Then $(\sigma_1\ \tau^i(\bar{I}))_t \wedge (\sigma_2\ \tau^i(\bar{I}))_t = m$ and $m \neq \perp$. By the definition of $\wedge$, if $m = F$ then at least one of $(\sigma_1\ \tau^i(\bar{I}))_t$ and $(\sigma_2\ \tau^i(\bar{I}))_t$ should be equal to F , but if $m = T$ then both should be equal to T .

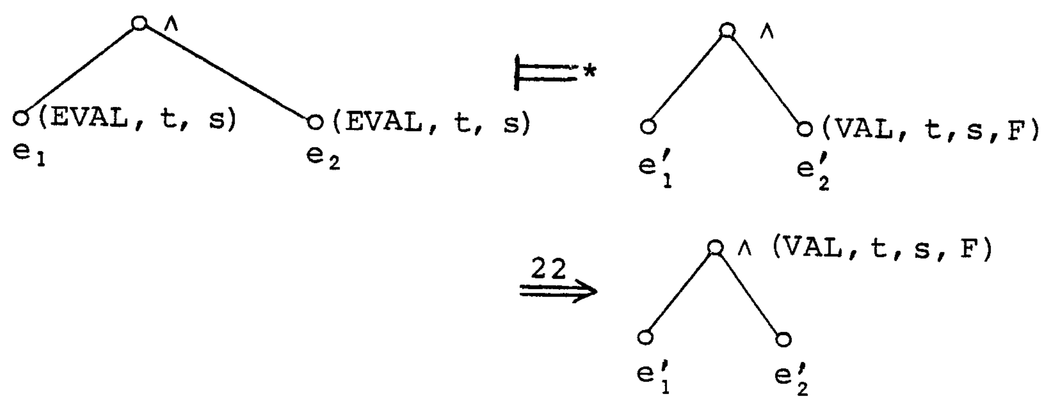
Let $e_1 \simeq [\sigma_1]$ and $e_2 \simeq [\sigma_2]$, then



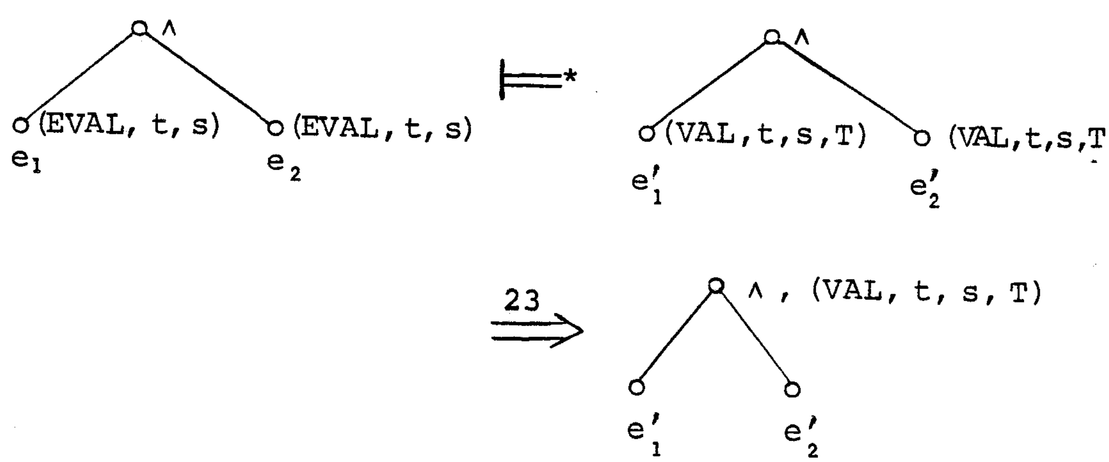
- If $m = F$ and $(\sigma_1\tau^i(\bar{I}))_t = F$ then by hypothesis on σ_1



- If $m = F$ and $(\sigma_2\tau^i(\bar{I}))_t = F$ then by hypothesis on σ_2



- If $m = T$ then by hypothesis on σ_1 and σ_2



In all possible cases for any $e \simeq [\sigma_1 \wedge \sigma_2]$ and for some $e' \simeq [\sigma_1 \wedge \sigma_2]$

$$\begin{matrix} o(\mathrm{EVAL},t,s) \\ e \end{matrix} \models^{*} \begin{matrix} o\ (\mathrm{VAL},t,s,m) \\ e' \end{matrix}$$

(h) Consider $\underline{\text{if}}\ \sigma_1\ \underline{\text{then}}\ \sigma_2\ \underline{\text{else}}\ \sigma_3$ and suppose that for some t, $((\underline{\text{if}}\ \sigma_1\ \underline{\text{then}}\ \sigma_2\ \underline{\text{else}}\ \sigma_3)\ \tau^i(\bar{I}))_t = m$ and $m \neq \bot$.

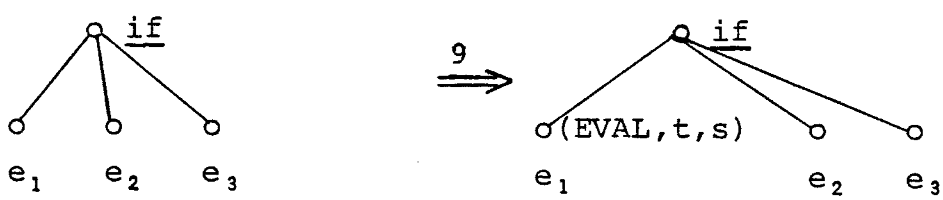
By the definition of the if-then-else operator,

$$(\sigma_1\ \tau^i(\bar{I}))_t = m_1 \quad \text{for some} \quad m_1 \in \{T, F\}\ ,$$

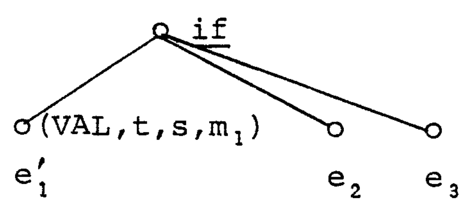
and if $m_1 = T$ then $(\sigma_2\ \tau^i(\bar{I}))_t = m$,

but if $m_1 = F$ then $(\sigma_3\ \tau^i(\bar{I}))_t = m$.

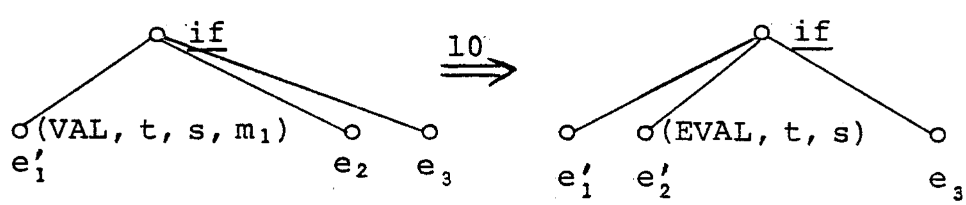
Let $e_1 \simeq [\sigma_1]$, $e_2 \simeq [\sigma_2]$ and $e_3 \simeq [\sigma_3]$.



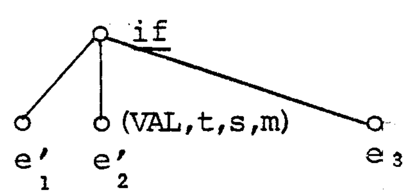
(by hypothesis on σ_1) $\models^{*}$

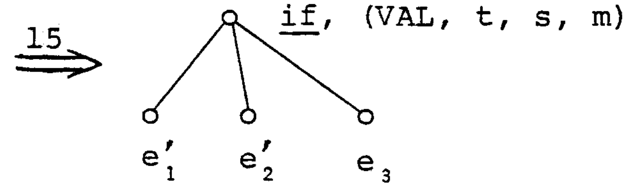


- if $m_1 = T$ then

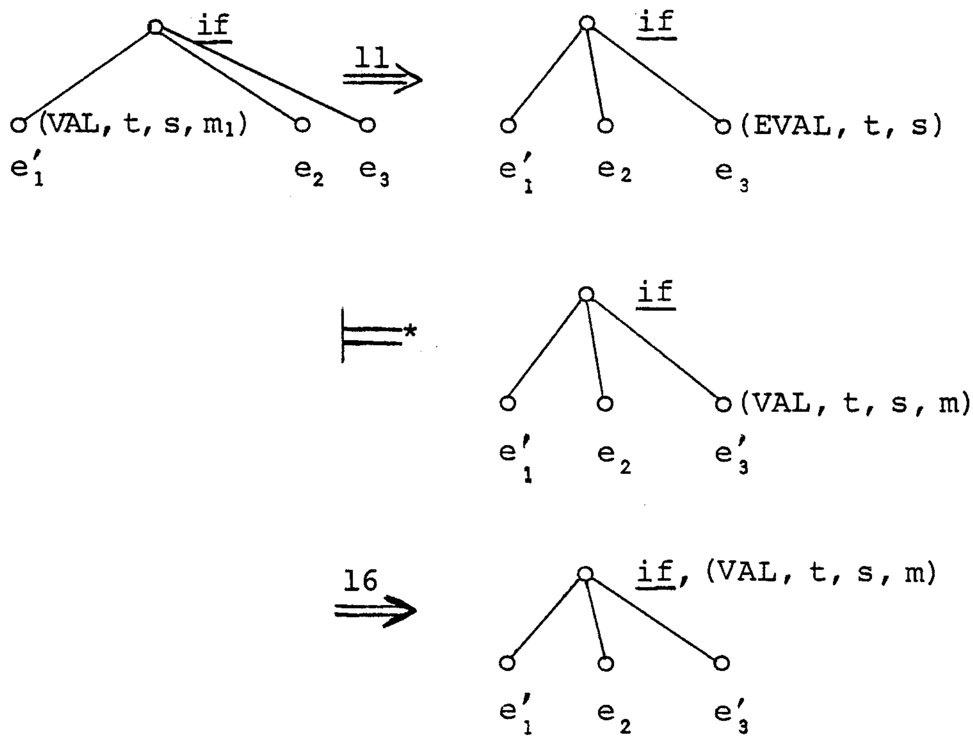


$\models^{*}$





- <u>if</u> $m_1 = F$ then



In both possible cases, for any $e \simeq$ [<u>if</u> σ_1 <u>then</u> σ_2 elseσ_3]

$e \simeq$ [<u>if</u> σ_1 <u>then</u> σ_2 <u>else</u> σ_3]

$$\underset{e}{\circ(\text{EVAL}, t, s)} \models^* \underset{e'}{\circ(\text{VAL}, t, s, m)}$$

for some $e' \simeq$ [<u>if</u> σ_1 <u>then</u> σ_2 <u>else</u> σ_3] .

Therefore, any possible term σ formed from one or more σ_i's (i = 1, 2, 3) using a Lucid special function or an arithmetic or logical operator, has property $\pi_1(i)$ provided that σ_1, σ_2 and σ_3 have property $\pi_1(i)$ for some i. $\square$

Now we can prove Theorem 5.1.

<u>Proof</u> (Theorem 5.1)

It will be done by induction on i.

<u>Base step</u> for $i = 0$, $(\sigma\, \tau^i(\bar{\bot}))_t = (\sigma(\bar{\bot}))_t$

Suppose that $(\sigma(\bar{\bot}))_t = m$ and $m \neq \bot$ for some t.

Let us perform a <u>structural induction</u> on σ.

<u>Basis</u>: - If σ is a constant function m, let $e \simeq [\sigma] = \circ\underline{\text{const}}, m$. By production 13, for any t and s,

$$\circ_e\, \underline{\text{const}}, m, (\text{EVAL}, t, s) \xRightarrow{13} \circ_e\, \underline{\text{const}}, m, (\text{VAL}, t, s, m)$$

Thus property $\pi_1(0)$ is verified by any constant.

- If σ is a variable x_j (or projection function p_j), it is trivial because $p_j(\bar{\bot}) = \bot$.

<u>Induction</u>: Lemma 5.1 with $i = 0$ shows that this step is verified. Thus, every term σ verifies property $\pi_1(0)$.

<u>Induction step</u> Suppose that every term σ verifies property $\pi_1(k)$ for some k, and let us show that every term σ verifies property $\pi_1(k+1)$. Let σ be any term, and let t such that

$$(\sigma\, \tau^{k+1}(\bar{\bot}))_t = m \quad \text{and} \quad m \neq \bot .$$

<u>Structural induction on σ</u>

<u>Basis</u>: - Let σ be a constant function m, and

$e \simeq [\sigma] = \circ\, \underline{\text{const}}, m$. For any t and $s \in \mathbb{N}^*$,

$$\underset{e}{\circ}\ \underline{\text{const}}, m, (\text{EVAL}, t, s) \overset{13}{\Longrightarrow} \underset{e}{\circ}\ \underline{\text{const}}, m, (\text{VAL}, t, s, m)$$

Thus property $\pi_1(k+1)$ is verified by every constant.

- Let σ be a variable x_j (or projection function p_j). We have that $(p_j\ \tau^{k+1}(\bar{I}))_t = (\tau_j\ \tau^k(\bar{I}))_t$. Let $e \simeq [x_j] = \circ\, \underline{\text{var}}, x_j$, and $e_j \simeq [\tau_j(\bar{X})]$. For any t and s,

$$\underset{e_j}{\circ} \longleftarrow \circ\ \underline{\text{var}}, x_j, (\text{EVAL}, t, s) \overset{12}{\Longrightarrow} \underset{e_j}{\circ}(\text{EVAL}, t, s) \longleftarrow \circ\ \underline{\text{var}}, x_j, (\text{WAIT}, t, s)$$

(by induction hypothesis on τ_j)

$$\models\!\!=^* \underset{e'_j}{\circ}(\text{EVAL}, t, s) \longleftarrow \circ\ \underline{\text{var}}, x_j, (\text{WAIT}, t, s)$$

$$\overset{14}{\Longrightarrow} \underset{e'_j}{\circ} \longleftarrow \circ\ \underline{\text{var}}, x_j, (\text{VAL}, t, s, m)$$

Thus property $\pi_1(k+1)$ is verified by every variable (projection function).

<u>Induction</u>: Lemma 5.1 with $i = k+1$ shows that this step is verified.

This completes the induction step on i.

$\square$ (Theorem 5.1)

6. Consistency of the interpreter

Here we will prove that if the interpretation of some term $\sigma(\overline{X})$ with time parameter t leads to a value m, then the value of $(\sigma(\overline{X}))_t$ in the minimal solution is also m. A more precise statement follows.

Theorem 6.1.

For any term $\sigma(\overline{X})$ in P, any $e \simeq [\sigma(\overline{X})]$, and any $t \in \mathbb{N}^*$, if there exists $s \in \mathbb{N}^*$ such that for some $e' \simeq [\sigma(\overline{X})]$ and some m,

$$\underset{e}{o}(\text{EVAL}, t, s) \models^* \underset{e'}{o}(\text{VAL}, t, s, m)$$

then

$$\text{(i)} \quad \forall s' \in \mathbb{N}^* \quad \underset{e}{o}(\text{EVAL}, t, s') \models^* \underset{e'}{o}(\text{VAL}, t, s', m)$$

and (ii) $\exists i : (\sigma\, \tau^i(\overline{\bot}))_t = m$ and $m \neq \bot$. $\square$

Note that condition (i) expresses the result of the evaluation is independent of the stack s.

Before proving this theorem we need the notion of complexity of a derivation which will be defined below.

Consider the semantics description of Lucid as given in Table II. Let E and E' be linked trees such that $E \models^* E'$ and $E, E' \simeq [P]$ for some program P. Also let e and e' be subtrees of E and E' respectively, such that

$$\underset{e}{o}(\text{EVAL}, t, s) \models^* \underset{e'}{o}(\text{VAL}, t, s, m)$$

for some $t, s \in \mathbb{N}^*$ and $m \in \mathbb{D}$.

In what follows n_o denotes the root node of e (or e') in E (or E').

Definition 6.1. An evaluation path generated by $(n_o, (\text{EVAL}, t, s))$ in the derivation implied by

$$\underset{e}{o}\,(\text{EVAL}, t, s) \models\!\!=^{*} \underset{e'}{o}\,(\text{VAL}, t, s, m)$$

is a (finite) set $H = \{(n_o, t, s), (n_1, t^1, s^1), \ldots, (n_k, t^k, s^k)\}$ such that (i) between every pair of nodes (n_i, n_{i+1}) there is an edge or a pointer,

and (ii) $\{(n_o, (\text{EVAL}, t, s)\}$ produces $\{(n_i, (\text{EVAL}, t^i, s^i)) \mid i = 1, \ldots, k\}$ and $\{(n_i, (\text{VAL}, t^i, s^i, m^i)) \mid i = 1, \ldots, k\}$ in the derivation implied above. □

Note that an evaluation path always includes (n_o, t, s).

Definition 6.2. Evaluation path H is said to be of complexity c if there are exactly c pairs of elements of H, $< (n_i, t^i, s^i), (n_{i+1}, t^{i+1}, s^{i+1}) >$ such that there is a pointer between n_i and n_{i+1}. □

The derivation implied by

$$\underset{e}{o}\,(\text{EVAL}, t, s) \models\!\!=^{*} \underset{e'}{o}\,(\text{VAL}, t, s, m)$$

has a finite number of evaluation paths since the number of productions used is finite and each of them produces a finite number of control labels. Let $c_1, c_2, \ldots, c_p$ be their respective complexities. The complexity of the derivation is defined as $\text{Max}\{c_1, c_2, \ldots, c_p\}$.

Now we will prove a lemma which simplifies the proof of Theorem 6.1. We will say that a term $\sigma(\overline{X})$ has property $\pi_2(k)$ if it satisfies Theorem 6.1 and the complexity of the derivation in question is less than or equal to k .

<u>Lemma 6.1</u>.

If σ_1, σ_2 and σ_3 are terms having property $\pi_2(c)$ for some c , then any term formed from one or more σ_j's ($j = 1, 2, 3$) by means of any Lucid special operator, any arithmetic and logical operators also has property $\pi_2(c)$.

<u>Proof</u>.

Several cases have to be considered.

a) Consider $f\sigma_1$ where $f \in \{\mathbf{first}, \mathbf{next}, \mathbf{latest}, \mathbf{latest}^{-1}\}$.

Let $e, e' \simeq [f\sigma_1] = \begin{array}{c} \circ\ f \\ | \\ \circ \\ [\sigma_1] \end{array}$ such that

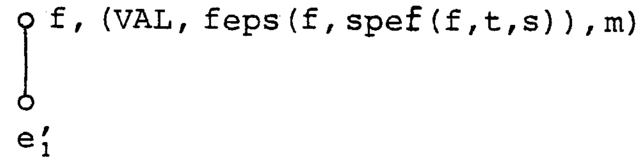
$$\underset{e}{\circ}\ (\text{EVAL}, t, s) \models^* \underset{e'}{\circ}\ (\text{VAL}, t, s, m)$$

whose complexity is c .

The first production used in this derivation has to be production 1

Thus: $$\begin{array}{l} \circ\ f, (\text{EVAL}, t, s) \\ | \\ \underset{e_1}{\circ} \end{array} \overset{1}{\Longrightarrow} \begin{array}{l} \circ\ f \\ | \\ \underset{e_1}{\circ}\ (\text{EVAL}, \text{spef}(f, t, s)) \end{array}$$

$$(\text{by hypothesis on } \sigma_1) \models^* \begin{array}{l} \circ\ f \\ | \\ \underset{e_1'}{\circ}\ (\text{VAL}, \text{spef}(f, t, s), m) \end{array}$$

$\overset{28}{\Longrightarrow}$ 

Also, feps(f, spef(f, t, s)) = (t, s) by definition of feps and spec .

Let (t', s') = spef(f, t, s). By hypothesis on σ_1 , m is independent of s' , thus m is independent of s . This satisfies part (i) of $\pi_2(c)$.

Also by hypothesis on σ_1 , $\exists i$: $m = (\sigma_1\ \tau^i(\overline{\bot}))_t$, and $m \neq \bot$. But $(f\,\sigma_1\,\tau^i(\overline{\bot}))_t = (\sigma_1\ \tau^i(\overline{\bot}))_t$, as can be checked from the definition of f and spef . It follows that

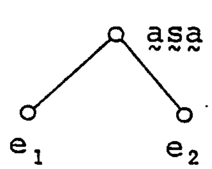
$$(f\,\sigma_1\,\tau^i(\overline{\bot}))_t = m \quad \text{and} \quad m \neq \bot .$$

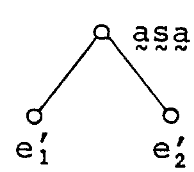
Thus, $f\,\sigma_1$ has property $\pi_2(c)$ if σ_1 has property $\pi_2(c)$.

b) Consider $\sigma_1\ \underset{\sim}{asa}\ \sigma_2$ and let e , $e' \simeq [\sigma_1\ \underset{\sim}{asa}\ \sigma_2]$ such that

$$\underset{e}{\circ}\ (\text{EVAL}, t, s) \quad \models\!\!=^{*} \quad \underset{e'}{\circ}\ (\text{VAL}, t, s, m)$$

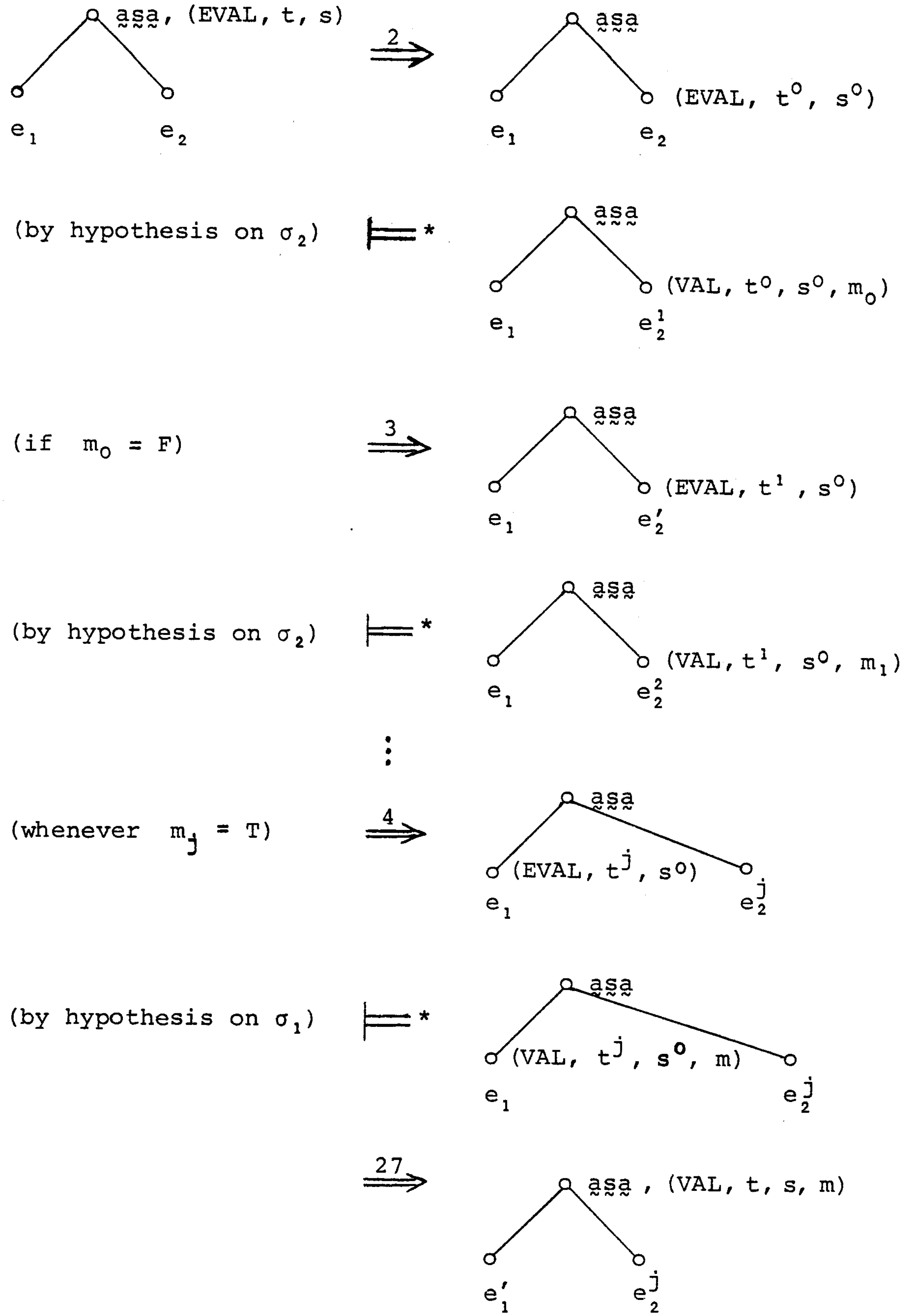
the complexity of which is c . Since any other possible control label at node n_0 (root of e in E) is not used in the above derivation, we may suppose, without loss of generality, that

e = 

, and that e' = 

for some e_1, $e_1' \simeq [\sigma_1]$,

and some e_2 , $e_2' \simeq [\sigma_1]$. The derivation should have the form:


asa, (EVAL, t, s)
e1
e2
2
asa
(EVAL, t0, s0)
e1
e2
(by hypothesis on σ2)
⊨*
asa
(VAL, t0, s0, m0)
e1
e2 1
(if m0 = F)
3
asa
(EVAL, t1, s0)
e1
e2′
(by hypothesis on σ2)
⊨*
asa
(VAL, t1, s0, m1)
e1
e2 2
⋮
(whenever mj = T)
4
asa
(EVAL, tj, s0)
e1
e2 j
(by hypothesis on σ1)
⊨*
asa
(VAL, tj, s0, m)
e1
e2 j
27
asa, (VAL, t, s, m)
e1′
e2 j

The hypothesis on σ_1 and σ_2 can be used because the complexity of the derivation involving each of them has to be the same as the complexity of the main derivation. If $t = t_0 t_1 \ldots t_n$, then $t^j = j t_1 \ldots t_n$, and it follows that feps ($\underset{\sim}{\text{first}}$, t^j, s^o) = (t, s) .

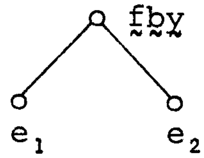
This explains the last step of the derivation.

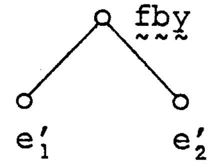
Moreover, $m_0, m_1, \ldots, m_j$ being all independent of s^o by hypothesis on σ_2, and m being independent of s^o by hypothesis on σ_1, it follows that m is independent of s . By hypothesis on σ_2, $m_0 = (\sigma_2\ \tau^{i_0}(\overline{I}))_{t^0}$, $m_1 = (\sigma_2\ \tau^{i_1}(\overline{I}))_{t^1}$, $\ldots$, $m_j = (\sigma_2\ \tau^{i_j}(\overline{I}))_{t^j}$ where $t^0 = 0t_1 \ldots t_n$, $t^1 = 1t_1 \ldots t_n$, $\ldots$, $t^j = j\, t_1 \ldots t_n$, $m_0 = m_1 = \ldots = m_{j-1} = F$, and $m_j = T$. Also, by hypothesis on σ_1, $m = (\sigma_1\ \tau^{i_{j+1}}(\overline{I}))_{t^j}$ and $m \neq \perp$.

Let $i = \mathrm{Max}\{i_0, i_1, \ldots, i_{j+1}\}$. Then $m = (\sigma_1\ \tau^i(\overline{I}))_{t^j}$ and $(\sigma_2\ \tau^i(\overline{I}))_{t^\ell} = F$ for $\ell < j$ and $(\sigma_2\ \tau^i(\overline{I}))_{t^j} = T$ and $m \neq \perp$.

Thus, by definition of $\underset{\sim}{\text{asa}}$, $m = ((\sigma_1\ \underset{\sim}{\text{asa}}\ \sigma_2)\ \tau^i(\overline{I}))_t$, $m \neq \perp$, and $\sigma_1\ \underset{\sim}{\text{asa}}\ \sigma_2$ has property $\pi_2(c)$ if σ_1 and σ_2 have it.

c) Consider $\sigma_1\ \underset{\sim}{\text{fby}}\ \sigma_2$, and let $e, e' \simeq [\sigma_1\ \underset{\sim}{\text{fby}}\ \sigma_2]$ such that $o_e(\mathrm{EVAL}, t, s) \models^* o_{e'}(\mathrm{VAL}, t, s, m)$ the complexity of which derivation is c .

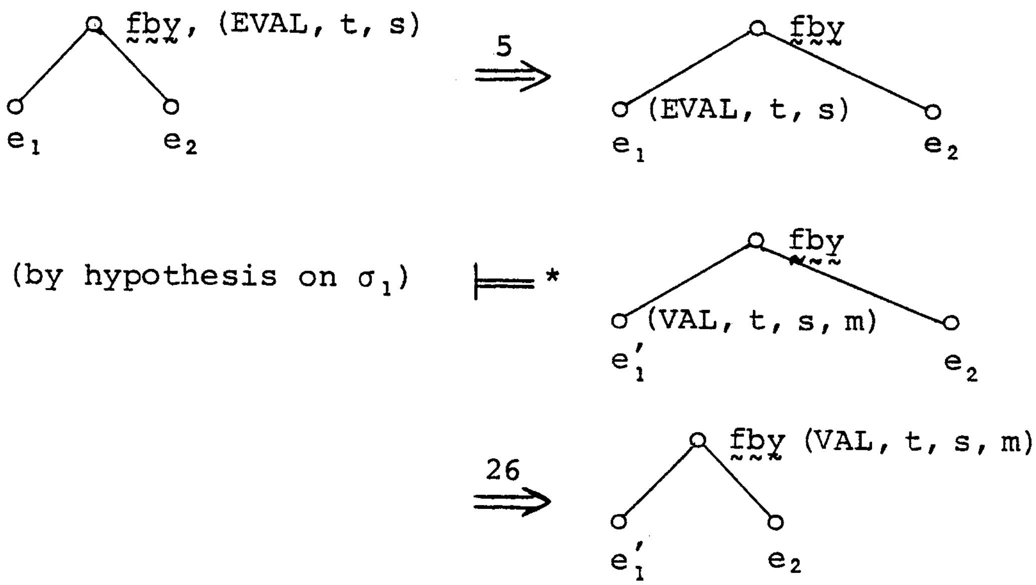
The other control labels at node n_0 (root of e in E) being of no effect in the above derivation, we may suppose without loss of generality that $e =$ 

, and that $e' =$ 

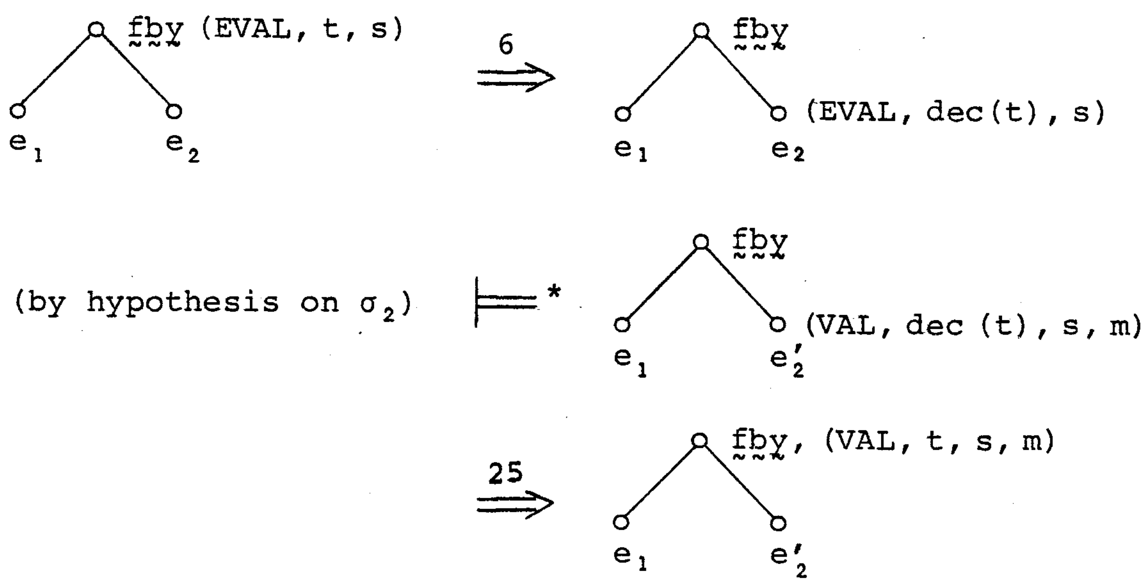
where e_1 , $e_1' \simeq [\sigma_1]$ and e_2 , $e_2' \simeq [\sigma_2]$.

The above derivation would have one of the two following forms depending on **t** :

- if $t \in \{0\} \times \mathbb{N}^*$



- if $t \notin \{0\} \times \mathbb{N}^*$



The hypothesis on σ_1 in the first case, and on σ_2 in the second one can be used because the complexity of the corresponding derivations is c .

The last derivation is obtained because $inc(dec(t)) = t$. Also, the hypotheses on σ_1 in the first case, and σ_2 in the second, show that m is independent of s , and

$$m = (\sigma_1\ \tau^{i_1}(\bar{I}))_t \qquad \text{if} \qquad t \in \{0\} \times \mathbb{N}^* \qquad (m \neq \bot)$$

$$m = (\sigma_2\ \tau^{i_2}(\bar{I}))_t \qquad \text{if} \qquad t \notin \{0\} \times \mathbb{N}^* \qquad (m \neq \bot)$$

By definition of $\underset{\sim}{\mathrm{fby}}$, if $i = \mathrm{Max}\,\{i_1, i_2\}$ then

$$m = ((\sigma_1\ \underset{\sim}{\mathrm{fby}}\ \sigma_2)\ \tau^{i}(\bar{I}))_t \quad \text{and} \quad m \neq \bot .$$

Thus, $\sigma_1\ \underset{\sim}{\mathrm{fby}}\ \sigma_2$ has property $\pi_2(c)$ if σ_1 and σ_2 have it.

d) Consider $g\sigma_1$ where $g \in \{-, \neg\}$, and let $e , e' \simeq [g\sigma_1]$ such that $\underset{e}{\circ}\ (EVAL, t, s) \models^* \underset{e}{\circ}\ (VAL, t, s, m)$ the complexity of the derivation being c .

Without loss of generality, we may suppose that $e = \begin{array}{l} \circ\ g \\ | \\ \circ \\ e_1 \end{array}$ and that $e' = \begin{array}{l} \circ\ g \\ | \\ \circ \\ e'_1 \end{array}$ where $e_1 , e'_1 \simeq [\sigma_1]$.

The above derivation would have the following form

$$\begin{array}{l} \circ\ g,\ (EVAL, t, s) \\ | \\ \circ \\ e_1 \end{array} \quad \overset{7}{\Longrightarrow} \quad \begin{array}{l} \circ\ g \\ | \\ \circ\ (EVAL, t, s) \\ e_1 \end{array}$$

$$\text{(by hypothesis on } \sigma_1\text{)} \quad \models^* \quad \begin{array}{l} \circ \; g \\ | \\ \circ \; (VAL, t, s, m_1) \\ e_1' \end{array}$$

$$\xRightarrow{24} \quad \begin{array}{l} \circ \; g, (VAL, t, s, uop(g, m_1)) \\ | \\ \circ \\ e_1' \end{array}$$

where m_1 is such that $uop(g, m_1) = m$.

Also, m_1 being independent of s by hypothesis on σ_1 , we have that $uop(g, m_1)$ is independent of s . By the same hypothesis on σ_1 , $m_1 = (\sigma_1 \; \tau^i(\bar{\perp}))_t$ and $m_1 \neq \perp$. Thus $(g\sigma_1 \; \tau^i(\bar{\perp}))_t = g(\sigma_1 \; \tau^i(\bar{\perp}))_t = gm_1$ it follows that

$$m = (g\sigma_1 \; \tau^i(\bar{\perp}))_t \quad \text{and} \quad m \neq \perp .$$

This shows that if σ_1 has property $\pi_2(c)$ then so does $g\sigma_1$.

e) Consider $\sigma_1 \; h \; \sigma_2$ where $h \in ALOP - \{\vee , \wedge\}$ and let $e , e' \simeq [\sigma_1 \; h \; \sigma_2]$ such that

$$\begin{array}{l} \circ \; (EVAL, t, s) \\ e \end{array} \quad \models^* \quad \begin{array}{l} \circ \; (VAL, t, s, m) \\ e' \end{array}$$

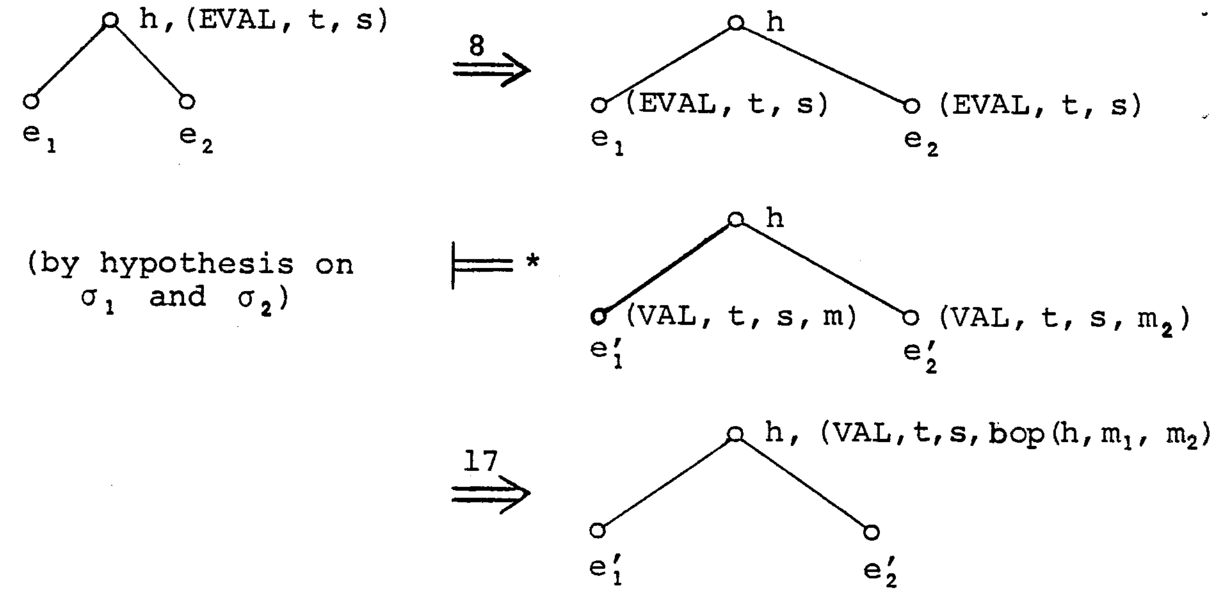
the complexity of the derivation being c .

Without loss of generality we may suppose that

$$e = \begin{array}{ccc} & \circ \, h & \\ \diagup & & \diagdown \\ \circ & & \circ \\ e_1 & & e_2 \end{array} \quad \text{and} \quad e' = \begin{array}{ccc} & \circ \, h & \\ \diagup & & \diagdown \\ \circ & & \circ \\ e_1' & & e_2' \end{array} \quad \text{where}$$

$e_1 , e_1' \simeq [\sigma_1]$ and $e_2 , e_2' \simeq [\sigma_2]$.

The derivation implied above should be as follows:



where m_1 and m_2 are such that $bop(h,\ m_1,\ m_2) = m$. By hypothesis on σ_1 and σ_2 , m_1 and m_2 are independent of s . Thus $m = bop\ (h, m_1, m_2)$ is independent of s . Also,

$m_1 = (\sigma_1\ \tau^{i_1}(\bar{\perp}))_t$, $m_2 = (\sigma_2\ \tau^{i_2}(\bar{\perp}))_t$ and $m_1 \neq \perp$, $m_2 \neq \perp$.

Thus for $i = \mathrm{Max}\{i_1 , i_2\}$,
$$
\begin{aligned}
((\sigma_1\ h\ \sigma_2)\ \tau^i(\bar{\perp}))_t &= (\sigma_1\ \tau^i(\bar{\perp}))_t\ h(\sigma_2\ \tau^i(\bar{\perp}))_t \\
&= m_1\ h\ m_2 \\
&= bop(h,\ m_1,\ m_2) = m
\end{aligned}
$$
and $m \neq \perp$.

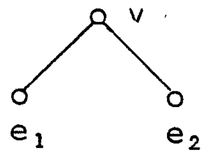
This shows that if σ_1 and σ_2 have property $\pi_2(c)$ then so does $\sigma_1\ h\ \sigma_2$.

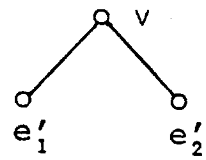
f) Consider $\sigma_1 \vee \sigma_2$ and let e , $e' = [\sigma_1 \vee \sigma_2]$ such that

$$
\underset{e}{\circ\ (EVAL,\ t,\ s)} \quad \models\!\!=^{*} \quad \underset{e'}{\circ\ (VAL,\ t,\ s,\ m)}
$$

the complexity of the derivation being c .

Without loss of generality we may suppose that

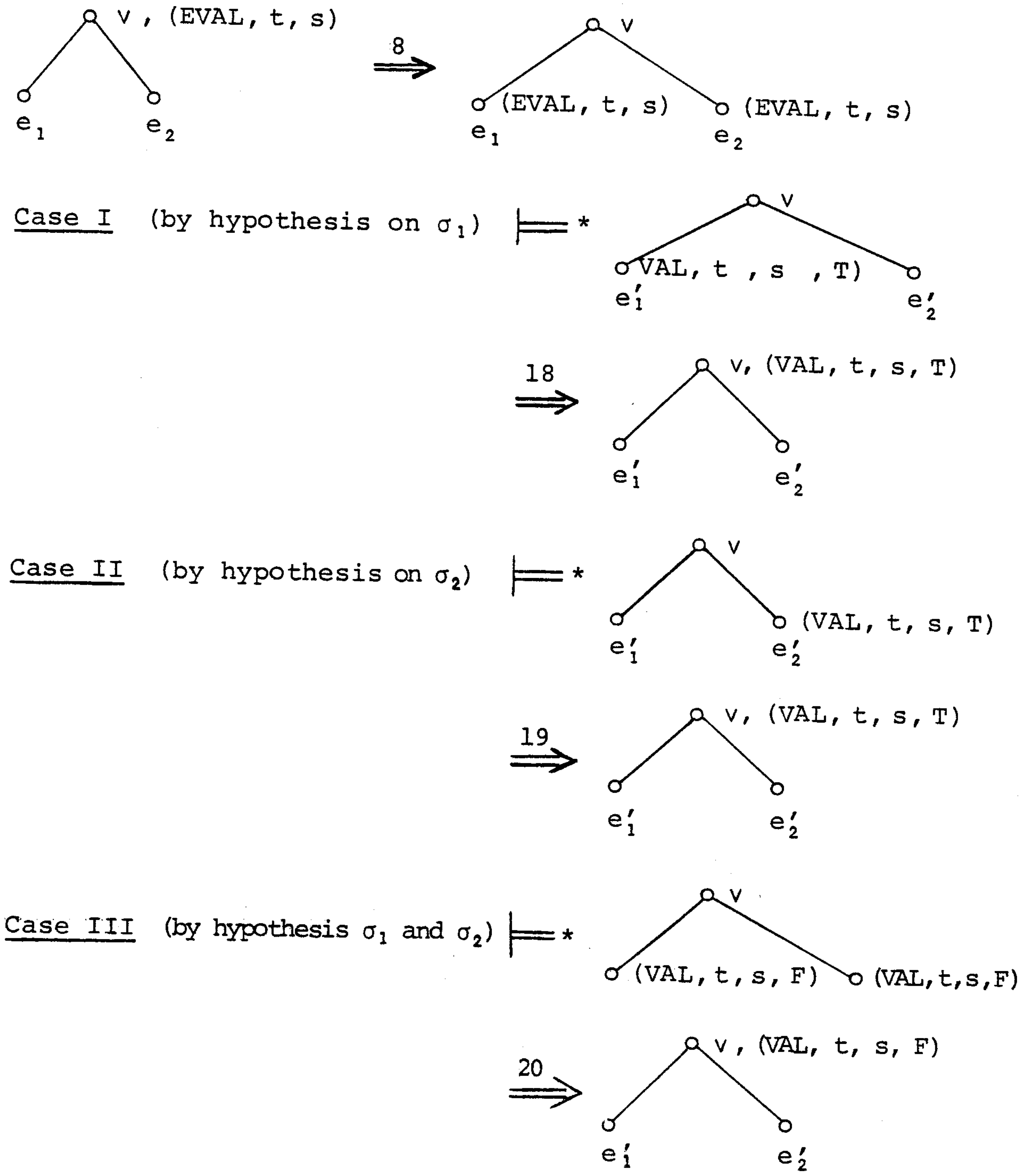
$e =$ 

and $e' =$ 

where $e_1,\ e_1' \simeq [\sigma_1]$

and $e_2,\ e_2' \simeq [\sigma_2]$.

The derivation would be of any of the three possible forms that follow:



By hypothesis on σ_1 and σ_2 the values T or F are independent of s . In case I, m = T and $(\sigma_1\ \tau^{i_1}(\bar{\imath}))_t = T$. But $((\sigma_1 \vee \sigma_2)\ \tau^{i_1}(\bar{\imath}))_t = (\sigma_1\ \tau^{i_1}(\bar{\imath}))_t \vee (\sigma_2\ \tau^{i_1}(\bar{\imath}))_t$

$= T$ by definition of $\vee$.

Case II is similar to case I with σ_2 instead of σ_1 . In case III, m = F , $(\sigma_1(\tau^{i_1}(\bar{\imath})))_t = F$ and $(\sigma_2\ \tau^{i_2}(\bar{\imath}))_t = F$. Let $i = \mathrm{Max}\{i_1, i_2\}$ then $((\sigma_1 \vee \sigma_2)\ \tau^{i}(\bar{\imath}))_t = (\sigma_1\ \tau^{i}(\bar{\imath}))_t \vee (\sigma_2\ \tau^{i}(\bar{\imath}))_t$

$= F \vee F = F$

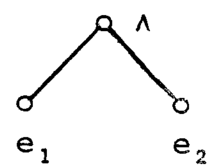
Thus in all cases $\sigma_1 \vee \sigma_2$ has property $\pi_2(c)$ if σ_1 and σ_2 have property $\pi_2(c)$.

g) Consider $\sigma_1 \wedge \sigma_2$ and let e , $e' \simeq [\sigma_1 \vee \sigma_2]$ such that

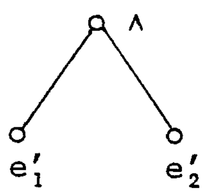
$$\underset{e}{o}\,(\mathrm{EVAL}, t, s) \models^{*} \underset{e'}{o}\,(\mathrm{VAL}, t, s, m)$$

the complexity of which is c .

Without loss of generality we may suppose that e =

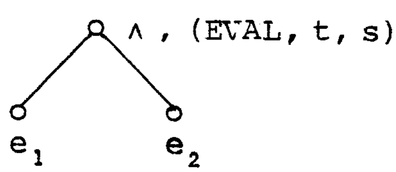


and e' =

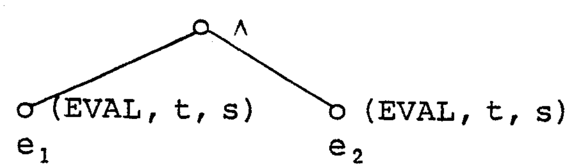


where e_1 , $e_1' \simeq [\sigma_1]$ and e_2 , $e_2' \simeq [\sigma_2]$.

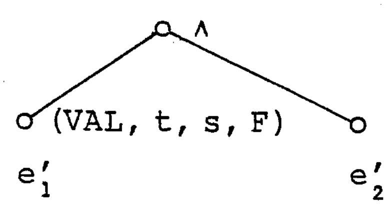
The derivation could then be any of the following three:



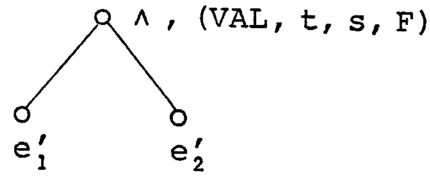
$\overset{8}{\Longrightarrow}$



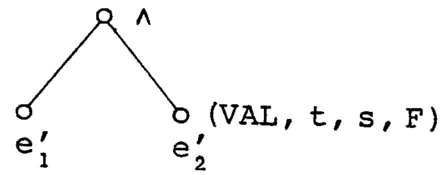
Case I (by hypothesis on σ_1) $\models^{*}$



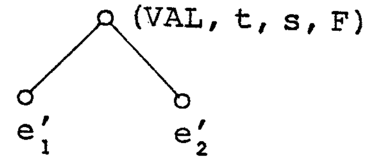
$\xRightarrow{21}$



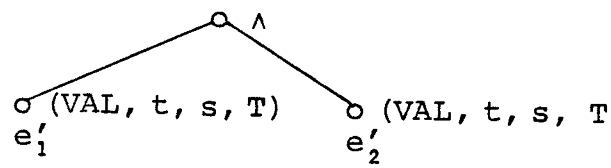
Case II (by hypothesis on σ_2) $\models^*$



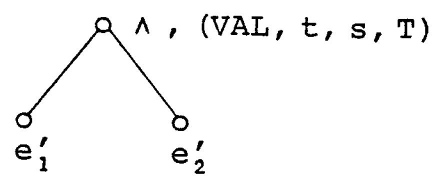
$\xRightarrow{22}$



Case III (by hypothesis on σ_1 and σ_2) $\models^*$



$\xRightarrow{23}$



By hypothesis on σ_1 and σ_2 the values F or T obtained are independent of s . In case I $(\sigma_1\ \tau^{i_1}(\bar{I}))_t = F$ (hypothesis on σ_1) , $m = F$ and $((\sigma_1 \wedge \sigma_2)\ \tau^{i_1}(\bar{I}))_t = (\sigma_1\ \tau^{i_1}(\bar{I}))_t \wedge (\sigma_2\ \tau^{i_1}(\bar{I}))_t$

$= F$ by definition of $\wedge$.

Case II is similar to case I.

In case III $(\sigma_1\ \tau^{i_1}(\bar{I}))_t = T$, $(\sigma_2\ \tau^{i_1}(\bar{I}))_t = T$ and $m = T$. So, if we let $i = \mathrm{Max}\,\{i_1 , i_2\}$ then

$$((\sigma_1 \wedge \sigma_2)\ \tau^{i}(\bar{I}))_t = (\sigma_1\ \tau^{i}(\bar{I}))_t \wedge (\sigma_2\ \tau^{i}(\bar{I}))_t$$

$$= T \wedge T = T$$

Thus, in all three cases $m = ((\sigma_1 \wedge \sigma_2)\ \tau^i(\bar{\bot}))_t$ for some i and $m \neq \bot$.

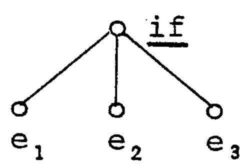
Hence, $\sigma_1 \wedge \sigma_2$ has property $\pi_2(c)$ if σ_1 and σ_2 have it.

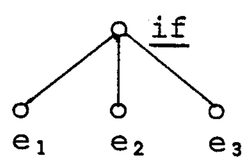
h) Consider $\underline{if}\ \sigma_1\ \underline{then}\ \sigma_2\ \underline{else}\ \sigma_3$ and let $e, e' \approx [\underline{if}\ \sigma_1\ \underline{then}\ \sigma_2\ \underline{else}\ \sigma_3]$ such that

$$\overset{\circ\ (EVAL,\ t,\ s)}{e} \quad \models\!\!=^{*} \quad \overset{\circ\ (VAL,\ t,\ s,\ m)}{e'}$$

the complexity of the derivation being c.

Without loss of generality we may assume that

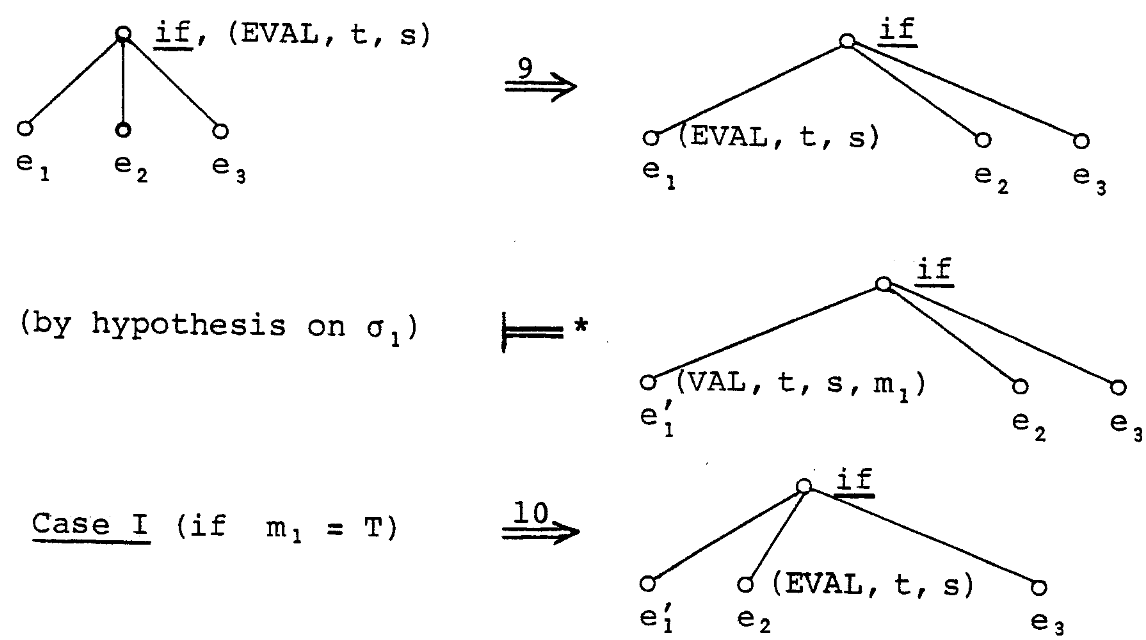
e = 

and e′ = 

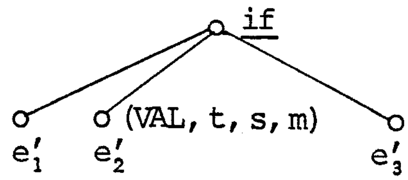
where $e_1, e_1' \approx [\sigma_1]$

$e_2, e_2' \approx [\sigma_2]$ and $e_3, e_3' \approx [\sigma_3]$.

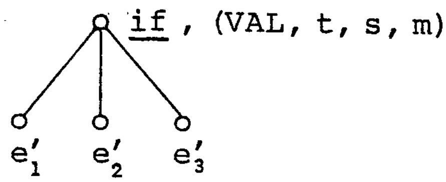
The above derivation has to have one of the following two forms:



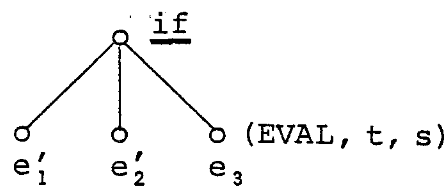
(by hypothesis on σ_2) $\models^*$



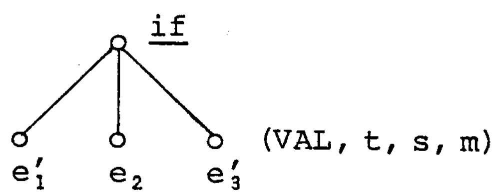
$\xRightarrow{15}$



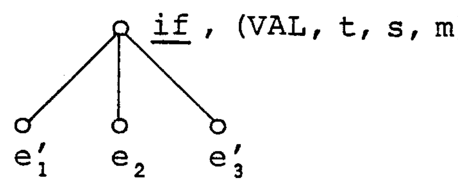
<u>Case II</u> (if $m_1 = F$) $\xRightarrow{11}$



(by hypothesis on σ_3) $\models^*$



$\xRightarrow{16}$



By hypothesis on σ_1, σ_2 and σ_3, in both cases, m is independent of s. Moreover in case I, $(\sigma_1\ \tau^{i_1}(\bar{I}))_t = T$ and $(\sigma_2\ \tau^{i_2}(\bar{I}))_t = m$. So, if we let $i = \mathrm{Max}\,\{i_1, i_2\}$ then

$$((\underline{\text{if}}\ \sigma_1\ \underline{\text{then}}\ \sigma_2\ \underline{\text{else}}\ \sigma_3)\ \tau^i(\perp))_t = \underline{\text{if}}(\sigma_1\ \tau^i(\bar{I}))_t\ \underline{\text{then}}(\sigma_2(\tau^i(\bar{I})))_t\ \underline{\text{else}}(\sigma_3\ \tau^i(\bar{I}))_t$$

(by definition of <u>if</u> <u>then</u> <u>else</u>) $= (\sigma_2\ \tau^i(\bar{I}))_t = m$ and $m \neq \perp$ by hypothesis on σ_1.

In case II, $(\sigma_1\ \tau^{i_1}(\overline{\perp}))_t = F$ and $(\sigma_3\ \tau^{i_3}(\overline{\perp}))_t = m$. If we let $i = \text{Max}\,\{i_1, i_3)\}$ then

$$((\underline{\text{if}}\ \sigma_1\ \underline{\text{then}}\ \sigma_2\ \underline{\text{else}}\ \sigma_3)\ \tau^i(\overline{\perp}))_t = \underline{\text{if}}(\sigma_1\ \tau^i(\overline{\perp}))_t\ \underline{\text{then}}(\sigma_2\ \tau^i(\overline{\perp}))_t\ \underline{\text{else}}(\sigma_3\ \tau^i(\overline{\perp}))_t$$

(by definition of if-then-else) $= (\sigma_3\ \tau^i(\overline{\perp}))_t = m$

Thus, if we let $i = \text{Max}\ (i_1, i_2, i_3\}$ then in any case

$$((\text{if}\ \sigma_1\ \text{then}\ \sigma_2\ \text{else}\ \sigma_3)\ \tau^i(\overline{\perp}))_t = m \quad \text{and} \quad m \neq \perp .$$

Hence, $\underline{\text{if}}\ \sigma_1\ \underline{\text{then}}\ \sigma_2\ \underline{\text{else}}\ \sigma_3$ has property $\pi_2(c)$ if σ_1, σ_2 and σ_3 have property $\pi_2(c)$. □

Now we can prove Theorem 6.1.

Proof (Theorem 6.1).

It will be carried out by induction on the complexity c of the derivation.

Base step $(c = 0)$

In the derivation implied by

$$\underset{e}{o}\ (\text{EVAL}, t, s) \models^{*} \underset{e}{o}\ (\text{VAL}, t, s, m)$$

every evaluation path generated by (EVAL, t, s) is of complexity 0, i.e. has no pair of nodes with a pointer between them.

Let us perform an induction on the structure of the term σ.

Basis: - If σ is a constant function m' let $e \simeq [\sigma]$.

The only production that can be used is production 13:

$$o\ \underline{\text{const}}, m', (\text{EVAL}, t, s) \xrightarrow{13} o\ \underline{\text{const}}, m', (\text{VAL}, t, s, m')$$

Thus, m' has to be equal to m.

Also m' is independent of s, and so is m.

As $(m\,\tau^{i}(\bar{I}))_t = m$, it follows that property $\pi_2(0)$ is verified by any constant.

- σ cannot be a variable x_j because otherwise the derivation would have to be of the form

$$\underset{e_j}{\circ} \leftarrow \circ\ \underline{var}, x_j, (EVAL, t, s) \overset{12}{\Longrightarrow} \underset{e_j}{\circ}(EVAL,t,s) \leftarrow \circ\ \underline{var}, x_j, (WAIT,t,s)$$

$$\models^{*} \underset{e'_j}{\circ}(VAL, t, s, m) \leftarrow \circ\ \underline{var}, x_j, (WAIT, t, s)$$

$$\overset{14}{\Longrightarrow} \underset{e'_j}{\circ} \leftarrow \circ\ \underline{var}, x_j, (VAL, t, s, m)$$

where $e_j \simeq [\tau_j(\bar{X})]$.

This means that the root node r_j of e_j is in an evaluation path generated by $(EVAL, t, s)$ at the root node r of e . This is impossible because there is a pointer between r and r_j , and $c = 0$.

Induction: Lemma 6.1 used with $c = 0$ shows that this step is verified. Thus every term $\sigma(\bar{X})$ has property $\pi_2(0)$.

Induction step

Suppose that every term σ has property $\pi_2(k)$ and let us show that every term σ has property $\pi_2(k+1)$. Let $e \simeq [\sigma]$ for some σ such that

$$\underset{e}{\circ}\,(EVAL, t, s) \models^{*} \underset{e'}{\circ}\,(VAL, t, s, m)$$

and suppose that the complexity of this derivation is $k + 1$.

<u>Induction on the structure of σ</u> .

<u>Basis</u>: - If σ is a constant function, then the proof given in the case $c = 0$ applies here. Thus every constant has property $\pi_2(k+1)$.

- If σ is a variable x_j then let $e, e' \simeq [x_j]$ and $e_j \simeq [\tau_j(\overline{X})]$.

We may assume that $e = \circ\ \underline{\text{var}}, x_j$ because no other control label at the root node of e would be used in the derivation. We have

$$\begin{array}{l} \qquad\circ\ \underline{\text{var}},\ x_j,\ (\text{EVAL}, t, s) \\ \swarrow \\ \circ \\ e_j \end{array} \overset{12}{\Longrightarrow} \begin{array}{l} \qquad\circ\ \underline{\text{var}},\ x_j,\ (\text{WAIT}, t, s) \\ \swarrow \\ \circ\,(\text{EVAL}, t, s) \\ e_j \end{array}$$

$$\models^{*} \begin{array}{l} \qquad\circ\ \underline{\text{var}},\ x_j,\ (\text{WAIT}, t, s) \\ \swarrow \\ \circ\,(\text{EVAL}, t, s, m) \\ e'_j \end{array}$$

$$\overset{14}{\Longrightarrow} \begin{array}{l} \qquad\circ\ \underline{\text{var}},\ x_j,\ (\text{VAL}, t, s, m) \\ \swarrow \\ \circ \\ e'_j \end{array}$$

If $k+1$ is the complexity of this derivation then the complexity of the sub-derivation evaluating e_j is k . So, by the induction hypothesis, m is independent of s and $m = \tau_j(\tau^i(\overline{I}))_t$. But we have that $(p_j\ \tau^{i+1}(\overline{I}))_t = (\tau_j\ \tau^i(\overline{I}))_t = m$.

Induction: Lemma 6.1 with $c = k+1$ shows that this induction step is satisfied.

Thus any term σ has property $\pi_2(k+1)$ if every term has property $\pi_2(k)$.

$\square$ (Theorem 6.1)

7. Total correctness of the interpreter

The total correctness of the evaluation of any right hand side term in a Lucid assertion results from Theorems 5.1 and 6.1 and is expressed as follows.

Theorem 7.1.

For any $j \in \{1, \ldots, v\}$, any $e_j, e'_j \simeq [\tau_j(\bar{X})]$ and any $t, s \in \mathbb{N}^*$:

$$e_j^{o\,(EVAL,\, t,\, s)} \models^{*} e_j'^{\,o\,(VAL,\, t,\, s,\, m)}$$

iff

$$\exists i: (\tau_j\ \tau^i(\bar{\imath}))_t = \text{ and } m \neq \perp .$$

Proof.

Immediate from Theorems 5.1 and 6.1 since $\tau_j(\bar{X})$ is a term in program P . $\square$

The input/output total correctness of a Lucid program is shown next.

<u>Theorem 7.2</u>.

Given a program P with a variable OUTPUT, $k \in \mathbb{N}$ and $e \simeq [P]$:

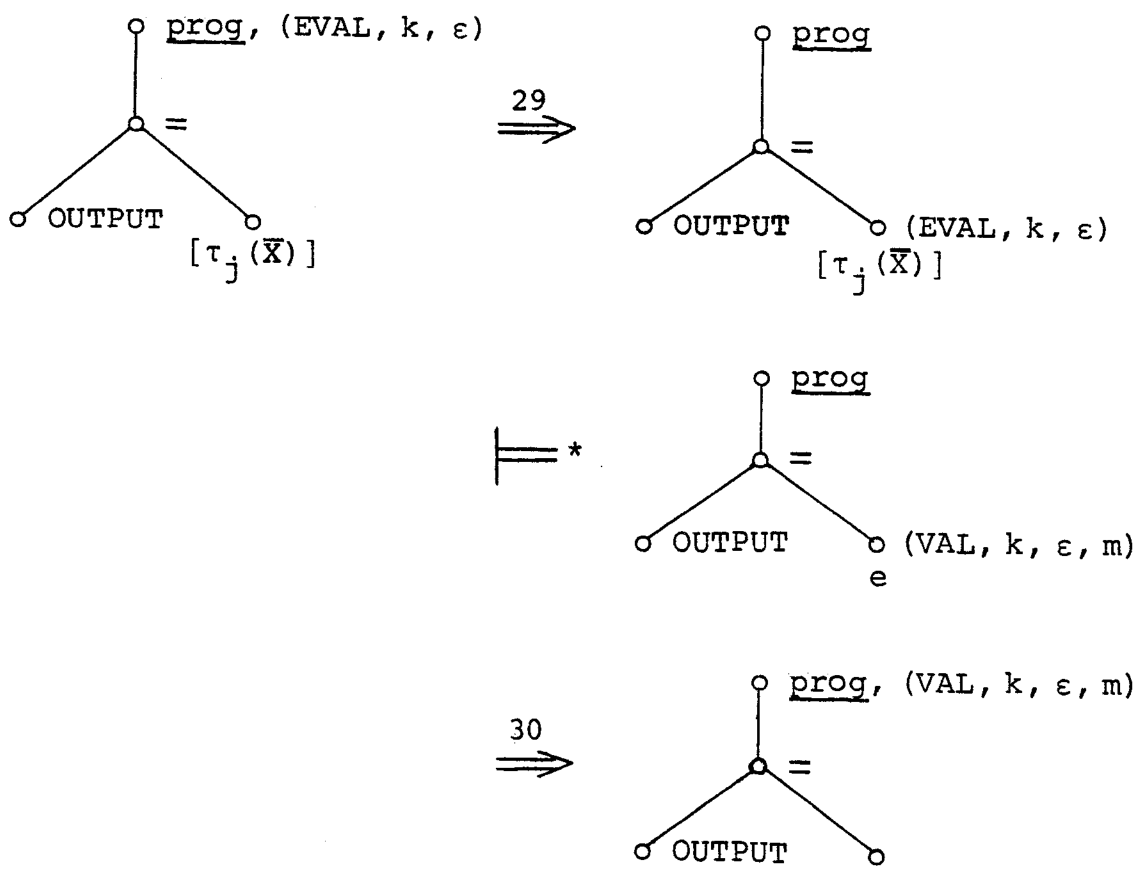
$$\left(\text{START}, \underset{[P]}{o}(\text{EVAL}, k, \varepsilon)\right) \vdash^{*} \left(\text{STOP}, \underset{e}{o}(\text{VAL}, k, \varepsilon, m)\right)$$

iff

$\exists i$, $\exists j$: OUTPUT $= x_j$ and $(\tau_j\ \tau^i(\bar{\bot}))_k = m$ and $m \neq \bot$.

<u>Proof</u>.

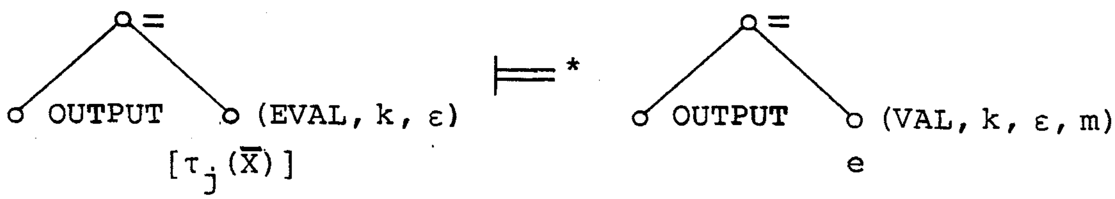
a) <u>Only if part</u>: Consider the derivation implied by the given transformations. Production 29 labelled START should be used first, and STOP is the label that is reached at the end of the derivation. Assuming that OUTPUT $= \tau_j(\bar{X})$ is the assertion defining variable OUTPUT, the relevant part of the derivation has to be:



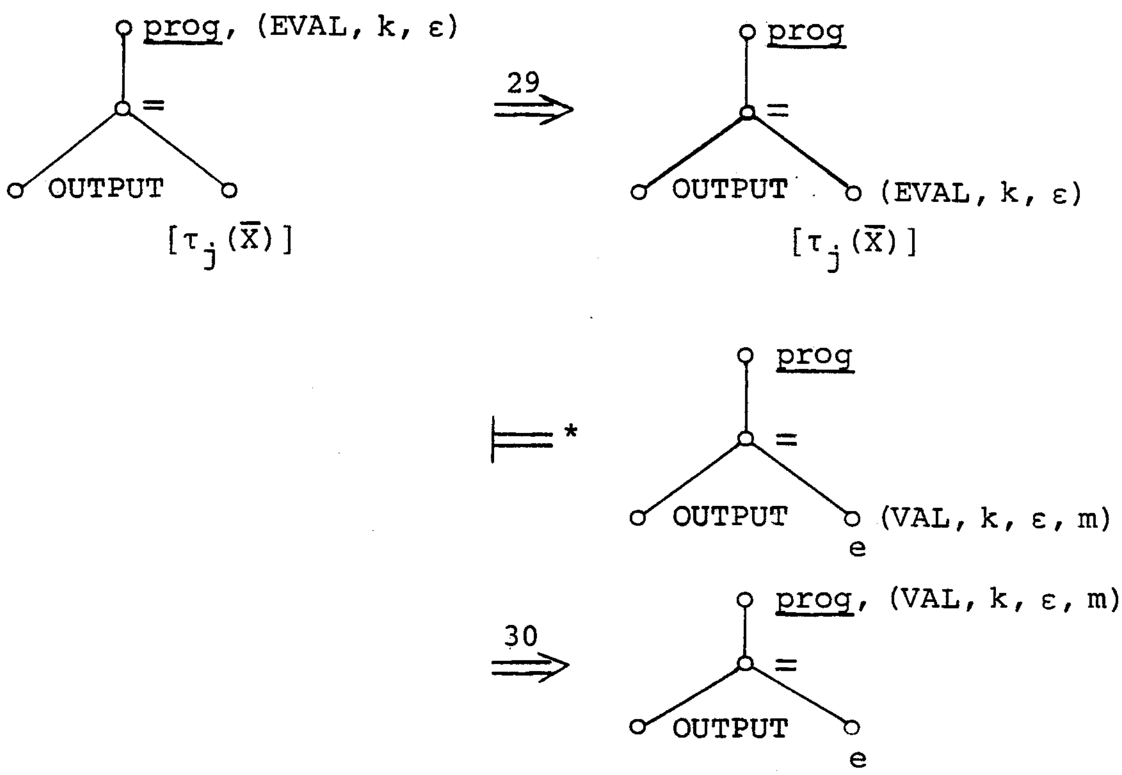
With x_j = OUTPUT, Theorem 7.1 implies that

$$\exists i: \ (\tau_j \ \tau^i(\bar{\perp}))_t = m \ \text{ and } \ m \neq \perp .$$

b) <u>if part</u>: Suppose that for x_j = OUTPUT, $\tau_j(\tau^i(\bar{\perp}))_k = m$ and $m \neq \perp$. Then by Theorem 7.1 and for some $e \simeq [\tau_j(\bar{X})]$ we have



Therefore



Hence the result. □

References

[1] Ashcroft, E. A., and Wadge, W. W. "Lucid, a Formal System for Writing and Proving Programs", SIAM J. on Computing, 5, 3 (September, 1976).

[2] Ashcroft, E. A., and Wadge, W. W. "Lucid, a Non-procedural Language with Iteration", Comm. ACM 20, 7 (July, 1977).

[3] Cargill, T. A., "Deterministic Operational Semantics of Lucid", Research Report CS-76-19, Dept. of Computer Science, University of Waterloo.

[4] Culik II, K. "A Model for the Formal Definition of Programming Languages", Intern. J. Computer Mathematics, Section A, Vol. 3, pp. 315-345.

[5] Culik II, K., Farah, M. "Linked Forest Manipulation Systems a Tool for Computational Semantics", Research Report CS-77-18, Dept. of Computer Science, University of Waterloo.

[6] Farah, M. "A Formal Description of ALTRAN Using Linked Forest Manipulation Systems", Research Report CS-73-08, Dept. of Computer Science, University of Waterloo.

[7] Manna, Z. "Introduction to Mathematical Theory of Computation", McGraw-Hill, 1974.

[8] Zoltan, A. C. "A Formal Definition of ALGOL 60 Using Linked Forest Manipulation Systems", Research Report CSRR-1072, Dept. of Computer Science, University of Waterloo.