RAIRO
INFORMATIQUE THÉORIQUE

ALEX PELIN

A formalism for treating equivalence of recursive procedures

RAIRO – Informatique théorique, tome 19, n° 3 (1985), p. 293-313.

<http://www.numdam.org/item?id=ITA_1985__19_3_293_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 19, n° 3, 1985, p. 293 à 313)

A FORMALISM FOR TREATING EQUIVALENCE OF RECURSIVE PROCEDURES (*)

by Alex PELIN (1)

Communicated by J. GALLIER

Abstract. — *We present a formal system for studying the equivalence of recursive procedures. The procedures are defined to include assignment statements and such constructs as branching and recursion. A fair amount of applications is provided to show how this approach can be used in proving equivalence of procedures. We conclude with results concerning the use of memory variables by the procedures.*

Résumé. — *Nous présentons un système formel pour étudier l'équivalence des schémas récursifs. Les schémas récursifs sont définis de telle sorte qu'ils ont des assignations, des tests et des procédures récursives. Un certain nombre d'examples sont donnés pour montrer comment cette approche peut être utilisée pour prouver l'équivalence de procédures. Nous finissons en montrant quelques résultats au sujet de l'utilisation de variables mémoires par des procédures.*

1. INTRODUCTION

The concept of " procedure " is fundamental in Computer Science. The solutions to most problems are given by procedures and in a lot of cases, by recursive procedures. In general, a recursive procedure has fewer instructions than its iterative version and in most cases it is easier to prove the correctness of a recursive procedure than the correctness of an equivalent iterative procedure (*see* R. S. Bird [2]). Most programming languages today support recursion and there are ways of transforming recursive procedures into equivalent iterative versions (*see* E. Horowitz and S. Sahni [4] and R. S. Bird [1]).

An important concept in studying procedures is the notion of equivalent procedures. Two procedures P_1 and P_2 are said to be equivalent if for any given input they either both stop and produce the same output or they both loop forever.

(*) Received and accepted in August 1984.
(1) Computer Science Department Temple University Philadelphia, PA. 19122 U.S.A.

In studying the equivalence of recursive procedures it is easier to make abstraction of the programming language in which the procedures are implemented. The approach taken here is to see a procedure as a recursive schema with an interpretation.

In order to make the study of procedures more useful, the language of recursive schemata has to have assignments and such programming constructs as sequencing, branching and recursion.

Following D. Scott [9, 10, 11] and J. W. De Bakker [3] procedures are interpreted as the minimal fixed point of the transformation induced by the body of the procedure. The formalism presented here is an extension of J. W. De Bakker's [3].

De Bakker interprets procedures as partial functions from a domain D to D. In our formalism we make explicit use of memory variables in predicates and we have assignment statements. We interpret procedures as maps from $D^{\ell M}$ to $D^{\ell M}$, where ℓM is a given set of memory locations and D is a non-empty domain. The predicates are interpreted as total functions in our system while De Bakker interprets them as partial functions. The first 15 axioms in § 3 are from De Bakker [3]. The other 4 axioms are added to deal with memory variables. The constructions of while loop, and, or and negation predicates are De Bakker's. Due to the fact that we interpret predicates as total functions we are able to establish the commutativity and associativity of the and or predicates.

The results in § 5 are new.

Our axiom system is similar to the one given for the predicate calculus with equality (see E. Mendelsohn [7]).

A fair amount of applications are provided to show how this formalism is used for proving the equivalence of procedures. These include theorems concerning the use of memory.

2. THE LANGUAGE OF THE EXTENDED μ-CALCULUS

The formalism presented here is an extension of J. W. De Bakker's μ-Calculus (*see* J. W. De Bakker [3]). Following the approach used by logicians the formal symbols will be introduced first, then the terms of the language (which are recursive schemata) and later on the formulae.

The *formal symbols* of the language of the Extended μ-Calculus are :

1. A set of memory variables $\ell M = \{ m_1, m_2, ..., m_k, ... \}$, $k \in \omega$. An arbitrary n-tuple of memory variables will be denoted by $\langle m^i_{i_1}, m^i_{i_2}, ..., m^i_{i_n} \rangle$;

2. A set of function constants $1F = \{ f(i,j) \}$ where $i, j \in \omega$; $f(i,j)$ represents the ith function of arity j;

3. A set of procedure variables $\mathbb{X} = \{ Xi \}$, $i \in \omega$;

4. A set of predicte constants 1 $P = \{ p(i, j) \}$ where $i, j \in \omega$; $p(i, j)$ is the ith predicate of arity j;

5. The set 1 $A = \{ ',', ';', '\leqslant', '=', '\leftarrow', '\rightarrow', '\Omega', 'E', '\mu', '[', ']', '(', ')' \}$.

Here ω denotes the set of finite ordinals. The terms of the language are given by the following definition :

DEFINITION 1 :

1. Ω, *E and* $X \in \mathbb{X}$ *are terms ;*

2. *If* $f(i, n) \in 1$ F *and* $\langle x_1, x_2, ..., x_n, x \rangle$ $(n + 1)$ *tuple of memory variables, then* $x \leftarrow f(i, n)(x_1, x_2, ..., x_n)$ *is a term ;*

3. *If* τ_1 *and* τ_2 *are terms, so is* $(\tau_1 ; \tau_2)$ *;*

4. *If* τ_1 *and* τ_2 *are terms,* $p(i, n) \in 1$ P *and* $\langle x_1, x_2, ..., x_n \rangle$ *is an n-tuple of memory variables, then* $(p(i, n)(x_1, ..., x_n) \rightarrow \tau_1, \tau_2)$ *is a term ;*

5. *If* τ *is a term and X a procedure variable then* $\mu X[\tau]$ *is a term.*

Rule 2 introduces assignments, rule 3 composition, rule 4 branching and rule 5 recursion.

DEFINITION 2 :

1. *An* atomic formula *is either an equivalence* $(\tau_1 = \tau_2)$ *or an inclusion* $(\tau_1 \leqslant \tau_2)$ *;*

2. *A formula is a list of zero or more atomic formulae, written* $\Phi_1, \Phi_2, ..., \Phi_n$, each Φ_i, $1 \leqslant i \leqslant n$, *an atomic formula.*

Example 1 : The following are terms :

1. $(m_1 \leftarrow f(0, 1)(m_2) ; m_2 \leftarrow f(1, 2)(m_3, m_4))$.

2. $\mu X_1[(p(0, 1)(m_1) \rightarrow (m_1 \leftarrow f(0, 1)(m_1) ; X_1), E)]$.

An *assignment* is a quadruple $s = \langle D, F, T, V \rangle$ where :

1. D is some non-empty domain;

2. F is a function which associates with each function symbol $f(i, n)$ a total function $F(f(i, n)) : D^n \rightarrow D$: if $n = 0$ then $F(f(i, n)) \in D$;

3. T attaches to each predicate symbol $p(i, n)$ a total function $T(p(i, n)) : D^n \rightarrow \{ 0, 1 \}$: if $n = 0$ then $T(p(i, n)) \in \{ 0, 1 \}$;

4. Let $D^{\ell M}$ denote the set of all total functions from 1 M to D, i.e. $D^{\ell M}$ set of states as in J. McCarthy [6].

Let $P(D^{\ell M})$ be the set of all partial functions from $D^{\ell M}$ to $D^{\ell M}$. The function V attaches to each $X_i \in \mathbb{X}$ a function $V(X_i) \in P(D^{\ell M})$.

Given an assignment $s = \langle D, F, T, V \rangle$ we can define an interpretation I for an assertion $A : \phi \vdash \Psi$ as shown below :

1. *Interpretation of terms* :

Given an assignment $s = \langle D, F, T, V \rangle$, each term τ is interpreted as a partial function $I_t(\tau)(s)$ from $D^{\ell M}$ to $D^{\ell M}$ as follows :

1.1. For each $\psi \in D^{\ell M}$, $I_t(\Omega)(s)(\psi)$ undefined i.e. $I_t(\Omega)(s)$ is the totally undefined function.

1.2. For each $\psi \in D^{\ell M}$, $I_t(E)(s)(\psi) = \psi$.

1.3. For each $X_i \in \mathbb{X}$, $I_t(X_i)(s) = V(X_i)$.

1.4. If $\tau = m \leftarrow f(i, n)(m_{i_1}, m_{i_2}, \ldots, m_{i_n})$ then for any

$$\psi \in D^{\ell M},\ (I_t(\tau)(s))(\psi) = \psi' \in D^{\ell M}$$

where

$$\psi'(y) = \begin{cases} \psi(y) & \text{if } y \neq m \\ F(f(i, n))(\psi(m_{i_1}), \ldots, \psi(m_{i_n})) & \text{if } y = m. \end{cases}$$

1.5. If $\tau = (p(i, n)(m_{i_1}, m_{i_2}, \ldots, m_{i_n}) \rightarrow \tau_1, \tau_2)$ then for any $\psi \in D^{\ell M}$

$$(I_t(\tau)(s))(\psi) = \begin{cases} (I(\tau)(s))(\psi) & \text{if } T(p(i, n))(\psi(m_{i_1}), \ldots, \psi(m_{i_n})) = 1 \\ (I(\tau)(s))(\psi) & \text{if } T(p(i, n))(\psi(m_{i_1}), \ldots, \psi(m_{i_n})) = 0. \end{cases}$$

1.6. If $\tau = (\tau_1 ; \tau_2)$ then $I(\tau)(s) = (I(\tau_2)(s)) \circ (I(\tau_2)(s))$ where $\circ$ is the composition operation for partial functions.

1.7. On the set of partial functions $P(D^{\ell M})$ we introduce the operations $(\ ;\ )$ and $(p - > —, —)$ and a partial order $\leqslant$. The operation $(\ ;\ )$ takes as input two partial functions f and g and outputs their composition $g \circ f$, i.e. $(f; g) = g \circ f$. The operation $(p \rightarrow f, g) = f$ and if $p = 1$ then $(p \rightarrow f, g) = g$. Since p is a constant the operation $(p \rightarrow —, —)$ is a binary operation from $P(D^{\ell M}) \times P(D^{\ell M})$ to $P(D^{\ell M})$. Let $f, g \in P(D^{\ell M})$. We say that $f \leqslant g$ if and only if, for all $m \in \ell M$, if $f(m)$ is defined then $g(m)$ is defined and $g(m) = f(m)$.

The relation $\leqslant$ is a partial order on $P(D^{\ell M})$. It has a minimal element (the function nowhwere defined) and it has greater lower bounds (glb's) for all subsets. It also has least upper bounds (lub's) for chains. We can also show that the operations $(\ ;\ )$ and $(p \rightarrow —, —)$ preserve lub's of chains. We can show by induction that any expression $\tau(X)$ obtained from constant functions in $P(D^{\ell M})$, the variable X and the operations $(\ ;\ )$ and $(p \rightarrow —, —)$ preserves lub's of chains. Since $P(D^{\ell M})$ has lub's of chains and it has a minimal element, the transformation $X \rightarrow \tau(X)$ has fixed points and in particular a minimal fixed point. This theorem attributed to Tarski occurs in Scott [11]. A generalization of it using category theory can be found in M. Wand [12]. A proof similar to D. Scott's [11] can be found in Appendix A.

We can now define the interpretation of terms of the form $\mu X[\tau]$.

$$I(\mu X[\tau])(s) = glb\{ f \mid f \in P(D^{\ell M}) \,\&\, I(\tau) \langle D, F, T, V/X := f \rangle = f \}.$$

That is we interpret all terms in τ except X. The result is a function

$$I(\tau)(s) : P(D^{\ell M}) \to P(D^{\ell M}),$$

X being a variable. This function satisfies Tarski's theorem and it has a minimal fixed point. We interpret $\mu X[\tau]$ as the minimal fixed point of this transformation. That is, a procedure is the minimal fixed point of the transformation $f \to I(\tau)(s/X := f)$. We make the convention that whenever I refers to term t then $I(\tau)$ stands for $I_t(\tau)$ and when I refers to a formula A then $I(A)$ stands for $I_f(A)$.

2. *Interpretation of Atomic Formulae :*

2.1. $I_f((\tau_1 \leqslant \tau_2))$ is true iff $I_t(\tau_1) \leqslant I_t(\tau_2)$ i.e. if for all $\psi \in D^{\ell M}$, $I_t(\tau_1)(\psi) = \psi^1$ implies $I_t(\tau_2)(\psi) = \psi^1$. This means that if $I_t(\tau_1)$ is defined for a state ψ then $I_t(\tau_2)$ is also defined and $I_t(\tau_1)(\psi) = I_t(\tau_2)(\psi)$.

2.2. $I_t((\tau_1 = \tau_2))$ is true iff $I_t(\tau_1)(\psi) = \psi^1$ implies $I_t(\tau_2)(\psi) = \psi^1$ and $I_t(\tau_2)(\psi) = \psi^1$ implies $I_t(\tau_1)(\psi) = \psi^1$.

3. *Interpretation of formulae :*

A list $I_f(\Phi) = I_f(\Phi_1, ..., \Phi_n)$ is true iff each $I_f(\phi_i)$ is true, $1 \leqslant i \leqslant n$; if ϕ is the empty list of formulae, $I_f(\phi)$ is false.

3. THE AXIOMS AND RULES OF THE EXTENDED μ-CALCULUS

The axioms and rules are given below :

I. *Composition Axioms :*

1. $\vdash \Omega; X = \Omega \vdash X; \Omega = \Omega$.
2. $\vdash X; (Y; Z) = (X; Y); Z$.
3. $\vdash E; X = X \vdash X; E = X$.

II. *Ordering Axioms :*

4. $\vdash X \leqslant X$.
5. $\vdash \Omega \leqslant X$.
6. $X \leqslant Y, Y \leqslant X \vdash X = Y$.
7. $X \leqslant Y, Y \leqslant Z \vdash X \leqslant Z$.
8. $X \leqslant Y \vdash \tau \leqslant \tau[Y/X]$ (monotonicity axiom).

Here $\tau[Y/X]$ denotes the term τ in which Y was substituted for X.

III. *Branching Axioms :*

In the next five axioms p and q stand for predicates $p(i, n)(x_1, ..., x_n)$ where $p(i, n) \in 1\ P$, $i, n \in \omega$ and $\langle x_1, ..., x_n \rangle$ is an n-tuple of memory variables.

9. $\vdash (p \to X, X) = X$.
10. $\vdash (p \to X, (p \to Y, Z)) = (p \to X, Z)$.
11. $\vdash (p \to (p \to X, Y), Z) = (p \to X, Z)$.
12. $\vdash (p \to (q \to X, Y), (q \to U, V)) = (q \to (p \to X, U), (p \to Y, V))$.
13. $\vdash (p \to X, Y); Z = (p \to X; Z, Y; Z)$.

IV. *The other axioms and rules are :*

14. $\vdash \tau[\mu X[\tau]/X] \leqslant \mu X[\tau]$ (the fixed point axiom).
15. (The μ-induction rule)

$$\psi \vdash \Phi[\Omega/X]$$
$$\psi, \Phi \vdash \Phi[\tau/X]$$
$$\psi \vdash \Phi[\mu X[\tau]/X]$$

provided that X does not occur free in ψ.

Before introducing the next three axioms some essential concepts are to be defined. Let $\tau = x \leftarrow f(i, n)(x_1, x_2, ..., x_n)$. The set of *input variables* of τ is $i(\tau) = \{ x_1, ..., x_n \}$ and the set of *output variables of* τ is $o(\tau) = \{ x \}$. The *input set* for a given predicate $p = p(i, n)(x_1, x_2, ..., x_n)$ is $i(p) = \{ x_1, x_2, ..., x_n \}$. Also $i(\Omega) = o(\Omega) = \phi$, $i(E) = o(E) = \phi$ and for any $X \in \mathbb{X}$, $i(X) = o(X) = \ell\ M$.

16. $\vdash A; B \leqslant B$; A provided that A, B are assignments and

$$i(A) \cap o(B) = i(B) \cap o(A) = o(A) \cap o(B) = \phi .$$

17. $\vdash A; B = B$ provided that A, B are assignments,

$$o(A) \subseteq o(B) \quad \text{and} \quad o(A) \cap i(B) = \phi .$$

18. $\vdash A; (p \to X, Y) = (p \to A; X, A; Y)$ provided that A is an assignment and $o(A) \cap i(p) = \phi$.
19. Substitution rule for memory variables : Let $b : {}^{\ell}M \to \ell\ M$ be a bijection, i.e. a one-to-one and onto function, Φ a formula and Φb the formula obtained from Φ by replacing each $m \in \ell\ M$ by $b(m)$. Then $\Phi \vdash \Phi b$.

The rules dealing with $\leqslant$ and $=$ are the same as in propositional calculus, where $\leqslant$ stands for logical implication and $=$ for equivalence. The substitution rule for procedure variables is the same as in first order predicate calculus. The validity proofs for the above axioms are presented in Appendix A.

Intuitively, the composition axioms state that the procedures form a monoid

with zero under sequencing. The ordering axioms assert that $\leqslant$ is a partial order with a minimal element; moreover it has the monotonicity property. The branching axioms state properties involving elimination of predicates, switching the order of two consecutive predicates and distributing ';' over branching.

Axiom 14 is used for proving the existence of the minimal fixed point and rule 15 allows proofs by induction. Axiom 16 states cases when sequencing of assignments is commutative, axiom 17 deals with the removal of useless instructions and axiom 18 states cases when ';' is left distributive over branching.

4. USEFUL THEOREMS OF THE EXTENDED μ-CALCULUS

An important result is that $\mu X[\tau]$ is the minimal fixed point of τ (J. W. De Bakker [3]). This result can be established as follows :

Part A : $\mu X[\tau]$ is a fixed point :

1. $\vdash \tau[\mu X[\tau]/X] \leqslant \mu X[\tau]$ by axiom 14.
2. $\vdash \Omega \leqslant \tau[\Omega/X]$ from axiom 1.
3. $X \leqslant \tau \vdash \tau \leqslant \tau[\tau/X]$ from axiom 8.
4. $\vdash \mu X[\tau] \leqslant \tau[\mu X[\tau]/X]$ from 2 and 3 by rule 15.
5. $\vdash \mu X[\tau] = \tau[\mu X[\tau]/X]$ from 1 and 4 by axiom 6.

Thus $\mu X[\tau]$ is a fixed point of τ.

Part B : $\mu X[\tau]$ is a minimal fixed point i.e. :

$Y = \tau[Y/X] \vdash \mu X[\tau] \leqslant Y$.

1. $Y = \tau[Y/X] \vdash \Omega \leqslant Y$ by tautology from axiom 5.
2. $Y = \tau[Y/X]$, $X \leqslant Y \vdash \tau \leqslant \tau[Y/X]$ by axiom 8.
3. $Y = \tau[Y/X] \vdash \mu X[\tau] \leqslant Y$ by the μ induction rule, from 1 and 2.

J. W. De Bakker [3] defined the " while " loop as follows

$$(D_1) \vdash p * A = \mu X[(p \rightarrow (A ; X), E)] .$$

Here p is a predicate (the repetition test) and A is the loop body.

To simplify the notation, '$*$' is assumed to have higher priority than ';'. Furthermore, from now on parenthesis around $(\tau_1 ; \tau_2)$ will be removed since ';' is associative (axiom 2). In most cases ';' will also be omitted. The following property of loops can be proven by using the formalism of the Extended μ-Calculus :

$$(T_2) \vdash p * A_1 ; A_2 = \mu X[(p \rightarrow A_1 X, A_2)] .$$

The proof of (T_2) reduces to showing that
(*) $\vdash p * A_1 ; A_2 \leqslant \mu X[(p \to A_1\ X, A_2)]$ and
(**) $\vdash \mu X[(p \to A_1\ X, A_2)] \leqslant p * A_1 ; A_2$ are theorems.
The proof of (*) is given below :

1. $\vdash \Omega A_2 = \Omega$ by axiom 1.
2. $\vdash \Omega \leqslant \mu X[(p \to A_1\ X, A_2)]$ by axiom 5.
3. $\vdash \Omega A_2 \leqslant \mu\ X[(p \to A_1\ X, A_2)]$ by tautology from 1 and 2.
4. $\vdash (p \to A_1\ X, E) ; A_2 = (p \to A_1\ XA_2, A_2)$ by axioms 13,3.
5. $XA_2 \leqslant \mu X[(p \to A_1\ X, A_2)] \vdash (p \to A_1\ X, E)\ A_2 \leqslant (p \to A_1\ \mu X[(p \to A_1\ X, A_2)], A_2)$ by axiom 8 from 4.
6. $\vdash (p \to A_1\ \mu X[(p \to A_1\ X, A_2)], A_2) \leqslant \mu X[(p \to A_1\ X, A_2)]$ by axiom 14.
7. $XA_2 \leqslant \mu X[(p \to A_1\ X, A_2)] \vdash (p \to A_1\ X, E)\ A_2 \leqslant \mu X[(p \to A_1\ X, A_2)]$ by tautology from 5, 6.
8. $\vdash p * A_1 ; A_2 \leqslant \mu X[(p \to A_1\ X, A_2)]$ by μ-induction from 1, 7.

Proof of (**).

1. $\vdash \Omega \leqslant p * A_1 ; A_2$ by axiom 5.
2. $X \leqslant p * A_1 ; A_2 \vdash (p \to A_1\ X, A_2) \leqslant (p \to A_1\ p * A_1 ; A_2, A_2)$ by axiom 8.
3. $\vdash (p \to A_1\ p * A_1 ; A_2, A_2) = (p \to A_1\ p * A_1, E) ; A_2$ by axiom 13.
4. $\vdash (p \to A_1 ; p * A_1, E) \leqslant p * A_1$ by axiom 14.
5. $X \leqslant p * A_1 ; A_2 \vdash (p \to A_1\ X, A_2) \leqslant p * A_1 ; A_2$ by tautology from 2, 3, 4.
6. $\vdash \mu X[(p \to A_1\ X, A_2)] \leqslant p * A_1 ;\ A_2$ by μ induction from 1, 6.

The logical connectives $'\wedge'$, $'\vee'$, $'\neg'$ are introduced by the following definitions :

$$(D_3) \vdash (p_1 \wedge p_2 \to X, Y) = (p_1 \to (p_2 \to X, Y), Y).$$
$$(D_4) \vdash (p_1 \vee p_2 \to X, Y) = (p_1 \to X, (p_2 \to X, Y)).$$
$$(D_5) \vdash (\neg p \to X, Y) = (p \to Y, X).$$

One can then formally prove that $'\wedge'$ and $'\vee'$ are commutative :

$$(T_6) \vdash (p_1 \wedge p_2 \to X, Y) = (p_2\ p_1 \to X, Y).$$
$$(T_7) \vdash (p_1 \vee p_2 \to X, Y) = (p_2 \vee p_1 \to X, Y).$$

Proof of (T_6) :

1. $\vdash (p_1 \wedge p_2 \to X, Y) = (p_1 \to (p_2 \to X, Y), Y)$ by (D_3).
2. $\vdash (p_2 \to Y, Y) = Y$ by axiom 8.
3. $\vdash (p_1 \to (p_2 \to X, Y), Y) = (p_1 \to (p_2 \to X, Y), (p_2 \to Y, Y))$ by tautology from 2.

4. $\vdash (p_1 \to (p_2 \to X, Y), (p_2 \to Y, Y)) = (p_2 \to (p_1 \to X, Y), (p_1 \to Y, Y))$ by axiom 12.
5. $\vdash (p_1 \to Y, Y) = Y$ by axiom 8.
6. $\vdash (p_2 \to (p_1 \to X, Y), (p_1 \to Y, Y)) = (p_2 \to (p_1 \to X, Y), Y)$ by tautology from 5.
7. $\vdash (p_2 \to (p_1 \to X, Y), Y) = (p_2 \wedge p_1 \to X, Y)$ by (D_3).
8. $\vdash (p_1 \wedge p_2 \to X, Y) = (p_2 \wedge p_1 \to X, Y)$ by tautology from 1, 3, 4, 6, 7.

The " While " loop described in (D_1) can be extended to include "$\wedge$", "$\vee$", and "$\neg$" constructions as follows :

$$(D_8) \vdash p_1 \wedge p_2 * A = \mu X[(p_1 \to (p_2 \to AX, E), E)] .$$
$$(D_9) \vdash p_1 \vee p_2 * A = \mu X[(p_1 \to AX, (p_2 \to AX, E))] .$$
$$(D_{10}) \vdash \neg p * A = \mu X[(p \to E, AX)] .$$

The following properties of " While " loops can be proven in this formalism.

$$(T_{11}) \quad Y \leqslant Y^1 \vdash \mu X[\tau] \leqslant \mu X[\tau][Y^1/Y] .$$
$$(T_{12}) \vdash p * A = (p \to A; p * A, E) .$$
$$(T_{13}) \vdash p * p * A = p * A .$$
$$(T_{14}) \vdash p * A = p * (A; p * A) .$$
$$(T_{15}) \vdash p * A_1; (p \to A_2, A_3) =$$
$$(T_{16}) \vdash p * E = (p \to \Omega, E) .$$

Formal proofs for theorems (T_{11})-(T_{16}) are given in Appendix B.

5. RESULTS CONCERNING MEMORY USE BY PROCEDURES

A *procedure schema* is a term in the language of the Extended μ-Calculus which has no free procedure variables. A *procedure* is a pair $P = (\tau, I)$ where τ is a procedure schema and I is an interpretation.

In this section the focus is on two points.

(1) Given a procedure schema τ we can define Mem (τ) as being the set of memory variables which occur in τ. We define $c(\tau)$ as the set of memory locations $m \subseteq$ Mem (τ) for which there is an interpretation I and an initial memory content $\psi \in D^{\ell M}$ such that $(I(\tau)(\psi))(m) \neq \psi(m)$ i.e. the content of m gets changed under at least one interpretation. We want to characterize the set $c(\tau)$.

(2) For a procedure $P = (\tau, I)$ the result of the computation is determined by the initial content of the memory location in Mem (τ). This so since the interpretation fixes the meaning of the predicates and functions and the

computation is deterministic. However, some memory locations in Mem (τ) may play only a passive role and their initial content does not affect the execution of the procedure. For example if $\tau = m_1 \leftarrow f(m_2, m_3); m_4 \leftarrow f(m_3, m_2)$; $(p(m_4) \rightarrow E; \Omega)$ the final content of the memory locations in Mem (τ) = $\{ m_1, m_2, m_3, m_4 \}$ is determined by the initial content of $\{ m_2, m_3 \}$ and the interpretation I. The initial content of the same set $\{ m_2, m_3 \}$ together with I determines if (τ, I) terminates or not. For a procedure schema τ we want to define a set $d(\tau)$ satisfying the following properties :

1. For any initial content of the memory $\psi : \ell M \rightarrow D$ and any interpretation I if we know the restriction of ψ to $d(\tau)$ then we know :

1.1. If $I(\tau)(\psi)$ is defined or not, and

1.2. If $I(\tau)(\psi)$ is defined then we know the final content of all memory variables in Mem (τ).

2. $d(\tau)$ is minimal.

We want to characterize the set $d(\tau)$.

The set of memory variables changed by a procedure $P = (\tau, I)$ can be defined as : $c(\tau, I) = \{ m \mid$ there is $\psi \in D^{\ell M}$ such that $I(\tau)(\psi)$ is defined and $I(\tau)(\psi)(m) \neq \psi(m) \}$. Of course $c(\tau) = \bigcup_I c(\tau, I)$.

Further on $A \restriction f$ will denote the restriction of the function f to subdomain A.

DEFINITION 3 : *Let $P = (\tau, I)$ be a procedure and A a subset of ℓM. A is called* significant *for P if for every ψ, $\phi \in D^{\ell M}$, $A \restriction \psi = A \restriction \phi$ implies $c(\tau, I) \restriction \tau(I)(\psi) = c(\tau, I) \restriction \tau(I)(\phi)$. The set of significant sets for P is written $D(\tau, I)$.*

The set of memory variables which determines P can now be defined as $d(\tau, I) = \bigcap_{S \in D(\tau, I)}$ i.e. it is the glb (greatest lower bound) of the set of significant sets. For a procedure schema τ, $d(\tau)$ can be defined as $d(\tau) = \bigcup_I d(\tau, I)$.

For a procedure schema τ one can define the set of input memory variables of τ, $i(\tau) = \{ m \mid m$ is input to a predicate or an assignment in $\tau \}$ and the set of output memory variables of τ, $o(\tau) = \{ m \mid m$ is the output variable of some assignment in $\tau \}$. The following propositions are helpful in finding $c(\tau)$ and $d(\tau)$ for a procedure schema τ.

PROPOSITION 1 : *If $\vdash \tau_1 = \tau_2$ then $c(\tau_1) = c(\tau_2)$ and $d(\tau_1) = d(\tau_2)$.*

PROPOSITION 2 : *For any procedure schema τ, $c(\tau) \subseteq o(\tau)$ and $d(\tau) \subseteq i(\tau)$.*

Proposition 1 states that the sets $c(\tau)$ and $d(\tau)$ are invariant under formal deduction (but not $o(\tau)$ and $i(\tau)$) and proposition 2 gives an upper bound for those sets.

The proof of these two propositions can be found in A. Pelin [8]. Unfortunately, there is no algorithm for finding the sets $c(\tau)$ and $d(\tau)$ for an arbitrary schema τ. The proof that the above problem is unsolvable is based on the fact that the equivalence problem for flowchart schemata is unsolvable (see Z. Manna [5]).

This in turn is used for showing that the Extended μ-Calculus is incomplete since the flowchart schemata can be simulated by terms in the language of the Extended μ-Calculus. The incompleteness in turn implies that there is no algorithm for computing $d(\tau)$ and $c(\tau)$. The details of the proof of the above stated results can be found in A. Pelin [8].

An important metatheorem is the one given below :

METATHEOREM 1 : *If A and B are procedure schemata (i.e. no free procedure variables) and if $o(A) \cap i(B) = o(B) \cap i(A) = o(A) \cap o(B) = \phi$ then $\vdash A; B = B; A$.*

The proof can be found in Appendix C. This metatheorem which states a case when ';' is commutative is very important if one considers the execution of A and B in parallel. Metatheorem 1 states that if A does not modify the input of B, B does not modify the input of A and if the output sets of A and B are disjoint then A and B commute. The condition that $o(A) \cap o(B) = \phi$ is essential since for $A = z \leftarrow f(1, 2)\ (x, y)$ and $B = z \leftarrow f(1, 2)\ (y, y)$ the conditions that $i(A) \cap o(B) = \phi$ and $i(B) \cap o(A) = \phi$ are satisfied but $A; B = B; A$ is not a tautology i.e. there are interpretations for which $I(A; B) \neq I(B; A)$.

Open Problem

It would be desirable for the system to be ale to give a formal proof of the fact that (*) $\vdash A; X; B = X; B$ where A, B are assignments, $o(A) \cap (i(X) \cup i(B)) = \phi$ and $o(A) \subseteq (B)$. This would be a generalisation of axiom 17. If $i(A) \cap o(X) = \phi$ then $\vdash A : X; B = X; B$ by using metatheorem 1. What axioms would have to be modified in order to have (*) ? Putting (*) as an axiom would be cheating since it would complicate the semantics and shift the proof to the metalanguage.

APPENDIX A

The consistency proofs for axioms and rules are simpler if some properties of the partial functions on a non-empty set are presented first. Let $P(S)$ denote the set of partial functions from S to S.

DÉFINITION 1 : *Let S be a non-empty set and f, g be two partial functions on S. $f \leqslant g$ iff for every $x \in X$ if $f(x)$ is defined then $g(x)$ is defined and $g(x) = f(x)$.*

LEMMA 1 : *$(P(S), \leqslant)$ is a partial order.*

Proof :

DÉFINITION 2 : *Let $(S, \leqslant)$ be a partial order. A chain in $(S, \leqslant)$ is a sequence $(s) = s_0 \leqslant s_1 \leqslant s_2 \leqslant \cdots \leqslant s_n \leqslant \cdots$ of elements in S.*

LEMMA 2 : *Let S be a set and $A \subseteq P(S)$. Then A has a lub in $(P(S), \leqslant)$.*

Proof : Define glb (A) as a partial function $g : S \to S$ as follows : for any $x \in S$, $g(x)$ is defined iff $f(x)$ is defined for all $f \in A$ and for all f, $h \in A$ if $f(x)$ and $h(x)$ are defined then $f(x) = h(x)$; otherwise $g(x)$ is undefined.

It is now easy to check that g is the lub for A. In particular $P(S)$ has a lub in $(P(S), \leqslant)$. This element is the nowhere defined function and it will be denoted by 0.

LEMMA 3 : *Let (f) be a chain in $(P(S), \leqslant)$. Then (f) has a lub.*

Proof : Let (f) be a chain in $(P(S), \leqslant)$. Define $g : S \to S$ as follows :

$$g(x) \quad \begin{cases} fi(x) & \text{if there is } i \in \omega \text{ such that } fi(x) \\ \text{undefined} & \text{is defined otherwise.} \end{cases}$$

Clearly g satisfies the conditions of definition 2.

LEMMA 4 : *Let $\bullet : P(S) \times P(S) \to P(S)$ be defined by $\bullet(f, g) = f.g$ where $\bullet$ is the operation of composition of partial functions (f is applied first). Then for any f, g, $h \in P(S)$.*

1. $0.f = 0$ *and* $0.f = 0$;
2. $f.(gh) = (f.g).h$;
3. $f.\mid S = f$ *and* $\mid S.f = f$.

where $\mid S$ is the identity function on S.

The proofs are very simple and are not presented here.

Let now I be any interpretation and let D be the domain of I. Any term τ is interpreted as a partial function $I(\tau) : D^{\ell M} \to D^{\ell M}$. By rule 5.1, $I(\Omega) = 0$ where 0 is the totally undefined function from $D^{\ell M}$ to $D^{\ell M}$. By rule 5.2, $I(E)$ is the identify function on $D^{\ell M}$. By rule 5.5 for any terms τ_1, τ_2, $I(\tau_1 ; \tau_2) = I(\tau_1).I(\tau_2)$. Thus axioms 1, 2, 3 are valid by lemma 4.

DÉFINITION 2 : *Let $(A, \leqslant_A)$ and $(B, \leqslant_B)$ be two partial orders and $f : A \to B$ be a total function. The function f is monotonic if for every a_1, $a_2 \in A$, $a_1 \leqslant_A a_2$ implies $f(a_1) \leqslant_B f(a_2)$.*

DÉFINITION 3 : *Let* $(A, \leqslant_A)$ *and* $(B, \leqslant_B)$ *be two partial orders with* glb's *for chains and* $f : A \to B$ *be a monotonic function. The function* f *is called* ω-*continuous if for any chain* (a), $\text{lub}(f((a))) = f(\text{lub}((a)))$ *i.e.* f *preserves* lub's *of chains. This means that for any chain* $a_0 \leqslant a_1 \leqslant \cdots \leqslant a_n \leqslant \cdots$ *in* A,

$$f(\text{lub}(\{ a_0, a_1, ..., a_n, ... \})) = \text{lub}(\{ f(a_0), f(a_1), ..., f(a_n), ... \}).$$

DÉFINITION 4 : *Let* $(A, \leqslant_A)$ *and* $(B, \leqslant_B)$ *be two partial orders. A new relation* $\leqslant_{A \times B}$ *can be defined on* $A \times B$, *where* $A \times B$ *is the cartesian product of* A *and* B *as follows :* $(a_1, b_1) \leqslant_{A \times B} (a_2, b_2)$ iff $a_1 \leqslant_A a_2$ *and* $b_1 \leqslant_B b_2$.

LEMMA 5 : *Let* $(A, \leqslant_A)$ *and* $(B, \leqslant_B)$ *be partial orders with* glb's *for any subsets and with* lub's *of chains. Then* $(A \times B, \leqslant_{A \times B})$ *is a partial order with* glb's *of any subsets and with* lub's *of chains.*

COROLLARY 6 : *For any* $D \neq \phi$, $(P(D^{\ell M}), \leqslant) \times (P(D^{\ell M}), \leqslant)$ *has* glb's *of subsets and* lub's *of chains.*

LEMMA 7 : *Let* $p : D^{\ell M} \to \{ 0, 1 \}$ *be a total predicate and let* $(p \to x, y) : P(D^{\ell M}) \times P(D^{\ell M}) \to P(D^{\ell M})$ *be the function defined as :*

$$(p \to x, y)(\psi) = \begin{cases} x(\psi) & \text{if } p(\psi) = 1 \\ y(\psi) & \text{if } p(\psi) = 0 . \end{cases}$$

Then $(p \to x, y)$ *is an* ω-*continuous function.*

Proof : It is clear that if $x_1 \leqslant x_2$ and $y_1 \leqslant y_2$ then $(p \to x_1, y_1) \leqslant (p \to x_2, y_2)$, i.e. $(p \to x, y)$ is monotonic. Let $(c) : (x_0, y_0) \leqslant (x_1, y_1) \leqslant \cdots \leqslant (x_n, y_n) \leqslant \cdots$ be a chain in $(P(D^{\ell M}) \times P(D^{\ell M}), \leqslant)$ and let $(a) : x_0 \leqslant x_1 \leqslant \cdots \leqslant x_n \leqslant \cdots$ and $(b) : y_0 \leqslant y_1 \leqslant \cdots \leqslant y_n \leqslant \cdots$ be the two chains obtained from (c) by projection. Let $x = \text{lub}((a))$ and $y = \text{lub}((b))$. Then $(p \to x, y)$ is an upper bound for the chain $p((c)) : (p \to x_0, y_0) \leqslant (p \to x_1, y_1) \leqslant \cdots \leqslant (p \to x_n, y_n) \leqslant \cdots$ since $x_n \leqslant x$ and $y_n \leqslant y$. If u is another upper bound for $p((c))$ then $u = (t, v)$ and $(p \to x_n, y_n) \leqslant (p \to t, v)$ for all $n \in \omega$. This in turn implies that t is an upper bound for (a) and v an upper bound for (b), hence $(x, y) \leqslant (t, v)$, i.e. $(p \to x, y)$ is the lub of the chain $p((c))$.

LEMMA 8 : *Let* D *be a non-empty set and* $\circledcirc : P(D^{\ell M}) \times P(D^{\ell M})$ *be the operation of composition of partial functions defined in lemma* 4. *Then* $\circledcirc$ *is an* ω-*continuous function.*

Proof : The proof is similar to theorem for lemma 7.

COROLLARY 9 : *Any function from* $(P(D^{\ell M}))^n \to P(D^{\ell M})$ *obtained by using the composition and branching operations is* ω-*continuous.*

For example, $(p \to (x, y), z) \circledcirc (q \to u, v) : (P(D^{\ell M}))^5 \to . P(D^{\ell M})$ *is* ω*-continous.*

Now the ordering axioms are obviously valid since every term τ is interpreted as a partial function from $D^{\ell M}$ to $D^{\ell M}$, i.e. $I(\tau) \in P(D^{\ell M})$ and $\leqslant$ is interpreted as the order on the partial functions in $P(D^{\ell M})$ given by definition 1. The axioms 9-13 are easy to prove by applying interpretation rules 1.5 and 1.6.

DÉFINITION 5 : *Let* $f : S \to S$ *be a function. An element* $x \in S$ *is called a fixed point of* f *if* $f(x) = x$.

LEMMA 10 (Tarski) : *Let* $(S, \leqslant)$ *be a partial order with* glb's *for subsets and* lub's *of chains and let* $f : S \to S$ *be an* ω*-continuous function. Then* f *has a minimal fixed point.*

Proof : Since $(S, \leqslant)$ has glb's for any subset $A \subseteq S$ it must have a minimal element (set $A = s$). Let 0 be the minimal element in $(S, \leqslant)$. The chain (a) is defined recursively as follows : $a_0 = 0$, an $f(a(n-1))$ for $n \geqslant 1$. Since f is ω-continuous, it is monotonic, thus $0 \leqslant f(0) \leqslant f(f(0)) \leqslant \cdots$ i.e., $a \leqslant a_1 \leqslant a_2 \leqslant \cdots$. This is so because $a_0 \leqslant a_1$ since a_0 is minimal and then one can show that $f(a_n) \leqslant f(a(n+1))$ by induction, keeping in mind that f is monotonic. Let $x = \text{lub}\,((a))$. Thus, for any $n \in \omega$, an $\leqslant x$. Since f is ω-continuous $f(x) = \text{lub}\,(f((a)))$. But

$$\text{lub}\,(f((a))) = \text{lub}\,(\{ f(a_0), f(a_1), \ldots \}) = \text{lub}\,(\{ a_1, a_2, \ldots \}) = x\,.$$

Thus x is a fixed point of f. In order to show that x is the least fixed point one uses mathematical induction to prove that for any fixed point y of f *an* $\leqslant y$. $0 \leqslant y$ since 0 is the minimal element in $(S, \leqslant)$. If *an* $\leqslant y$ then

$$a(n+1) = f(a_n) \leqslant f(y) = y$$

since f is monotonic and y is a fixed point. Thus y is an upper bound for the chain (a). Since x is the lub of (a), $x \leqslant y$, i.e. x is the minimal fixed point.

COROLLARY 11 : *Let* τ *be a term in the language of the Extended* μ*-Calculus, I an interpretation with domain D and X a free procedure variable. Then*

$$I(\tau)(X) : P(D^{\ell M}) \to P(D^{\ell M})\,,$$

where $I(\tau)(X)$ *is obtained by replacing X in* $I(\tau)$ *by a partial function in* $P(D^{\ell M})$, *has a minimal fixed point.*

Proof : $I(\tau)$ is ω-continuous by corollary 9. By lemma 10, $I(\tau)$ has a minimal fixed point.

Axiom 14 states one half of the fixed point property : $I(\mu \times [\tau])$ is the minimum fixed point of the transformation $I(\tau) : P(D^{\ell M}) \to P(D^{\ell M})$ obtained by replacing X in $I(\tau)$. It states that replacing X by the minimal fixed point of the transformation $I(\tau)$ yields a partial function less than or equal to the least fixed point.

Rule 15 is very important because it provides an inductive proof for certain properties. It parallels the construction of the minimal fixed point in lemma 10.

Axiom 16 is obvious.

Axiom 17 is a little different. It states that useless instructions may be removed. Its proof is obvious.

APPENDIX B

Formal proof of (T_{11}) : (monotonicity of μ).

1. $Y \leqslant Y \vdash \Omega \leqslant \mu X[\tau][Y^1/Y]$ by tautology from axiom 5.
2. $Y \leqslant Y^1, X \leqslant \mu X[\tau][Y^1/Y] \vdash \tau \leqslant \tau[\mu X[\tau][Y^1/Y]/X]$ by the monotonicity of τ in X.
3. $Y \leqslant Y^1, X \leqslant \mu X[\tau][Y^1/Y] \vdash \tau[\mu X[\tau][Y^1/Y]/X] \leqslant \tau[Y^1/Y][\mu X[\tau][Y^1/Y]/X]$ by the monotonicity of τ in Y.
4. $\vdash \tau[Y^1/Y][\mu X[\tau][Y^1/Y]/X] = \tau[Y^1/Y][\mu X[\tau[Y^1/Y]]/X]$.
5. $\vdash \tau[Y^1/Y][\mu X[\tau][Y^1/Y]]/X] \leqslant \mu X[\tau][Y/Y^1]$ by axiom 14.
6. $Y \leqslant Y^1, X \leqslant \mu X[\tau][Y^1/Y] \vdash \tau \leqslant \mu X[\tau][Y^1/Y]$ by tautology from 2, 3, 4, 5.
7. $Y \leqslant Y^1 \vdash \mu X[\tau] \leqslant \mu X[\tau][Y^1/Y]$ by the μ-induction rule from 1, 6.

Proof of (T_{12}) :

1. $\vdash p * A = \mu X[(p \to AX, E)]$ by (D_1).
2. $\vdash \mu X[(p \to AX, E)] = (p \to A\mu X[(p \to AX, E)], E)$ by Example 1, where $\tau = (p \to AX, E)$.
3. $\vdash (p \to A\mu X[(p \to AX, E)], E) = (p \to A; p * A, E)$ by tautology.
4. $\vdash p * A = (p \to A; p * A, E)$ by tautology from 1, 2, 3.

Proof of (T_{13}) :

1. $\vdash p * p * A = (p \to p * A; p * p * A, E)$ by (T_{12}).
2. $\vdash p * A; p * p * A = \mu X[(p \to AX, E)]; p * p * A$ by the definition of $*$.
3. $\vdash \mu X[(p \to AX, E)]; p * p * A = \mu X[(p \to AX, p * p * A)]$ by (T_2).

4. $\vdash \mu X[(p \to AX, E)]$;

$$p * p * A = \mu X[(p \to AX,\ (p \to p * A; p * p * A, E))]$$

by (T_{12}).

5. $\vdash \mu X[(p \to AX, (p \to p * A; p * p * A, E))] = \mu X[(p \to AX, E)]$ by axiom 10 and the monotonicity of μX.
6. $\vdash \mu X[(p \to AX, E)] = p * A$ by definition (D_1).
7. $\vdash p * A = (p \to A; p * A, E)$ by theorem (T_{12}).
8. $\vdash p * p * A = (p \to (p \to A; p * A, E), E)$ by tautology from 1, 2, 3, 4, 5, 6, 7.
9. $\vdash (p \to (p \to A; p * A, E), E) = (p \to A; p * A, E)$ by axiom 11.
10. $\vdash p * p * A = (p \to A; p * A, E)$ by tautology from 8, 9.
11. $\vdash p * p * A = p * A$ by tautology from 10, 7.

Proof of (T_{14}) :

1. $\vdash p * (A; p * A) = (p \to (A; p * A); p * (A; p * A), E)$ by theorem (T_{12}).
2. $\vdash (A; p * A); p * (A; p * A) = A; \mu X[(p \to AX, E)]; p * (A; p * A)$ by definition (D_1).
3. $\vdash A; \mu x[(p \to AX, E)]; p * (A; p * A) = A; \mu X[(p \to AX, p * (A; p * A))]$ by theorem (T_2).
4. $\vdash \mu X[(p \to AX, p * (A; p * A)] = \mu X[(p \to AX, (p \to A; p * A; p * (A; p * A), E))]$ by theorem (T_{12}) and the monotonicity of μ.
5. $\vdash \mu X[(p \to AX, (p \to A; p * A; p * (A; p * A), E))] = \mu X[(p \to AX, E)]$ by axiom 11 and the monotonicity of μ from 4.
6. $\vdash \mu X[(p \to AX, p * (A; p * A)] = p * A$ by tautology from 1, 5.
7. $\vdash p * (A; p * A) = (p \to A; p * A, E)$ by tautology from 1, 2, 3, 6.
8. $\vdash (p \to A; p * A, E) = p * A$ by theorem (T_{12}).
9. $\vdash p * (A; p * A) = p * A$ by tautology, 7, 8.

Proof of (T_{15}) :

1. $\vdash p \to A_1; (p \to A_2, A_3) = \mu X[(p \to A_1\ X, (p \to A_2, A_3))]$ by theorem (T_2).
2. $\vdash \mu X[(p \to A_1\ X. (p \to A_2, A_3))] = \mu X[(p \to A_1\ X, A_3)]$ by axiom 10 and the monotonicity of μ.
3. $\vdash p * A_1; A_3 = \mu X[(p \to A_1\ X, A_3)]$ by theorem (T_2).
4. $\vdash p \to A_1; (p \to A_2, A_3) = p * A_1; A_3$ by tautology from 1, 2, 3.

Proof of (T_{16}) :

1. $\vdash p * E = (p \to E; p * E, E)$ by theorem (T_{12}).
2. $\vdash \Omega \leqslant E; p * E$ by axiom 5.

3. $\vdash (p \to \Omega, E) \leqslant (p \to E; p * E, E)$ by axiom 8 from 2.
4. $\vdash (p \to \Omega, E) \leqslant p * E$ by tautology from 3.
5. $\vdash \Omega \leqslant (p \to E, \Omega)$ by axiom 5.
6. $X \leqslant (p \to E, \Omega) \vdash E; X \leqslant (p \to E, \Omega)$ by axiom 3.
7. $X \leqslant (p \to E, \Omega) \vdash (p \to E; X, E) \leqslant (p \to (p \to E, \Omega), E)$ by axiom 8.
8. $\vdash (p \to (p \to E, \Omega), E) = (p \to E, \Omega)$ by axiom 10.
9. $X \leqslant (p \to E, \Omega) \vdash (p \to EX, \Omega) \leqslant (p \to E, \Omega)$.
10. $\vdash * E \leqslant (p \to E, \Omega)$ by the μ induction rule from 5 and 9.
11. $\vdash * E(p \to E, \Omega)$ by axiom 6 from 3 and 10.

APPENDIX C

For a schema A with no free procedure variables, $i(A)$ was defined to be the set of all input variables to the assignments and predicates in A and $o(A)$ was defined to be the set of all output variables of the assignments in A. From the definition it follows that $i(\Omega) = i(E) = o(\Omega) = o(E) = \phi$ since no assignments or predicates occur in E or Ω.

For a free predicate variable X, $i(X) = o(X) = \ell M$ since X can affect any memory variables and it also can use any memory variables. The following lemma is very useful in checking commutativity of programs :

LEMMA 1 : *If B is an assignment, A is a procedure schema,*

$$i(B) \cap o(A) = i(A) \cap o(B) = o(A) \cap o(B) = \phi$$

then $\vdash A ; B = B ; A$.

Proof : The proof is by induction on the height (complexity) of the term A.

(Basis)

Case 1 : A is E or $A = \Omega$. Then $\vdash A; B = B; A$ by axioms 3 and 1 respectively.

Case 2 : A is another assignment. Then by applying axiom 16 twice one obtains $A; B = B; A$.

(Inductive Step)

Assume that $A : B = B; A$ is true for all terms S having height less than n satisfying the condition

$$i(A) \cap o(B) = i(B) \cap o(A) = o(A) \cap o(B) = \phi .$$

Let A have height n. There are three cases.

Case 1 : $A = C; D$ where C and D have height less than n. Of course

$i(C) \subseteq i(A)$, $i(D) \subseteq i(A)$, $o(D) \subseteq o(A)$, $o(D) \subseteq o(A)$, thus C and D satisfy the conditions of the lemma. By the induction hypothesis, $\vdash C; B = B; C$ and $\vdash D; B = B; D$. Below is a proof for $\vdash A; B = B; A$.

1. $\vdash A; B = (C; D); B$ by assumption.
2. $\vdash (C; D); B = C; (D; B)$ by axiom 2.
3. $\vdash D; B = B; D$ by the induction hypothesis.
4. $\vdash C; (D; B) = C; (B; D)$ by substitution from 3.
5. $\vdash C; (B; D) = (C; B); D$ by axiom 2.
6. $\vdash C; B = B; C$ by induction hypothesis.
7. $\vdash (C; B); D = (B; C); D$ by substitution from 6.
8. $\vdash (B; C); D = B; (C; D)$ by axiom 2.
9. $\vdash B; (C; D) = B; A$.
10. $\vdash A; B = B; A$ by tautology from 1, 2, 4, 5, 7, 8, 9.

Case 2 : $A = (p \rightarrow C, D)$ where C and D have height less than n. Again $i(C) \subseteq i(A)$, $i(D) \subseteq i(A)$, $o(C) \subseteq o(A)$ and $o(D) \subseteq o(A)$ thus C and D satisfy the conditions of the lemma. By the induction hypothesis, $\vdash C; B = B; C$ and $\vdash D; B = B; D$. Below is a proof for $\vdash A; B = B; A$.

1. $\vdash A; B = (p \rightarrow C, D); B$ by assumption.
2. $\vdash (p \rightarrow C, D); B = (p \rightarrow C; B, D; B)$ by axiom 13.
3. $\vdash C; B = B; C$ by induction hypothesis.
4. $\vdash D; B = B; D$ by induction hypothesis.
5. $\vdash (p \rightarrow C; B, D; B) = (p \rightarrow B; C, B; D)$ by substitution, using 3 and 4.
6. $\vdash (p \rightarrow B; C, B; D) = B; (p \rightarrow C, D)$ by axiom 18.
7. $\vdash B; (p \rightarrow C, D) = B; A$.
8. $\vdash A; B = B; A$ by tautology from 1, 2, 5, 6, 7.

Case 3 : $A = \mu X[C]$ where C has no other free procedure variables except X. The proof that $\mu X[C]; B = B; \mu X[C]$ is by μ-induction.

1. $\vdash C[\Omega/X]; B = B; C[\Omega/X]$ by induction hypothesis.
2. $X; B = B; X \vdash C; B = B; C$ since there are no free procedure variables other than X.
3. $\vdash \mu X[C]; B = B; \mu X[C]$ by μ induction.

Lemma 2 is a generalization of axiom 18.

LEMMA 2 : *If p is a predicate, X and Y are schemata, A is a procedure schemata and $i(p) \cap o(A) = \phi$ then $\vdash A; (p \rightarrow X, Y) = (p \rightarrow A; X, A; Y)$.*

Proof : By induction on the height of A.

(Basis)

Case 1 : $A = \Omega$ or $A = E$. Then the result holds by axiom 1, respectively axiom 3.

Case 2 : A is an assignment. Then $\vdash A; (p \to X, Y) = (p \to A; X, A; Y)$ is an instance of axiom 18.

(Inductive Step)

Assume the lemma to be true for all terms A of height $< n$. Let A have height n.

Case 1 : $A = C; D$. Then both C and D satisfy the conditions of the lemma and have height less than n.

Below is a proof that the lemma holds for A.

1. $\vdash A; (p \to X, Y) = (C; D)(p \to X, Y)$ by assumption.
2. $\vdash (C; D)(p \to X, Y) = C; (D; (p \to X, Y))$ by axiom 2.
3. $\vdash D; (p \to X, Y) = (p \to D; X, D; Y)$ by induction hypothesis.
4. $\vdash C; (D; (p \to X, Y)) = C; (p \to D; X, D; Y)$ by substitution, using 3.
5. $\vdash C; (p \to D; X, D; Y) = (p \to C; (D; X), C; (D; Y))$ by induction hypothesis.
6. $\vdash (p \to C; (D; X), C : (D; Y)) = (p \to (C; D) X, (C; D) Y)$ by substitution and axiom 2.
7. $\vdash (p \to (C; D) X, (C; D) Y) = (p \to AX, AY)$.
8. $\vdash A; (p \to X, Y) = (p \to AX, AY)$ by tautology from 1, 2, 4, 5, 6, 7.

Case 2 : $A = (q \to C, D)$. Then C and D have height $< n$ thus the induction hypothesis applies to them. A formal proof of $A; (p \to (q \to AX, AY))$ is given below :

1. $\vdash (p \to A; X, A; Y) = (p \to (q \to C, D) X, (q \to C, D) Y)$ by assumption.
2. $\vdash (p \to (q \to C, D) X, (q \to C, D) Y)) = (p \to (q \to CX, DX)(q \to CY, DY))$ by applying axiom 13 twice.
3. $\vdash (p \to (q \to CX, DX), (q \to CY, DY)) = (q \to (p \to CX, CY), (p \to DX, DY))$ by axiom 12.
4. $\vdash (q \to (p \to CX, CY), (p \to DX, DY)) = (q \to C(p \to X, Y), D(p \to X, Y))$ by applying the induction hypothesis to C and D.
5. $\vdash (q \to C(p \to X, Y), D(p \to X, Y)) = (q \to C, D)(p \to X, Y)$ by tautology and axiom 13.
6. $\vdash (q \to C, D)(p \to X, Y) = A; (p \to X, Y)$ by assumption.
7. $\vdash A; (p \to X, Y) = (p \to A; X, A; Y)$ by tautology from 1, 2, 3, 4, 5, 6.

Case 3 : $A = \mu Z[C]$ where C has height $< n$ and contains no free procedure variables except Z. Below is a proof that lemma holds for A :

1. $\vdash C\Omega/Z]; (p \to X, Y) = (p \to C[\Omega/Z] X, C[\Omega/Z] Y)$ by the induction hypothesis.

2. $Z; (p \to X, Y) = (p \to ZX, ZY) \vdash C; (p \to X, Y) = (p \to CX, CY)$ is a theorem since C contains no other free variables except Z.
3. $\vdash A; (p \to X, Y) = (p \to AX, AY)$ by μ induction from 1, 2.

METATHEOREM 1 : *If A and B are procedure schemata,*

$$i(A) \cap o(B) = o(A) \cap i(B) = o(A) \cap o(B) = \phi$$

then $\vdash A ; B = B ; A$.

Proof : By induction on the height of B.

(Basis)

Case 1 : $B = \Omega$ or $B = E$. Then $\vdash A; B = B; A$ by axiom 1, respectively axiom 3.

Case 2 : B is an assignment. Then $\vdash A; B = B; A$ by lemma 1.

(Inductive Step)

Assume that for A and B satisfying the conditions of the theorem and height $(B) > n$ the metatheorem is satisfied. Let B have height n.

Case 1 : $B = C; D$. Then one can establish $\vdash A; B = B; C$ by using a proof similar to the one used in Case 1 of the inductive step in lemma 1.

Case 2 : $B = (p \to C, D)$. Then $i(p) \subseteq i(B) \cap o(A) = \phi$ and $\vdash A;$

$$(p \to C, D) = (p \to AC, AD)$$

by lemma 2. The proof that $A; B = B; A$ is a theorem is given below :

1. $\vdash A; B = A; (p \to C, D)$ by assumption.
2. $\vdash A; (p \to C, D) = (p \to A; C, A; D)$ by lemma 2.
3. $\vdash (p \to A; C, A; D) = (p \to C; A, D; A)$ by induction hypothesis.
4. $\vdash (p \to C; A, D; A) = (p \to C, D); A$ by axiom 13.
5. $\vdash (p \to C, D); A = B : A$.
6. $\vdash A; B = B; A$ by tautology from 1, 2, 3, 4, 5.

Case 3 : $B = \mu X[C]$ where C has no free procedure variables except maybe X. The proof for this case is similar to the proofs for cases 3 of the inductive steps done in lemmata 1 and 2.

ACKNOWLEDGMENTS

The author would like to thank the referee for many valuable suggestions.

BIBLIOGRAPHIE

[1] R. S. BIRD, *Notes on Recursion Elimination,* CACM, Vol. 20, No. 6, 1977, pp. 434-pp. 439.

[2] R. S. BIRD, *Improving Programs by the Introduction of Recursion,* CACM, Vol. 20, No. 11, 1977, 856-863.

[3] J. W. DE BAKKER, *Recursive Procedures,* Mathematical Centre Tracts 24, Amsterdam, 1971.

[4] E. HOROWITZ and S. SAHNI, *Fundamentals of Computer Algorithms,* Computer Science Press, 1978.

[5] Z. MANNA, *Mathematical Theory of Computation,* McGraw-Hill, 1974.

[6] J. MCCARTHY, *Towards a Mathematical Science of Computation,* Information Processing, Proceedings of IFIP Congress 1962, pp. 21-28, North Holland Publishing Co., Amsterdam.

[7] E. MENDELSON, *Introduction to Mathematical Logic,* D. Van Nostrand, 1964.

[8] A. PELIN, *An Extended Verson of De Bakker's μ Calculus,* Ph. D. Dissertation, 1977.

[9] D. SCOTT, *The Lattice of Flow Diagrams,* Symposium on Semantics of Algorithmic Languages, Springer Verlag, 1977.

[10] D. SCOTT, *Continuous Lattices,* Oxford University Computing Laboratory Technical Monograph PRG 7, 1971.

[11] D. SCOTT, *Data Types as Lattices,* unpublished notes, Amsterdam, 1972.

[12] M. WAND, *Fixed-Point Constructions in Order-Enriched Categories,* Indiana University Computer Science Department Technical Report, No. 23, 1975.