Notre Dame Journal of Formal Logic
Volume XVI, Number 4, October 1975
NDJFAM

THE PROGRAMMATIC SEMANTICS OF BINARY PREDICATOR CALCULI

JORGE BARALT-TORRIJOS, LUCIO CHIARAVIGLIO, and WILLIAM GROSKY

One may take the view that combinatory logic is concerned with some families of calculi that share well studied morphological and transformational features. These calculi allow for a variety of interpretations. There has been some interest among those concerned with the theory of programming, in these calculi. The intuitive programming interpretations of some of these calculi generally view them as abstract theories of rules that govern problem solving by idealized computing facilities (see Goodman [2]). In this paper,* we wish to present a formal programmatic semantics for a selection of combinatory calculi that is based on first-order model theoretic considerations. There may be extensions of the methods here proposed to higher order and type theoretic considerations (see Curry, Hindley, and Seldin [1], and Sanchis [4, 5]). For the present, we shall limit ourselves to the first-order case.

In the following sections of the paper, we first review the morphology and transformational syntax of the calculi, we then construct a first-order referential semantics and conclude with a section that develops the programmatic semantics. The capstone of the paper is a semantic completeness theorem that exhibits how the semantic notion of programmatic equivalence is related to the syntactical concepts of reducibility and extensional equivalence.

1 *A family of first-order binary predicator languages* Each calculus **FC** of the family has the following morphology:

a) C is a set of *individual constants*;
b) V is a denumerably infinite set of *individual variables*;
c) $\equiv$ is a two-place predicator;
d) ap is a two-place functional sign;

*This work was supported in part by National Science Foundation Grant GN-655.

Received June 5, 1974

e) the set of terms, T, is the closure of $C \cup V$ under the operation of prefixing ap appropriately to two terms (i.e., $\mathsf{ap}(t_1, t_2)$);
f) the set of formulae of **FC** is obtained by infixing $\equiv$ between terms (i.e., $t_1 \equiv t_2$).

The transformational syntax of each **FC** is as follows:

a) there is a possibly empty set of formulae that are axioms;
b) there is a rule of substitution that states that if f is a theorem of **FC**, if $x_1, \ldots, x_n$ are n distinct variables, and if $t_1, \ldots, t_n$ are in T, then the formula obtained by simultaneous unary substitution of $t_1, \ldots, t_n$ for $x_1, \ldots, x_n$ in f is also a theorem (briefly, if $\vdash f$, then $\vdash [t_1/x_1, \ldots, t_n/x_n] f$);

also,

c) perhaps other rules of inference.

The simultaneous unary substitution of n terms for n distinct variables in terms and formulae is given the usual formulation and satisfies the common distribution laws with respect to the functional sign and predicator (i.e., if t and t^* are in T, then $[t_1/x_1, \ldots, t_n/x_n]\ \mathsf{ap}(t, t^*)$ is identical to $\mathsf{ap}([t_1/x_1, \ldots, t_n/x_n]\ t, [t_1/x_1, \ldots, t_n/x_n]\ t^*)$ and $[t_1/x_1, \ldots, t_n/x_n]\ (t \equiv t^*)$ is identical to $([t_1/x_1, \ldots, t_n/x_n]\ t) \equiv ([t_1/x_1, \ldots, t_n/x_n]\ t^*)$.

The term t_1 is said to be the *functional component* and t_2 the *argument component* of the term $\mathsf{ap}(t_1, t_2)$. The relation *is a component of* is transitive and reflexive on the set of terms. A term is said to be *closed* if it has no components that are variables. A calculus **FC** is said to have the *substitution property* if and only if, for any terms t and t^* with component variables $x_1, \ldots, x_n$, if $\vdash [t_1/x_1, \ldots, t_n/x_n]\ (t \equiv t^*)$ for all closed terms $t_1, \ldots, t_n$, then $\vdash t \equiv t^*$.

By a *first-order referential model* of **FC** we understand a pair $\mathfrak{M} = (D, IC)$ such that,

a) $D \neq \emptyset$;
b) $IC\colon C \to D$;
c) $IC(\mathsf{ap})\colon D \times D \to D$;
d) $IC(\equiv) \subseteq D \times D$;
e) for any term t and $d \in \mathsf{Assig}(\mathfrak{M}) = \{d \mid d\colon V \to D\}$, td is the interpretation of t relative to d;
f) the truth of a formula f in $\mathfrak{M}$, $\mathfrak{M} \vDash f$, and the notion of satisfaction of f in $\mathfrak{M}$ relative to $d \in \mathsf{Assig}(\mathfrak{M})$, $\mathfrak{M} \vDash [f, d]$, are given the usual explication;
g) if $\vdash f$, then $\mathfrak{M} \vDash f$. The set D is called the *carrier* and IC is called the *interpreting function of* $\mathfrak{M}$.

Every calculus **FC** is semantically complete. In order to show this, we may construct a free model $\mathfrak{M} = (T, IC)$, where,

a) T is the set of terms of **FC**;
b) IC is the identity on $C \cup V$;
c) $IC(\mathsf{ap})$ is the operation of prefixing ap appropriately to terms;

d) $IC(\equiv)$ is a relation on T that holds between any two terms t_1 and t_2 if and and only if $\vdash t_1 \equiv t_2$.

It is clear that all and only all the theorems of **FC** are truths of this model.

2 *The programmatic semantics of the family of calculi* We take the view that nothing may be considered to be a plan or program unless we can envisage goals that might be attained by the proper execution of such plans or programs. Similarly, we also take the view that the notion of commanding or governing a computing facility presupposes the notion of goals. Our strategy then is first to delineate what a goal language is; second, to identify what we may mean by processes relative to the interpretation of the goal language; and third, to identify among these processes those that may be governed by plans or programs.

A *goal language*, $\mathcal{GC}$, for **FC** has the following characteristics:

a) the morphology of $\mathcal{GC}$ includes the morphology of **FC**;
b) $\mathcal{GC}$ may contain further descriptive vocabulary such as connectives, quantifiers, etc., but we assume that its set of variables is V;
c) $\mathcal{GC}$ does not have a theory proper that characterizes its descriptive vocabulary;
d) $\mathcal{GC}$ has first-order realizations, $\mathfrak{M} = (D, IG)$, where D is the carrier and IG the interpreting function.

A possible realization of $\mathcal{GC}$ that is a model of **FC** is denoted by $\mathfrak{M}^*$. Any function s from the natural numbers into the set $\mathrm{Assig}(\mathfrak{M}^*)$ is said to be a *process* in $\mathfrak{M}^*$. Any pair $G = (G_1, G_2)$ of formulae of $\mathcal{GC}$ is a *goal*.

Definition 1 A process s is *bound by a goal* $G = (G_1, G_2)$ in $\mathfrak{M}^*$ if and only if, $\mathfrak{M}^* \models [G_1, s(0)]$ implies that there is an $n \geqslant 0$ such that for all $m \geqslant n$, $\mathfrak{M}^* \models [G_2, s(m)]$.

The intuitive idea behind the concept of process is that of an activity that modifies elements of the carrier of $\mathfrak{M}^*$ producing new elements of the carrier. In order to have these activities localized relative to the goal language, they are conceived as transformations of assignments into assignments. A goal may then be conceived as a pair of formulae of $\mathcal{GC}$, or, if we desire to generalize, as a sequence of formulae of $\mathcal{GC}$ such that if the first is satisfied by the initial state of the process, then the succeeding formulae are satisfied in their order after finite delays. If there is a final formulae in the sequence, then it must be satisfied in proper order after a finite delay and thereafter. The notion of a process being bound by a goal is our reconstruction of the intuitive idea of a process succeeding at attaining a goal.

Among the processes in some realization of $\mathcal{GC}$ that is also a model of **FC**, there may be some that may be said to be the execution of terms of **FC** by some appropriate executing facility. We may now turn to the question of executing facilities.

Let Pos be the set composed of the elements o and all finite strings of

a's and f's. The set Pos may be thought of as a universal set of positions for the components of any terms. A *construction function* for terms of any calculus is a partial function c: $\mathsf{Pos} \times T \to T$ such that, for any $t \in T$, $c(o, t)$ is t; $c(a, t)$ is the argument component of t; $c(f, t)$ is the functional component of t; $c(aa, t)$ is the argument component of the argument component of t; and so on, until all the components of t are exhausted.

In order to conceive of terms of **FC** as governing some process in a realization $\mathfrak{M}^* = (D, IG)$, we need the following,

a) a *representation function* R that maps D into the set of closed terms of **FC**;
b) a *syntactical processing agency* P that maps terms of *FC* into terms of **FC**;
c) an *allocating function* al that maps V into Pos.

Let t be a term of **FC** and $x_1, \ldots, x_m$ the variables that are components of t. The *process generated by t in $\mathfrak{M}^*$ with the initial assignment $d \in$* $\mathsf{Assig}(\mathfrak{M}^*)$ *relative to a representation R, agency P and allocating function* al is given by:

Definition 2 $\mathsf{Proc}(t, \mathfrak{M}^*, d, R, P, \mathsf{al})(0) = d$, and, for any $x \in V$ and $n \geqslant 0$:

a) $(\mathsf{Proc}(t, \mathfrak{M}^*, d, R, P, \mathsf{al})(n + 1))(x) = IG(c(\mathsf{al}(x), P^n([R(d(x_1))/x_1, \ldots, R(d(x_m))/x_m]\, t)))$ if c is defined at $\mathsf{al}(x)$ and $P^n([R(d(x_1))/x_1, \ldots, R(d(x_m))/x_m]\, t)$;

otherwise,

b) $(\mathsf{Proc}(t, \mathfrak{M}^*, d, R, P, \mathsf{al})(n + 1))(x) = (\mathsf{Proc}(t, \mathfrak{M}^*, d, R, P, \mathsf{al})(n))(x)$.

The intuitive content of this definition and the preceding remarks can be exhibited by the following diagram:

$$\begin{tikzcd}
d \arrow[d, "R"'] & d' & d'' \dots \\
t' \arrow[r, "P"'] \arrow[ur, phantom, "{IG, \mathsf{al}}"] & t'' \arrow[u] \arrow[r, "P"'] \arrow[ur, phantom, "{IG, \mathsf{al}}"] & t''' \dots \arrow[u]
\end{tikzcd}$$

From an initial assignment d and a term t the representation function R determines a closed term t'. The syntactical transformation P yields a new term t''. The allocation function al together with the interpreting function IG determine a new assignment d'. The sequence of assignments $d, d', d'', \ldots$ is the process in $\mathfrak{M}^*$ governed by t relative to R, P, and al.

Definition 3 Two terms t_1 and t_2 of **FC** are *programmatically equivalent relative to R, P, and* al, $t_1\, \mathsf{Equiv}(R, P, \mathsf{al})\, t_2$, if and only if, for every $\mathfrak{M}^*$, every $d \in \mathsf{Assig}(\mathfrak{M}^*)$, and every goal G of $\mathcal{GC}$, $\mathsf{Proc}(t_1, \mathfrak{M}^*, d, R, P, \mathsf{al})$ is bound by G in $\mathfrak{M}^*$ if and only if $\mathsf{Proc}(t_2, \mathfrak{M}^*, d, R, P, \mathsf{al})$ is bound by G in $\mathfrak{M}^*$.

Terms are *programmatically equivalent* if the processes they govern co-succeed in attaining goals of $\mathcal{GC}$ in every possible realization that is also a model of the calculus to which the terms belong. We say that a syntactical processing agency P is *admissible* for a calculus **FC** if and only if, for every n and term t of **FC**, $\vdash P^n(t) \equiv t$.

Theorem 1 *If* **FC** *has the substitution property,* P *is admissible and* $t_1 \, \mathsf{Equiv}(R, P, \mathsf{al}) \, t_2$ *for all* al *and* R, *then* $\vdash t_1 \equiv t_2$.

Proof: Assume the hypothesis of the theorem. Let $G = (G_1, G_2)$ be a goal such that G_1 is a truth of $\mathfrak{M}^*$ and G_2 is the formula $[R(d(x_1))/x_1, \ldots, R(d(x_n))/x_n] \, t_1 \equiv x$, where $\mathsf{al}(x) = o$, $x_1, \ldots, x_n$ are the component variables of t_1 and t_2, and $d \in \mathsf{Assig}(\mathfrak{M}^*)$. From these assumptions, we may conclude that $\mathfrak{M}^* \models [G_2, \mathsf{Proc}(t_1, d, R, P, \mathsf{al})(m+1)]$ for any m, since the following holds:

a) $(\mathsf{Proc}(t_1, \mathfrak{M}^*, d, R, P, \mathsf{al})(m+1))(x) = IG(c(o, P^m([R(d(x_1))/x_1, \ldots, R(d(x_n))/x_n] \, t_1))) = IG(P^m([R(d(x_1))/x_1, \ldots, R(d(x_n))/x_n] \, t_1))$;
b) the interpretation of $[R(d(x_1))/x_1, \ldots, R(d(x_n))/x_n] \, t_1$ relative to any assignment d is simply the IG of this term since the term is closed;
c) the pairs formed by $(\mathsf{Proc}(t_1, \mathfrak{M}^*, d, R, P, \mathsf{al})(m+1))(x)$ and $IG([R(d(x_1))/x_1, \ldots, R(d(x_n))/x_n] \, t_1)$ are in $IG(\equiv)$ for $m = 0$, and hence for $m > 0$, because P is admissible, and $\mathfrak{M}^*$ is a model of **FC**.

Thus, we may conclude that for all m greater than some $k \geqslant 0$, $\mathfrak{M}^* \models [G_2, \mathsf{Proc}(t_2, \mathfrak{M}^*, d, R, P, \mathsf{al})(m+1)]$. Using the assumption that P is admissible and **FC** is semantically complete, we conclude that $\vdash [R(d(x_1))/x_1, \ldots, R(d(x_n))/x_n] \, (t_1 \equiv t_2)$ for any d and R. We have assumed that **FC** has the substitution property, and hence $\vdash t_1 \equiv t_2$. QED

We may say that a syntactical agency P is a *conversion* for **FC** if, whenever $\vdash t_1 \equiv t_2$, then there are $m, n \geqslant 0$ such that $P^m(t_1) = P^n(t_2)$. The following theorem is evident.

Theorem 2 *If* P *is a conversion for* **FC** *and* $\vdash t_1 \equiv t_2$, *then, for every possible realization* $\mathfrak{M}$ *of* $\mathcal{GC}$ ($\mathfrak{M}$ *need not be a model of* **FC**), *every* $d \in \mathsf{Assig}(\mathfrak{M})$, *and every goal* G, $\mathsf{Proc}(t_1, d, R, P, \mathsf{al})$ *is bound by* G *in* $\mathfrak{M}$ *if and only if* $\mathsf{Proc}(t_2, \mathfrak{M}, d, R, P, \mathsf{al})$ *is bound by* G *in* $\mathfrak{M}$.

This theorem can be generalized to goal languages that do not include the morphology of **FC**.

3 *Conclusion* We feel that this approach to programmatic semantics and to the problems concerned with the equivalence of programs (terms) offers distinct advantages over previous formulations. The major advantage is that of the formulation of the notion of equivalence between programs (terms) which does not include the concept of termination. Another advantage is that we are working in a formal system whose proof theory, we feel, is much simpler than that of the lower predicate calculus (see Manna [3]).

REFERENCES

[1] Curry, H. B., J. R. Hindley, and J. P. Seldin, *Combinatory Logic*, Volume II, North Holland Publishing Company, Amsterdam (1972).

[2] Goodman, N. D., "A simplification of combinatory logic," *The Journal of Symbolic Logic*, vol. 37 (1972), pp. 225-246.

[3] Manna, Z., "Mathematical theory of partial correctness," *Journal of Computer and System Sciences*, vol. 5 (1971), pp. 239-253.

[4] Sanchis, L. E., *Normal combinations and the theory of types*, Ph.D. Thesis, Pennsylvania State University, 1963.

[5] Sanchis, L. E., "Types in combinatory logic," *Notre Dame Journal of Formal Logic*, vol. V (1964), pp. 161-180.

Georgia Institute of Technology
Atlanta, Georgia