Examples in Program Composition

K. Mani Chandy
Stephen Taylor

1 The notation

This paper describes a few examples in Program Composition Notation (PCN), a notation for composing programs. PCN runs on a variety of machines including sequential machines, message-passing multicomputers and shared-memory machines. PCN uses four primitive composition operators: sequential, parallel, choice and fair composition with the symbols ';', '||', '?', and '[]', respectively. PCN uses (mutable) variables as in imperative programming — variables whose value can change — and definition names (defnames, for short), which can be assigned at most once. The value of a definition name is either the special symbol undefined or an expression that does not refer to (mutable) variables. For example, the value of a definition name can be x, or x + 5, or {x, {y}, 5}, where x and y are definition names. (Note that a tuple is a form of expression.)

Initially, all defnames are undefined. An undefined defname becomes defined by executing a define-statement in which it appears on the left-hand side. Once defined, a defname never changes.

Proof Obligation for Definitions Programmers must prove that a defname is defined at most once and that definitions are not circular:

1. At most one define-statement is executed for each defname, and
2. At each point in the computation, there exists an ordering of definitions such that if x is defined as e, y is defined as f, and f refers to x, then the definition for x appears before the definition for y in the ordering.

Reduction A defname y reduces to an expression or tuple e at a point t in the computation if y is defined as e at t, or if y reduces to e' at t and e can be obtained from e' by substituting the definition for a defname in
e', or if e can be obtained from e' by simplifying terms that reference only constants (e.g. replace 2+3 in e' by 5 in e).

**Values of Expressions** The value of an expression is a special symbol, *unknown*, if the expression refers to a defname that does not reduce to a constant; otherwise the value is obtained by replacing each defname by the constant to which it reduces, replacing each variable by its value, and evaluating.

The value of an expression is said to be known if its value is not *unknown*. Once the value of an expression is known, it remains known forever after.

### 1.1 Programs

A program consists of a heading, a declaration section, and a block. The heading is the program name and a list of arguments, as in other notations. The heading and the declaration section are not discussed here.

Parameters are passed by reference. The scope of a variable is limited to the program in which it appears.

The syntax of a block is given below in BNF. All nonterminal symbols are in italics, and all terminal symbols are in plain type. The notation <su>, where su is a syntactic unit represents a list of zero or more instances su, separated by commas.

\[
\text{block ::= define-statement | assignment | program-call | }
\begin{align*}
\{ & \text{control-composition-operator } < \text{block} > \} | \\
& \{ \text{selection-composition-operator } < \text{guard } \rightarrow \text{block} > \} | \\
& \{ \text{user-defined-composition-operator } < \text{argument} > \}
\end{align*}
\]

**Define Statement** A define-statement has the form \( v \leftarrow \text{rhs} \) for \( v \) a defname and \( \text{rhs} \) an expression. Execution of the statement causes \( v \) to become defined as \( e \), where \( e \) is obtained from \( \text{rhs} \) by replacing all variables in \( \text{rhs} \) by their current values.

**Assignment** An assignment has the form \( v := \text{rhs} \) for \( v \) a mutable variable and \( \text{rhs} \) an expression. The assignment is executed by waiting until \( \text{rhs} \) becomes known and then assigning its value to \( v \):

\[
\text{while } \text{rhs is unknown do skip; } \\
\text{assign the value of } \text{rhs} \text{ to } v.
\]

**Program Call** A program call is like a procedure call in Pascal; the invoked program is initiated with its parameters replaced by the arguments; parameter passing is by reference.