A Toy Programming Language

In our toy programming language, all variables are integer variables, so that there is no need to declare them. Integer expressions are formed from variables and integer constants using the operators $+$, $-$, and $\times$. Boolean expressions are formed from equalities and inequalities between integer expressions using the propositional connectives $\land$, $\lor$, $\neg$. Assignments are expressions of the form $V \leftarrow IE$ where $V$ is a variable and $IE$ is an integer expression. Programs are formed from assignments using three constructs:

- sequential composition

$$P; Q$$

where $P$ and $Q$ are programs,

- conditional composition

$$\text{if } BE \text{ then } P \text{ else } Q \text{ endif}$$

where $BE$ is a Boolean expression and $P$, $Q$ are programs,

- loop

$$\text{while } BE \text{ do } P \text{ enddo}$$

where $BE$ is a Boolean expression and $P$ is a program.

For instance,

$$\text{if } m > n$$
$$\text{ then } p \leftarrow m$$
$$\text{ else } p \leftarrow n$$
$$\text{ endif}$$

is a program. We will denote it by $P1$. Another example:

$$p \leftarrow 1;$$
$$i \leftarrow 0;$$
$$\text{while } i < n \text{ do}$$
$$\quad i \leftarrow i + 1;$$
$$\quad p \leftarrow p \times 2$$
$$\text{ enddo}$$

We will call this program $P2$. 
Partial Correctness and Annotated Programs

An assertion is a condition that may be true or false depending on the values of variables. For example, every Boolean expression is an assertion. An assertion may include symbols that are not allowed in programs. For instance, the equalities \( p = \max(m, n) \) and \( p = 2^n \) are assertions, even though the expressions \( \max(m, n) \) and \( 2^n \) are not part of our programming language.

We will describe properties of programs using formulas of the form

\[
\{ A \} \; P \; \{ B \}
\]

where \( A \) and \( B \) are assertions and \( P \) is a program. Assertion \( A \) is called the precondition of formula (1), and \( B \) is called its postcondition. Formula (1) expresses that if the values of variables satisfy condition \( A \) then their new values after the execution of \( P \) will satisfy condition \( B \), provided that the execution terminates. For instance, the formula

\[
\{ m = 3 \} \; n \leftarrow m + 5 \; \{ m = 3 \land n = 8 \}
\]

is true. Formulas

\[
\{ m = 3 \land n = 5 \} \; P1 \; \{ p = 5 \}
\]

and

\[
\{ n = 5 \} \; P2 \; \{ p = 32 \}
\]

are true as well. On the other hand, formula

\[
\{ n = 5 \} \; P1 \; \{ p = 5 \}
\]

is false. (Counterexample: before the execution of \( P1, n = 5 \) and \( m = 6 \).)

If formula (1) is true, we say that program \( P \) is partially correct for the precondition \( A \) and postcondition \( B \).

An annotated program is a program with assertions, enclosed in braces, possibly added to its text at various points. For instance, every program is an annotated program (no assertions added). Every formula of the form (1) is an annotated program also (the precondition and postcondition are added). We define below rules for constructing “complete” annotated programs. If we can turn a formula (1) into a complete annotated program by inserting additional assertions between its precondition and postcondition then we can say that the formula is proved.

Substitution

The following notation will be useful. If \( A \) is a condition, \( V \) is a variable, and \( IE \) is an integer expression, then by \( A^V_{IE} \) we denote the condition obtained from \( A \) by substituting \( IE \) for \( V \). For instance,

\[
(m = 3 \land n = 8)^n_{m+5} \text{ is } m = 3 \land m + 5 = 8.
\]
If $V$ occurs in $A$ several times then $A_{IE}^V$ is obtained from $A$ by substituting $IE$ for all occurrences of $V$ simultaneously. For instance,

\[(m < n \land n = 8)_{n+1}^n \text{ is } m < n + 1 \land n + 1 = 8.\]

If $V$ doesn’t occur in $A$ at all then $A_{IE}^V$ is the same as $A$:

\[(m = 3)_{m+5}^n \text{ is } m = 3.\]

**Rule of Assignment**

Any annotated program of the form

\[
\{A_{IE}^V\} \\
V \leftarrow IE \\
\{A\}
\]

is complete.

For instance, the annotated program

\[
\{m = 3 \land m + 5 = 8\} \\
n \leftarrow m + 5 \\
\{m = 3 \land n = 8\}
\]

is complete.

**Rules of Consequence**

1. An annotated program of the form

\[
\{A\} \\
\{B\} \\
P
\]

is complete if $A$ implies $B$ and the annotated program

\[
\{B\} \\
P
\]

is complete.

2. An annotated program of the form

\[
P \\
\{A\} \\
\{B\}
\]

is complete if $A$ implies $B$ and the annotated program

\[
P \\
\{A\}
\]

\[3\]
is complete.

For instance, the annotated program

\[
\begin{align*}
\{ m = 3 \} \\
\{ m = 3 \land m + 5 = 8 \} \\
\text{n} \leftarrow m + 5 \\
\{ m = 3 \land n = 8 \}
\end{align*}
\]

is complete, because the assertion \( m = 3 \) implies \( m = 3 \land m + 5 = 8 \). This annotated program justifies formula (2).

**Rule of Composition**

*An annotated program of the form*

\[
\begin{align*}
P; \\
\{ A \} \\
Q
\end{align*}
\]

*is complete if each of the annotated programs*

\[
\begin{align*}
P \\
\{ A \}
\end{align*}
\]

*and*

\[
\begin{align*}
\{ A \} \\
Q
\end{align*}
\]

*is complete.*

For example, the annotated program

\[
\begin{align*}
\{ m = 2 \} \\
\{ m + 1 = 3 \} \\
\text{m} \leftarrow \text{m} + 1; \\
\{ m = 3 \} \\
\{ m = 3 \land m + 5 = 8 \} \\
\text{n} \leftarrow m + 5 \\
\{ m = 3 \land n = 8 \}
\end{align*}
\]

is complete. It justifies the formula

\[
\begin{align*}
\{ m = 2 \} \\
\text{m} \leftarrow \text{m} + 1; \\
\text{n} \leftarrow \text{m} + 5 \\
\{ m = 3 \land n = 8 \}.
\end{align*}
\]
Conditional Rule

An annotated program of the form

\[
\{ A \} \\
\text{if} \ BE \\
\text{then} \ \{ A \land BE \} \\
P \\
\{ B \} \\
\text{else} \ \{ A \land \neg BE \} \\
Q \\
\{ B \} \\
\text{endif} \\
\{ B \}
\]

is complete if the annotated programs

\[
\{ A \land BE \} \\
P \\
\{ B \}
\]

and

\[
\{ A \land \neg BE \} \\
Q \\
\{ B \}
\]

are complete.

For instance, the annotated program

\[
\{ m = 3 \land n = 5 \} \\
\text{if} \ m > n \\
\text{then} \ \{ m = 3 \land n = 5 \land m > n \} \\
\{ \text{false} \} \\
\{ m = 5 \} \\
p \leftarrow m \\
\{ p = 5 \} \\
\text{else} \ \{ m = 3 \land n = 5 \land m \leq n \} \\
\{ n = 5 \} \\
p \leftarrow n \\
\{ p = 5 \} \\
\text{endif} \\
\{ p = 5 \}
\]

is complete. It justifies formula (3).
While Rule

An annotated program of the form

\[
\begin{align*}
\{A\} \\
\text{while } BE \\
\quad \text{do } \{A \land BE\} \\
P \\
\{A\} \\
\text{enddo} \\
\{A \land \neg BE\}
\end{align*}
\]

is complete if the annotated program

\[
\begin{align*}
\{A \land BE\} \\
P \\
\{A\}
\end{align*}
\]

is complete.

For instance, the annotated program

\[
\begin{align*}
\{n = 5 & \land i = 0 \land p = 1\} \\
\{n = 5 & \land i \leq n \land p = 2^i\} \\
\text{while } i < n \text{ do } \{n = 5 & \land i \leq n \land p = 2^i \land i < n\} \\
\quad \{n = 5 & \land p = 2^i \land i < n\} \\
\quad \{n = 5 & \land i + 1 \leq n \land p \times 2 = 2^{i+1}\} \\
\quad i \leftarrow i + 1; \\
\quad \{n = 5 & \land i \leq n \land p = 2^i\} \\
\quad p \leftarrow p \times 2 \\
\quad \{n = 5 & \land i \leq n \land p = 2^i\} \\
\text{enddo} \\
\{n = 5 \land i \leq n \land p = 2^i \land i \geq n\} \\
\{n = 5 \land i = n \land p = 2^i\} \\
\{i = 5 \land p = 2^i\} \\
\{p = 32\}
\end{align*}
\]

is complete. It justifies the formula

\[
\begin{align*}
\{n = 5 & \land i = 0 \land p = 1\} \\
\text{while } i < n \text{ do} \\
\quad i \leftarrow i + 1; \\
\quad p \leftarrow x \times 2 \\
\text{enddo} \\
\{x = 32\}.
\end{align*}
\]