THEORY OF COMPUTATION
Programs and Computable Functions - 3

Prof. Dan A. Simovici

UMB

- The symbols

$$X_1 \quad X_2 \quad X_3 \quad \cdots$$

  are called *input variables*;

- the symbols

$$Z_1 \quad Z_2 \quad Z_3 \quad \cdots$$

  are called *local variables*;

- Y is the *output variable*;

- the symbols

$$A_1 \quad B_1 \quad C_1 \quad D_1 \quad E_1 \quad A_2 \quad B_2 \cdots$$

  are the *the labels* of $\mathcal{S}$.

A *statement* is one of the following

$$\begin{array}{l} V \leftarrow V + 1 \\ V \leftarrow V - 1 \\ V \leftarrow V \\ \text{IF } V \neq 0 \text{ GOTO } L, \end{array}$$

where V may be any variable and L may be any label.
An *instruction* is either a statement (also called unlabeled instruction) or $[L]$ followed by a statement.

A *program* is a finite sequence of instructions. The length of this list is called the *length* of the program.
The *empty program* is the program of length 0.

Definition

A *state of a program* $\mathcal{P}$ is a list of equations of the form $X = m$, where X is a variable and $m \in \mathbb{N}$ such that

- the list includes an equation for each variable that occurs in $\mathcal{P}$, and
- no two equations involve the same variable.

Example

$$
\begin{array}{ll}
[A] & \text{IF } X \neq 0 \text{ GOTO } B \\
 & Z \leftarrow Z + 1 \\
 & \text{IF } Z \neq 0 \text{ GOTO } E \\
[B] & X \leftarrow X - 1 \\
 & Y \leftarrow Y + 1 \\
 & Z \leftarrow Z + 1 \\
 & \text{IF } Z \neq 0 \text{ GOTO } A
\end{array}
$$

$STATES$:

$X = 4, Y = 3, Z = 3$

A state need not be attained by the program.

$X_1 = 4, X_2 = 5, Y = 4, Z = 4$

Variables that do not occur may also be included

$X = 3, Z = 3$ is not a state because Y is not included

$X = 3, X = 4, Y = 2, Z = 2$ is not a state because X appears twice.

Definition

Let σ be a state of a program $\mathcal{P}$ and let V be a variable that occurs in σ.
The *value* of V is the unique number q such that the equation $V = q$ is one of the equations that make up σ.

Example

The value of X at the state $X = 4, Y = 3, Z = 3$ is 4.

Definition

A *snapshot* or *instantaneous description* of a program $\mathcal{P}$ of length n is a pair (i, σ), where $1 \leqslant i \leqslant n+1$, and σ is a state of $\mathcal{P}$.

Intuition: i indicates that it is the i^{th} instruction that is about to be executed; $i = n+1$ corresponds to a "stop" instruction and the snapshot $(n+1, \sigma)$ is said to be a *terminal snapshot*.

The successor snapshot

The *successor snapshot* of (i, σ) is the snapshot (j, τ) defined as follows:

- if the i^{th} instruction of $\mathcal{P}$ is $V \leftarrow V + 1$ and σ contains the equation $V = m$, then $j = i + 1$ and τ is obtained from σ by replacing $V = m$ by $V = m + 1$;
- if the i^{th} instruction of $\mathcal{P}$ is $V \leftarrow V - 1$ and σ contains the equation $V = m$, then $j = i + 1$ and τ is obtained from σ by replacing $V = m$ by $V = m - 1$ if $m \neq 0$; if $m = 0$, then $\tau = \sigma$;
- if the i^{th} instruction of $\mathcal{P}$ is $V \leftarrow V$ then $\tau = \sigma$ and $j = i + 1$;

The successor snapshot cont'd

- if the i^{th} instruction of $\mathcal{P}$ is IF $V \neq 0$ GOTO L, then $\tau = \sigma$ and we may have two subcases:
  - if σ contains the equation $V = 0$, then $j = i + 1$;
  - if σ contains the equation $V = m$ where $m \neq 0$, them if there is an instruction of $\mathcal{P}$ labeled L, then j is the least number such that the j^{th} instruction is labeled L; otherwise, $j = n + 1$.

Example

Consider again the program shown in Slide 6:

$[A]$	IF $X \neq 0$ GOTO B	Let σ be the state $X = 4, Y = 0, Z = 0$.
	$Z \leftarrow Z + 1$	For $i = 1$, the successor is $(4, \sigma)$
	IF $Z \neq 0$ GOTO E	For $i = 2$, the successor is $(3, \tau)$
$[B]$	$X \leftarrow X - 1$	where τ consists of
	$Y \leftarrow Y + 1$	$X = 4, Y = 0, Z = 1$.
	$Z \leftarrow Z + 1$	For $i = 7$ the successor *is*
	IF $Z \neq 0$ GOTO A	$(8, \sigma)$ which is terminal.

Definition

A *computation* of a program $\mathcal{P}$ is defined as a sequence $(s_1, s_2, \ldots, s_k)$ of snapshots of $\mathcal{P}$ such that s_{i+1} is a successor of s_i for $1 \leqslant i \leqslant k-1$ and s_k is terminal.

A program may contain more than one instruction having the same label.
The definition of the successor snapshot implies that a branch instruction as always referring to the FIRST statement of the program having the label in question.

Example

The program

$$
\begin{array}{ll}
[A] & X \leftarrow X - 1 \\
 & \text{IF } X \neq 0 \text{ GOTO } A \\
[A] & X \leftarrow X + 1
\end{array}
$$

is equivalent to the program

$$
\begin{array}{ll}
[A] & X \leftarrow X - 1 \\
 & \text{IF } X \neq 0 \text{ GOTO } A \\
 & X \leftarrow X + 1
\end{array}
$$

Let $\mathcal{P}$ be a program in the language $\mathcal{S}$ and let $r_1, \ldots, r_m$ be m given numbers. Form the state σ of $\mathcal{P}$ that consists of:

- the equations $X_1 = r_1, X_2 = r_2, \ldots, X_m = r_m, Y = 0$,
- and of equations of the form $V = 0$ for each variable V in $\mathcal{P}$ other than $X_1, \ldots, X_n$ and Y.

This is the *initial state* σ of $\mathcal{P}$ and $(1, \sigma)$ is the initial snapshot.

Definition

The *m-argument function* $\psi_{\mathcal{P}}^{(m)}$ computed by the program $\mathcal{P}$ is:

- If there is a computation $s_1, \ldots, s_k$ of $\mathcal{P}$ beginning with the initial snapshot s_1 then $\psi_{\mathcal{P}}^{(m)}(r_1, \ldots, r_m)$ is the value of Y at the terminal snapshot.
- If there is no such finite computation, that is if there is an infinite computation $s_1, s_2, \ldots$ then $\psi_{\mathcal{P}}^{(m)}(r_1, \ldots, r_m)$ is undefined.

Very important: a program may be used with any number of inputs.

- If a program has n input variables but only $m < n$ are specified, the remaining input variables are set to 0 and the computation proceeds.
- If $m > n$ the extra input variables are ignored.

Example

Consider again the program with explicit line numbers:

[A]	IF $X \neq 0$ GOTO B	(1)
	$Z \leftarrow Z + 1$	(2)
	IF $Z \neq 0$ GOTO E	(3)
[B]	$X \leftarrow X - 1$	(4)
	$Y \leftarrow Y + 1$	(5)
	$Z \leftarrow Z + 1$	(6)
	IF $Z \neq 0$ GOTO A	(7)

Snapshots

$(1, \{X = 3, Y = 0, Z = 0\})$
$(4, \{X = 3, Y = 0, Z = 0\})$
$(5, \{X = 2, Y = 0, Z = 0\})$
$(6, \{X = 2, Y = 1, Z = 0\})$
$(7, \{X = 3, Y = 1, Z = 1\})$
$(1, \{X = 3, Y = 1, Z = 1\})$
$\vdots$
$(1, \{X = 0, Y = 3, Z = 3\})$
$(2, \{X = 0, Y = 3, Z = 3\})$
$(3, \{X = 0, Y = 3, Z = 4\})$
$(8, \{X = 0, Y = 3, Z = 4)\})$

- As previously mentioned, we are permitting each program to be used with any number of inputs.
- If a program has n input variables, but only $m < n$ are specified, the remaining input variables are set to 0 and the computation proceeds.
- If m values are specified, where $m > n$, the extra input variables are ignored.

For any program $\mathcal{P}$ and any positive integer m, the function $\psi_{\mathcal{P}}^{(m)}(x_1, \ldots, x_m)$ is said to be computed by $\mathcal{P}$.
A partial function g is said to be partially computable if it is computed by some program. That is, g is partially computable if there exists a program $\mathcal{P}$ such that

$$g(r_1, \ldots, r_m) = \psi_{\mathcal{P}}^{(m)}(r_1, \ldots, r_m)$$

When one side of this equation is undefined, then so is the other side.

A function g of m variables is total if $g(r_1, \ldots, r_m)$ is defined for all $r_1, \ldots, r_m)$.
A function is computable if it is both partially computable and total.

Example

The functions $x, x + y, x \cdot y$ are computable; the function $x - y$ is partially computable.

Example

For the program

$$\begin{array}{ll} [A] & X \leftarrow X + 1 \\ & \text{IF } X \neq 0 \text{ GOTO } A \end{array}$$

the one-argument function $\psi_{\mathcal{P}}^{1}(x)$ is undefined for all x. So, the nowhere defined function must be included in the class of partially computed functions.

Let f be a partially computable function computed by a program $\mathcal{P}$. We make the following assumptions:

- the variables in $\mathcal{P}$ belong to the list $Y, X_1, \ldots, X_n, Z_1, \ldots, Z_k$;
- the labels in $\mathcal{P}$ are included in the list $E, A_1, \ldots, A_\ell$;
- for each instruction IF $V \neq 0$ GOTO A there is an instruction in $\mathcal{P}$ labeled A (that is, E is the single exit label).

Then $\mathcal{P}$ is written as:

$$\mathcal{P} = \mathcal{P}(Y, X_1, \ldots, X_n, Z_1, \ldots, Z_k; E, A_1, \ldots, A_\ell).$$

The notation

$$\mathcal{P} = \mathcal{P}(Y, X_1, \ldots, X_n, Z_1, \ldots, Z_k; E, A_1, \ldots, A_\ell).$$

can be used to write:

$$\begin{aligned} \mathcal{Q} \quad = \quad & \mathcal{P}(Z_m, Z_{m+1}, \ldots, Z_{m+n}, Z_{m+n+1}, \ldots, Z_{m+n+k}; \\ & E_m, A_{m+1}, \ldots, A_{m+\ell}) \end{aligned}$$

to denote a program obtained from $\mathcal{P}$ by replacing the variables and labels by others.

To use a macro like $W \leftarrow f(V_1, \ldots, V_n)$ is regarded as an abbreviation of:

$$
\begin{array}{ll}
 & Z_m \leftarrow 0 \\
 & Z_{m+1} \leftarrow V_1 \\
 & \vdots \\
 & Z_{m+n} \leftarrow V_n \\
 & Z_{m+n+1} \leftarrow 0 \\
 & Z_{m+n+2} \leftarrow 0 \\
 & \vdots \\
 & Z_{m+n+k} \leftarrow 0 \\
 & \mathcal{Q}_m \\
{[E_m]} & W \leftarrow Z_m
\end{array}
$$

m is chosem so large that none of the variables or labels used in $\mathcal{Q}_m$ occur in the main program that contains $\mathcal{Q}_m$.

Note that:

- the expansion sets the variables corresponding to the output variable Y and to the local variables of $\mathcal{P}$, $Z_{m+n+1}, \ldots, Z_{m+n+k}$ to 0;
- the variables corresponding to $X_1, \ldots, X_n$ are set to the values of $V_1, \ldots, V_n$;
- setting the variables equal to 0 is necessary because the expansion may be part of a loop in the main program;
- when $\mathcal{Q}_m$ terminates the value of Z_m is $f(V_1, \ldots, V_n)$.

If $f(V_1, \ldots, V_n)\uparrow$ (is undefined), $\mathcal{Q}_m$ never terminates. Thus, f is not total and the macro

$$W \leftarrow f(V_1, \ldots, V_n)$$

is encountered in a program, the main program will never terminate.

Example

The program

$$Z \leftarrow X_1 - X_2$$
$$Y \leftarrow Z + X_3$$

computes the function $f(x_1, x_2, x_3)$ defined as

$$f(x_1, x_2, x_3) = \begin{cases} (x_1 - x_2) + x_3 & \text{if } x_1 \geqslant x_2, \\ \uparrow & \text{otherwise.} \end{cases}$$

Note that $f(2, 5, 6)$ is undefined! The computation never gets past the attempt to compute $2 - 5$.

Augmenting the language to include macros of the form

$$\text{IF } P(V_1, \ldots, V_n) \text{ GOTO } L$$

where $P(x_1, \ldots, x_n)$ is a computable predicate.
Recall the convention that TRUE $= 1$ and FALSE $= 0$.
This regards predicate as total functions whose values are always 0 or 1.

The macro expansion of

$$\text{IF } P(V_1, \ldots, V_n) \text{ GOTO } L$$

is

$$Z \leftarrow P(V_1, \ldots, V_n)$$
$$\text{IF } Z \neq 0 \text{ GOTO } L$$

Note that the predicate $P(x)$ defined by

$$P(x) = \begin{cases} \text{TRUE} & \text{if } x = 0, \\ \text{FALSE} & \text{otherwise} \end{cases}$$

is computable by the program

$$\text{IF } X \neq 0 \text{ GOTO } E$$
$$Y \leftarrow Y + 1$$

Example

An instruction used frequently is

$$\text{IF } V = 0 \text{ GOTO } L$$

This is legitimate because we can compute $V = 0$.