Prof. Dan A. Simovici

UMB
Outline

1. A Programming Language

2. Working Informally with Programs
The Language $S$

We introduce a “programming language” $S$ that will help us formalize the notion of computable function. Main features of $S$ are:

- variables assume only non-negative integer values $0, 1, 2, \ldots$;
- the letters $X_1, X_2, \ldots$ denote input variables;
- the letter $Y$ is the output variable;
- the letters $Z_1, Z_2, \ldots$ denote local variables.

We will often write $X$ and $Z$ instead of $X_1$ and $Z_1$, respectively. Unlike proper programming languages there is no upper limit on the values these variables may assume.
A program is a list of instructions that may or may not be labeled. The beauty of $S$ is that it consists only of four types of instructions:

- $V \leftarrow V + 1$ increase by 1 the value of $V$
- $V \leftarrow V - 1$ decrease by 1 the value of $V$
- if this value is positive; if the value is 0 leave it unchanged
- $V \leftarrow V$ do nothing instruction
- IF $V \neq 0$ GOTO $L$ if value of $V$ is nonzero perform the instruction with label $L$; otherwise proceed with next instruction
Example

A very simple program is

\[
X \leftarrow X + 1 \\
X \leftarrow X + 1
\]

The effect of this program is to increase the value of \( X \) by 2.
Labels and Variables

The labels of instructions in $S$ can be chosen among

$$A_1, B_1, C_1, D_1, E_1, A_2, B_2, C_2, D_2, E_2, A_3, \ldots$$

and the subscript 1 may be omitted.

Instructions may or may not have labels. Label is written to the left of the instruction in square brackets:

$$[B] \quad Z \leftarrow Z - 1$$

The output variable $Y$ and the local variables $Z_i$ have the value 0 initially.

Value of a variable $X_i$ will be denoted by $x_i$. 
Example

The program

\[
[A] \quad X \leftarrow X - 1 \\
Y \leftarrow Y + 1 \\
\text{IF } X \neq 0 \text{ GOTO } A
\]

computes the function defined by

\[
f(x) = \begin{cases} 
1 & \text{if } x = 0, \\
x & \text{otherwise.}
\end{cases}
\]
Example

In a program like

\[
\ldots
\]
\[
[A] \quad \ldots
\]
\[
\ldots
\]
\[
Z \leftarrow Z + 1
\]
\[
\text{IF } Z \neq 0 \text{ GOTO } A
\]
\[
\ldots
\]

the effect is equivalent to an unconditional jump to the statement labeled by \(A\). The effect of these two lines involving \(Z\) is the same as an unconditional jump GOTO \(A\).
Note that GOTO A is not among the three types of instruction of $S$.
We shall use GOTO A as an abbreviated form of the following fragment code:

$$Z \leftarrow Z + 1$$
$$\text{IF } Z \neq 0 \text{ GOTO } A$$

The label $E$ is the exit label. Therefore, GOTO $E$ triggers the end of the program.
Example

The next program copies the value of $X$ into $Y$:

[A] IF $X \neq 0$ GOTO B

$Z \rightarrow Z + 1$

IF $Z \neq 0$ GOTO E

[B] $X \leftarrow X - 1$

$Y \leftarrow Y + 1$

$Z \rightarrow Z + 1$

IF $Z \neq 0$ GOTO A

This program computes the function $f(x) = x$. 
The previous program “destroys” the value of $X$. A variant that preserves this value is given next.

[A] IF $X \neq 0$ GOTO B
    GOTO C
[B] $X \leftarrow X - 1$
    $Y \leftarrow Y + 1$
    $Z \leftarrow Z + 1$
    GOTO A
[C] IF $Z \neq 0$ GOTO D
    GOTO E
[D] $Z \leftarrow Z - 1$
    $X \leftarrow X + 1$
    GOTO C
Note that:

- in the first loop the program copies the value of $X$ in both $Y$ and $X$;
- in the second loop the value of $X$ is restored;
- when the program ends both $X$ and $Y$ contain the original value of $X$ and $z = 0$;

This program justifies the introduction of the macro $V \leftarrow V'$. 
The program

\[ [L] \quad V \leftarrow V - 1 \]
\[ \text{IF } V \neq 0 \text{ GOTO } L \]

sets the value of \( V \) to 0. It is abbreviated as the macro

\[ V \leftarrow 0 \]

If we want to expand the macro \( v \leftarrow 0 \), we need to take care that
the label \( L \) is different from any other label in the main program.
A program that computes the function $f(x_1, x_2) = x_1 + x_2$ is

$$
Y \leftarrow X_1 \\
Z \leftarrow X_2 \\
[B] \quad \text{IF } Z \neq 0 \text{ GOTO } A \\
\quad \text{GOTO } E \\
[A] \quad Z \leftarrow Z - 1 \\
\quad Y \leftarrow Y + 1 \\
\quad \text{GOTO } B
$$

Note that $Z$ is used to preserve the value of $X_2$. 
A program that multiplies

The next program computes the function $f(x_1, x_2) = x_1 x_2$:

\[
    Z_2 \leftarrow X_2
    
    [B] \quad \text{IF } Z_2 \neq 0 \text{ GOTO A}
    \quad \text{GOTO E}
    
    [A] \quad Z_2 \leftarrow Z_2 - 1
    \quad Z_1 \leftarrow X_1 + Y
    \quad Y \leftarrow Z_1
    \quad \text{GOTO B}
\]

Note that $Z_1 \leftarrow X_1 + Y$ is not permitted in $S$; this means that this instruction must be replaced by a program that computes it. This is called macro expansion.
Macro expansion of $Z_1 \leftarrow X_1 + Y$

- $Z_2 \leftarrow X_2$
- \[B\] IF $Z_2 \neq 0$ GOTO A
  - GOTO E
- \[A\] $Z_2 \leftarrow Z_2 - 1$
  - $Z_1 \leftarrow X_1$
  - $X_3 \leftarrow Y$
- \[B_2\] IF $Z_3 \neq 0$ GOTO $A_2$
  - GOTO $E_2$
  - $Z_3 \leftarrow Z_3 - 1$
  - $Z_1 \leftarrow Z_1 + 1$
  - GOTO $B_2$
- \[E_2\] $Y \leftarrow Z_1$
  - GOTO $B$
Note that

- The local variable $Z_1$ in the addition program on Slide 14 was replaced by $Z_3$ because $Z_1$ is also used as a local variable in the multiplication program.

- The labels $A, B, E$ are used in the multiplication program and, therefore, cannot be used in the macro expansion. Instead, we used $A_2, B_2, C_2$.

- GOTO $E_2$ terminates the addition. Hence it is necessary that the instruction immediately following the macro expansion be labeled $E_2$. 
Example

The next program computes a partial function, namely

\[ g(x_1, x_2) = \begin{cases} 
  x_1 - x_2 & \text{if } x_1 \geq x_2, \\
  \uparrow & \text{if } x_1 < x_2,
\end{cases} \]

The symbol “↑” means that the function is not defined (when \( x_1 < x_2 \)).
Y ← X₁
Z ← X₂
[C] IF Z ≠ 0 GOTO A
     GOTO E
[A] IF Y ≠ 0 GOTO B
     GOTO A
[B] Y ← Y − 1
     Z ← Z − 1
     GOTO C
Working Informally with Programs

\begin{align*}
Y & \leftarrow X_1 \\
Z & \leftarrow X_2 \\
\text{[C]} & \quad \text{IF } Z \neq 0 \text{ GOTO A} \\
& \quad \text{GOTO E} \\
\text{[A]} & \quad \text{IF } Y \neq 0 \text{ GOTO B} \\
& \quad \text{GOTO A} \\
\text{[B]} & \quad Y \leftarrow Y - 1 \\
& \quad Z \leftarrow Z - 1 \\
& \quad \text{GOTO C}
\end{align*}

start with $X_1 = 5$, $X_2 = 2$,  
set $Y = 5$ and $Z = 2$,  
then $Y = 4$ and $Z = 1$,  
then $Y = 3$ and $Z = 0$,  
computation ends with $Y = 3 = 5 - 2$

if $X_1 = m$ and $X_2 = n$, $m < n$  
them $Y$ becomes 0and  
program never terminates.