

### The Syntax of $\mathcal{L}$

We will now develop a precise mathematical description of the programming language  $\mathcal{L}$ .

The symbols

$X_1, X_2, X_3, \dots$

are called **input variables**,

$Z_1, Z_2, Z_3, \dots$

are called **local variables**,

and  $Y$  is called the **output variable** of  $\mathcal{L}$ .

September 15, 2009

Theory of Computation Lecture 3: Programs and Computable Functions II

1

### The Syntax of $\mathcal{L}$

The symbols

$A_1, B_1, C_1, D_1, E_1, A_2, B_2, C_2, \dots$

are called **labels** of  $\mathcal{L}$ .

For variables and labels, the subscript 1 can always be omitted.

September 15, 2009

Theory of Computation Lecture 3: Programs and Computable Functions II

2

### The Syntax of $\mathcal{L}$

A **statement** is one of the following:

$V \leftarrow V+1$

$V \leftarrow V-1$

$V \leftarrow V$

IF  $V \neq 0$  GOTO L

Here,  $V$  may be any variable and L may be any label.

Note that the statement  $V \leftarrow V$  leaves all values unchanged, so it is a “**dummy**” command.

We will later see why it is useful to have  $V \leftarrow V$  in our set of statements.

September 15, 2009

Theory of Computation Lecture 3: Programs and Computable Functions II

3

### The Syntax of $\mathcal{L}$

An **instruction** is either

- a statement (unlabeled instruction) or
- [L] followed by a statement (instruction labeled L)

A **program** is a list (i.e., a finite sequence) of instructions.

The **length** of this list is called the length of the program.

We also include the **empty program** – the program of length 0 – in the set of all programs.

September 15, 2009

Theory of Computation Lecture 3: Programs and Computable Functions II

4

### The Syntax of $\mathcal{L}$

While a program is being executed, its variables assume different numerical values.

This motivates the concept of the **state** of a program:

A state of a program  $\mathcal{P}$  is a list of equations of the form

$V = m,$

where  $V$  is a variable and  $m$  is a number,

including exactly one equation for each variable that occurs in  $\mathcal{P}$  (and possibly equations for other variables).

September 15, 2009

Theory of Computation Lecture 3: Programs and Computable Functions II

5

### The Syntax of $\mathcal{L}$

Consider the following program  $\mathcal{P}$ :

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

$Z \leftarrow Z+1$

IF  $Z \neq 0$  GOTO E

[B]  $X \leftarrow X-1$

$Y \leftarrow Y+1$

$Z \leftarrow Z+1$

IF  $Z \neq 0$  GOTO A

Is the list

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

a state of  $\mathcal{P}$ ?

Yes.

How about

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

Yes.

And  $X = 3, Z = 3$ ?

No.

September 15, 2009

Theory of Computation Lecture 3: Programs and Computable Functions II

6

### The Syntax of $\mathcal{L}$

Let  $\sigma$  be a state of  $\mathcal{P}$  and let  $V$  be a variable that occurs in  $\sigma$ .

The **value** of  $V$  at  $\sigma$  is then the unique number  $q$  such that the equation  $V = q$  is one of the equations in  $\sigma$ .

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

### The Syntax of $\mathcal{L}$

Consider a machine that can execute programs of  $\mathcal{L}$ .

During program execution, this machine needs to store the state of a program, i.e., keep track of the values of variables.

Is there anything else that needs to be stored?

Yes, we also need to store **which instruction** is to be executed next.

### The Syntax of $\mathcal{L}$

Therefore, we define a **snapshot** or **instantaneous description** of a program  $\mathcal{P}$  of length  $n$  to be a pair  $(i, \sigma)$  where  $1 \leq i \leq (n + 1)$ , and  $\sigma$  is a state of  $\mathcal{P}$ .

The number  $i$  indicates the number of the instruction that is to be executed next.

$i = n + 1$  corresponds to a "stop" instruction.

A snapshot  $(i, \sigma)$  of a program  $\mathcal{P}$  of length  $n$  is called **terminal** if  $i = n + 1$ .

If  $s = (i, \sigma)$  is a snapshot of  $\mathcal{P}$  and  $V$  is a variable of  $\mathcal{P}$ , then the value of  $V$  at  $s$  just means the value of  $V$  at  $\sigma$ .

### The Syntax of $\mathcal{L}$

If  $(i, \sigma)$  is a nonterminal snapshot of  $\mathcal{P}$ , we define the successor of  $(i, \sigma)$  to be the snapshot  $(j, \tau)$  defined as follows:

#### Case 1:

The  $i$ -th instruction of  $\mathcal{P}$  is  $V \leftarrow V+1$  and  $\sigma$  contains the equation  $V = m$ .

Then  $j = i + 1$  and  $\tau$  is obtained from  $\sigma$  by replacing the equation  $V = m$  by  $V = m + 1$  (the value of  $V$  at  $\tau$  is  $m + 1$ ).

### The Syntax of $\mathcal{L}$

#### Case 2:

The  $i$ -th instruction of  $\mathcal{P}$  is  $V \leftarrow V-1$  and  $\sigma$  contains the equation  $V = m$ .

Then  $j = i + 1$  and  $\tau$  is obtained from  $\sigma$  by replacing the equation  $V = m$  by  $V = m - 1$  if  $m \neq 0$ .

If  $m = 0$ , then  $\tau = \sigma$ .

#### Case 3:

The  $i$ -th instruction of  $\mathcal{P}$  is  $V \leftarrow V$ .

Then  $j = i + 1$  and  $\tau = \sigma$ .

### The Syntax of $\mathcal{L}$

#### Case 4:

The  $i$ -th instruction of  $\mathcal{P}$  is IF  $V \neq 0$  GOTO  $L$ .

Then  $\tau = \sigma$  and there are two subcases:

#### Case 4a:

$\sigma$  contains the equation  $V = 0$ .

Then  $j = i + 1$ .

#### Case 4b:

$\sigma$  contains the equation  $V = m$  where  $m \neq 0$ .

Then, if there is an instruction of  $\mathcal{P}$  labeled  $L$ ,  $j$  is the least number such that the  $j$ -th instruction of  $\mathcal{P}$  is labeled  $L$ . Otherwise,  $j = n + 1$ .

### The Syntax of $\mathcal{L}$

A **computation** of a program  $\mathcal{P}$  is defined to be a sequence  $s_1, s_2, \dots, s_k$  of snapshots of  $\mathcal{P}$  such that  $s_{i+1}$  is the successor of  $s_i$  for  $i = 1, 2, \dots, k - 1$  and  $s_k$  is terminal.

### Computable Functions

What does it exactly mean when we say that a program computes a function?

We would like to find a precise definition for this.

Let  $\mathcal{P}$  be any program in the language  $\mathcal{L}$  and let  $r_1, \dots, r_m$  be  $m$  given numbers.

We form the state  $\sigma$  of  $\mathcal{P}$  which consists of the equations

$X_1 = r_1, X_2 = r_2, \dots, X_m = r_m, Y = 0$   
together with the equations  $V = 0$  for all other variables  $V$  in  $\mathcal{P}$ .

We call this the **initial state**.

### Computable Functions

Moreover, we call the snapshot  $(1, \sigma)$  the **initial snapshot**.

When running the program, there are two possible outcomes:

**Case 1:**

There is a computation  $s_1, s_2, \dots, s_k$  of  $\mathcal{P}$  beginning with the initial snapshot.

Then we write  $\psi_{\mathcal{P}}^{(m)}(r_1, r_2, \dots, r_m)$  for the value of the variable  $Y$  at the terminal snapshot  $s_k$ .

### Computable Functions

**Case 2:**

There is no such computation; i.e., there is an infinite sequence  $s_1, s_2, s_3, \dots$  beginning with the initial snapshot.

In this case,  $\psi_{\mathcal{P}}^{(m)}(r_1, r_2, \dots, r_m)$  is undefined.

### Sample Computation

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

We write:

$\psi_{\mathcal{P}}^{(1)}(r) = r.$

- (1,  $\{X = 0, Y = r, Z = r\}$ ),
- (2,  $\{X = 0, Y = r, Z = r\}$ ),
- (3,  $\{X = 0, Y = r, Z = r + 1\}$ ),
- (8,  $\{X = 0, Y = r, Z = r + 1\}$ ).

### Computation

For any program  $\mathcal{P}$  and any positive integer  $m$ , the function  $\psi_{\mathcal{P}}^{(m)}(r_1, r_2, \dots, r_m)$  is said to be **computed** by  $\mathcal{P}$ .

We allow any program to be run with any number of inputs.

Consider the summation program from the previous lecture:

$\psi_{\mathcal{P}}^{(2)}(r_1, r_2) = r_1 + r_2$   
 $\psi_{\mathcal{P}}^{(1)}(r_1) = r_1 + 0 = r_1$   
 $\psi_{\mathcal{P}}^{(3)}(r_1, r_2, r_3) = r_1 + r_2$

### Computation

For any program  $\mathcal{P}$  and any positive integer  $m$ , the function  $\psi_{\mathcal{P}}^{(m)}(r_1, r_2, \dots, r_m)$  is said to be **computed** by  $\mathcal{P}$ .

In general, a **partial function**  $f$  on a set  $S^m$  is a function whose domain is a subset of  $S^m$ .

If a partial function on  $S^m$  has the domain  $S^m$ , then it is called **total**.

September 15, 2009

Theory of Computation  
Lecture 3: Programs and Computable Functions II

19

### Computation

A given partial function  $g$  (of one or more variables) is said to be **partially computable** if it is computed by some program.

This is the case if there is a program  $\mathcal{P}$  such that  $g(r_1, r_2, \dots, r_m) = \psi_{\mathcal{P}}^{(m)}(r_1, r_2, \dots, r_m)$  for all  $r_1, r_2, \dots, r_m$ .

This means not only that both sides have the same value when they are **defined**, but also that when either side of the equation is **undefined**, the other one is as well.

A function is said to be **computable** if it is both partially computable and total.

September 15, 2009

Theory of Computation  
Lecture 3: Programs and Computable Functions II

20

### Macros

So far we have used macros in an informal way. Let us now develop a more precise definition of them.

Let  $f(x_1, x_2, \dots, x_n)$  be a partially computable function, and  $\mathcal{P}$  be a program that computes  $f$ .

Let us assume that

- the variables in  $\mathcal{P}$  are named  $Y, X_1, \dots, X_n, Z_1, \dots, Z_k$ ,
- the labels in  $\mathcal{P}$  are named  $E, A_1, \dots, A_l$ , and
- for each instruction of  $\mathcal{P}$  of the form  
IF  $V \neq 0$  GOTO  $A_i$   
there is in  $\mathcal{P}$  an instruction labeled  $A_i$ .

September 15, 2009

Theory of Computation  
Lecture 3: Programs and Computable Functions II

21

### Macros

We can modify any program of the language  $\mathcal{L}$  to comply with these assumptions.

We write:

$$\mathcal{P} = \mathcal{P}(Y, X_1, \dots, X_n, Z_1, \dots, Z_k; E, A_1, \dots, A_l)$$

In particular, we will use:

$$Q_m = \mathcal{P}(Z_m, Z_{m+1}, \dots, Z_{m+n}, Z_{m+n+1}, \dots, Z_{m+n+k}; E_m, A_{m+1}, \dots, A_{m+l})$$

for a given value  $m$ .

September 15, 2009

Theory of Computation  
Lecture 3: Programs and Computable Functions II

22