

The Programming Language \mathcal{L}

We will explore computability theory using an extremely simple programming language called \mathcal{L} .
 Let us take a look at this language.
 \mathcal{L} uses variables holding **numbers**. Throughout the course, 'number' will refer to a **nonnegative integer**.
 The variables named X_1, X_2, X_3, \dots are the **input variables** of \mathcal{L} , Y is the **output variable** of \mathcal{L} , and Z_1, Z_2, Z_3, \dots are the **local variables** of \mathcal{L} .

September 10, 2009 Theory of Computation
Lecture 2: Programs and Computable Functions I 1

The Programming Language \mathcal{L}

We do not have to write the subscript 1, so instead of X_1 or Z_1 we can write X or Z .

\mathcal{L} also includes **labels**. These labels are named $A_1 B_1 C_1 D_1 E_1$
 $A_2 B_2 C_2 D_2 E_2$
 $\dots \dots \dots \dots \dots$

Again, the subscript 1 can be omitted.

September 10, 2009 Theory of Computation
Lecture 2: Programs and Computable Functions I 2

The Programming Language \mathcal{L}

A **program** of \mathcal{L} consists of a list (a finite sequence) of **instructions**.

What do the instructions look like?
 There are only three types of instructions in \mathcal{L} :

- **increment**
- **decrement**
- **conditional branch**

September 10, 2009 Theory of Computation
Lecture 2: Programs and Computable Functions I 3

The Programming Language \mathcal{L}

In the following list of instructions, the letter **V** stands for any **variable** in the program, and **L** stands for a **label**:

$V \leftarrow V+1$	increase by 1 the value of the variable V
$V \leftarrow V-1$	If the value of V is 0, leave it unchanged; otherwise decrease by 1 the value of the variable V .
If $V \neq 0$ GOTO L	If the value of V is nonzero, perform the instruction with label L next; otherwise proceed to the next instruction in the list.

September 10, 2009 Theory of Computation
Lecture 2: Programs and Computable Functions I 4

The Programming Language \mathcal{L}

These are the only three instructions in our language \mathcal{L} .
 You will be surprised how powerful this extremely simple programming language is.
 We just need two more conventions:

- The output variable Y and the local variables Z_i initially have **the value 0**.
- A program halts when it attempts to move to a nonexistent instruction (beyond the end of the list) or branch to a nonexistent label.

September 10, 2009 Theory of Computation
Lecture 2: Programs and Computable Functions I 5

The Programming Language \mathcal{L}

Note that we have an unlimited supply of variables and labels.
 Moreover, there is no upper limit on the value that a variable can contain.
 Therefore, the language \mathcal{L} is not a practical language, but it is well-suited for the theoretical evaluation of algorithms.

September 10, 2009 Theory of Computation
Lecture 2: Programs and Computable Functions I 6

Sample Programs

Consider the following program:

```
[A] X ← X-1
    Y ← Y+1
    IF X≠0 GOTO A
```

What function does this program compute?

$f(x) = 1$, if $x = 0$
 $= x$, otherwise.

What would we have to change if we wanted to compute the function $f(x) = x$?

September 10, 2009

Theory of Computation
Lecture 2: Programs and Computable Functions I

7

Sample Programs

The following program computes $f(x) = x$:

```
[A] IF X≠0 GOTO B
    Z ← Z+1
    IF Z≠0 GOTO E
[B] X ← X-1
    Y ← Y+1
    IF X≠0 GOTO B
```

September 10, 2009

Theory of Computation
Lecture 2: Programs and Computable Functions I

8

Sample Programs

In the previous example, the lines

```
Z ← Z+1
IF Z≠0 GOTO L
```

were used to implement an instruction that we could call

```
GOTO L
```

First, we make sure that Z has a nonzero value, and then the conditional branch (actually, here it is an unconditional branch) is executed.

September 10, 2009

Theory of Computation
Lecture 2: Programs and Computable Functions I

9

Sample Programs

From now on we will use the **macro** GOTO L in our programs.

We know that we could always replace GOTO L with its **macro expansion** to obtain a valid \mathcal{L} program.

Now remember the program computing the function $f(x) = x$.

Although it computes its output correctly, it deletes (sets to zero) the original input.

This is an undesirable behavior, so we should come up with an improved program.

September 10, 2009

Theory of Computation
Lecture 2: Programs and Computable Functions I

10

Sample Programs

```
[A] IF X≠0 GOTO B
    GOTO C
[B] X ← X-1
    Y ← Y+1
    Z ← Z+1
    GOTO A
[C] IF Z≠0 GOTO D
    GOTO E
[D] Z ← Z-1
    X ← X+1
    GOTO C
```

September 10, 2009

Theory of Computation
Lecture 2: Programs and Computable Functions I

11

Sample Programs

The previous program justifies the introduction of a macro

```
V ← V'
```

The execution of this macro will replace the contents of variable V by those of variable V' without changing the contents of V'.

However, there is one problem:

The previous program could rely on the initial condition $Y = 0$, which is not guaranteed for any variable V.

September 10, 2009

Theory of Computation
Lecture 2: Programs and Computable Functions I

12

Sample Programs

To solve this problem, we introduce the macro
 $V \leftarrow 0$

Its macro expansion is:

```
[L]  V ← V-1
      IF V≠0 GOTO L
```

Of course, the label L has to be chosen to be different from any other label in the program.

Now we can write down the macro expansion of $V \leftarrow V'$:

Sample Programs

```
V ← 0
[A] IF V'≠0 GOTO B
    GOTO C
[B] V' ← V'-1
    V ← V+1
    Z ← Z+1
    GOTO A
[C] IF Z≠0 GOTO D
    GOTO E
[D] Z ← Z-1
    V' ← V'+1
    GOTO C
```

Sample Programs

Another example: $f(x_1, x_2) = x_1 + x_2$

```
Y ← X1
Z ← X2
[B] IF Z≠0 GOTO A
    GOTO E
[A] Z ← Z-1
    Y ← Y+1
    GOTO B
```

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} .

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.

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.

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 10, 2009

Theory of Computation
Lecture 2: Programs and Computable Functions I

19

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 10, 2009

Theory of Computation
Lecture 2: Programs and Computable Functions I

20

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 10, 2009

Theory of Computation
Lecture 2: Programs and Computable Functions I

21