

Numerical Representation of Strings

Correspondingly, we define the following:

For $x \in \mathbb{N}$, let w be the string in \tilde{A}^* which represents x in base l .

Let w' be obtained from w by crossing out all of the symbols that belong to $\tilde{A} - A$, so $w' \in A^*$.

We write $\text{DOWNCHANGE}_{n,l}(x)$ for the number which w' represents in base n .

Example:

$\text{DOWNCHANGE}_{2,6}(109) = 5$

The representation of 109 in **base 6** is $s_2s_6s_1$. We cross out the s_6 and get s_2s_1 . In **base 2**, s_2s_1 represents the number 5.

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

1

Numerical Representation of Strings

$\text{UPCHANGE}_{n,l}$ and $\text{DOWNCHANGE}_{n,l}$ are actually primitive recursive functions, but now we will just show that they are computable.

This program computes $\text{UPCHANGE}_{n,l}$:

```
[A] IF X = 0 GOTO E
     Z ← LTENDn(X) // Z receives leftmost symbol
     X ← LTRUNCn(X) // removes this symbol from x
     Y ← l · Y + Z // add to output
     GOTO A
```

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

2

Numerical Representation of Strings

Let us look at the computation of $\text{UPCHANGE}_{2,10}(12)$ iteration by iteration (string $s_2s_1s_2$):

```
[A] IF X = 0 GOTO E    0. Z = 0, Y = 0, X = 12
     Z ← LTENDn(X)    1. Z = 2, Y = 2, X = 4
     X ← LTRUNCn(X)   2. Z = 1, Y = 21, X = 2
     Y ← l · Y + Z     3. Z = 2, Y = 212, X = 0
     GOTO A
                               Result: 212
```

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

3

Numerical Representation of Strings

This program computes $\text{DOWNCHANGE}_{n,l}$:

```
[A] IF X = 0 GOTO E
     Z ← LTENDl(X) // Z receives leftmost symbol
     X ← LTRUNCl(X) // removes this symbol from x
     IF Z > n GOTO A // check if we must cross out Z
     Y ← n · Y + Z // if not, add digit Z to output
     GOTO A
```

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

4

A Progr. Language for String Computations

The instructions in our programming language \mathcal{L} do not seem well-designed for string computations.

Let us consider the instruction type $V \leftarrow V + 1$.

For example, if we use the alphabet $\{a, b, c, d\}$, and the variable V contains the number corresponding to the string **cadd**, what will this instruction do?

It will turn **cadd** into **cbaa**.

Why would we want to have this as one of our fundamental operations in the programming language?

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

5

A Progr. Language for String Computations

To improve this situation, we will now introduce **specific programming languages** \mathcal{L}_n for computations on alphabets of size n for all $n > 0$.

All variables will be the same as in \mathcal{L} , except that we now consider their values to be from A^* with $|A| = n$.

The use of labels, formal rules of syntax, and macro expansions will also be identical to \mathcal{L} .

An m -ary partial function on A^* which is computed by a program in \mathcal{L}_n is said to be **partially computable** in \mathcal{L}_n . If the function is total and partially computable in \mathcal{L}_n , it is called **computable** in \mathcal{L}_n .

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

6

A Progr. Language for String Computations

Instruction	Interpretation
$V \leftarrow \sigma V$ (with $\sigma \in A$)	Place the symbol σ to the left of the string which is the value of V .
$V \leftarrow V^{\cdot}$	Delete the final symbol of the string which is the value of V . If the value of V is 0, leave it unchanged.
IF V ENDS σ GOTO L	If the value of V ends in the symbol σ , continue with the first instruction labeled L ; otherwise proceed to the next instruction.

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

7

A Progr. Language for String Computations

Although the instructions of \mathcal{L}_n refer to strings, we can still think of them as referring to the numbers associated with the strings.

For example, given an alphabet with n symbols, the instruction

$$V \leftarrow s_i V$$

has the effect of replacing the current value of the variable V with the value

$$i \cdot n^{|V|} + V.$$

So you see that the "natural" string operations in \mathcal{L}_n have complex numerical counterparts.

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

8

A Progr. Language for String Computations

Now let us look at some useful macros for use in \mathcal{L}_n and their expansions:

1. The macro IF $V \neq 0$ GOTO L has the expansion

$$\begin{aligned} & \text{IF } V \text{ ENDS } s_1 \text{ GOTO } L \\ & \text{IF } V \text{ ENDS } s_2 \text{ GOTO } L \\ & \vdots \\ & \text{IF } V \text{ ENDS } s_n \text{ GOTO } L \end{aligned}$$

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

9

A Progr. Language for String Computations

2. The macro $V \leftarrow 0$ has the expansion

$$\begin{aligned} \text{[A]} \quad & V \leftarrow V^{\cdot} \\ & \text{IF } V \neq 0 \text{ GOTO } A \end{aligned}$$

3. The macro GOTO L has the expansion

$$\begin{aligned} & Z \leftarrow 0 \\ & Z \leftarrow s_1 Z \\ & \text{IF } Z \text{ ENDS } s_1 \text{ GOTO } L \end{aligned}$$

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

10

A Progr. Language for String Computations

4. The macro $V' \leftarrow V$ has a complicated expansion.

So let us first introduce the description

$$\text{IF } V \text{ ENDS } s_i \text{ GOTO } B_i \quad (1 \leq i \leq n)$$

to stand for

$$\begin{aligned} & \text{IF } V \text{ ENDS } s_1 \text{ GOTO } B_1 \\ & \text{IF } V \text{ ENDS } s_2 \text{ GOTO } B_2 \\ & \vdots \\ & \text{IF } V \text{ ENDS } s_n \text{ GOTO } B_n \end{aligned}$$

This is also called a **filter**.

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

11

A Progr. Language for String Computations

$$Z \leftarrow 0$$

$$V' \leftarrow 0$$

$$\text{[A]} \quad \begin{aligned} & \text{IF } V \text{ ENDS } s_i \text{ GOTO } B_i \quad (1 \leq i \leq n) \\ & \text{GOTO } C \end{aligned}$$

$$\begin{aligned} \text{[B]}_i \quad & V \leftarrow V^{\cdot} \quad (1 \leq i \leq n) \\ & V' \leftarrow s_i V' \quad : \\ & Z \leftarrow s_i Z \quad : \\ & \text{GOTO } A \quad : \end{aligned}$$

$$\text{[C]} \quad \begin{aligned} & \text{IF } Z \text{ ENDS } s_i \text{ GOTO } D_i \quad (1 \leq i \leq n) \\ & \text{GOTO } E \end{aligned}$$

$$\begin{aligned} \text{[D]}_i \quad & Z \leftarrow Z^{\cdot} \quad (1 \leq i \leq n) \\ & V \leftarrow s_i V \quad : \\ & \text{GOTO } C \quad : \end{aligned}$$

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

12

A Progr. Language for String Computations

Finally, let us look at two useful functions, namely $f(x) = x + 1$ and $g(x) = x^{-1}$.

We want to show that these functions are computable in \mathcal{L}_n by writing programs that compute them.

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

13

A Progr. Language for String Computations

Example 1: An \mathcal{L}_n program that computes $f(x) = x + 1$

```
[B]  IF X ENDS  $s_i$  GOTO  $A_i$  ( $1 \leq i \leq n$ )
      Y  $\leftarrow$   $s_i$ Y
      GOTO E
[ $A_i$ ] X  $\leftarrow$  X $^{-1}$  ( $1 \leq i < n$ )
      Y  $\leftarrow$   $s_{i+1}$ Y :
      GOTO C :
[ $A_n$ ] X  $\leftarrow$  X $^{-1}$ 
      Y  $\leftarrow$   $s_1$ Y
      GOTO B
[C]  IF X ENDS  $s_i$  GOTO  $D_i$  ( $1 \leq i \leq n$ )
      GOTO E
[ $D_i$ ] X  $\leftarrow$  X $^{-1}$  ( $1 \leq i \leq n$ )
      Y  $\leftarrow$   $s_i$ Y :
      GOTO C :
```

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

14

A Progr. Language for String Computations

Example 2: An \mathcal{L}_n program that computes $g(x) = x^{-1}$

```
[B]  IF X ENDS  $s_i$  GOTO  $A_i$  ( $1 \leq i \leq n$ )
      GOTO E
[ $A_i$ ] X  $\leftarrow$  X $^{-1}$  ( $1 < i \leq n$ )
      Y  $\leftarrow$   $s_{i-1}$ Y :
      GOTO C :
[ $A_1$ ] X  $\leftarrow$  X $^{-1}$ 
      IF  $X \neq \epsilon$  GOTO  $C_2$ 
      GOTO E
[ $C_2$ ] Y  $\leftarrow$   $s_n$ Y
      GOTO B
[C]  IF X ENDS  $s_i$  GOTO  $D_i$  ( $1 \leq i \leq n$ )
      GOTO E
[ $D_i$ ] X  $\leftarrow$  X $^{-1}$  ( $1 \leq i \leq n$ )
      Y  $\leftarrow$   $s_i$ Y :
      GOTO C :
```

November 19, 2009

Theory of Computation
Lecture 19: Calculations on Strings IV

15