Register Machine Interpreter

Using Haskell

Johnnie Chang, Wei-Lin Wu

UC Santa Cruz, Computer Science

May 31, 2018

Outline

- Introduction to Register Machines
- A Sample Program: String Reversing
- Demo
- Summary and Future Work

Register Machines

Definition

A register machine is a computational model consisting of:

- A nonempty alphabet $\mathcal{A} = \{a_0, \ldots, a_r\}$ of characters
- Registers $\mathrm{R0}, \mathrm{R1}, \mathrm{R2}, \ldots$
  - Each register contains an arbitrarily long finite string of characters, and acts like a stack
- A program
- An infinitely long output tape

Programs

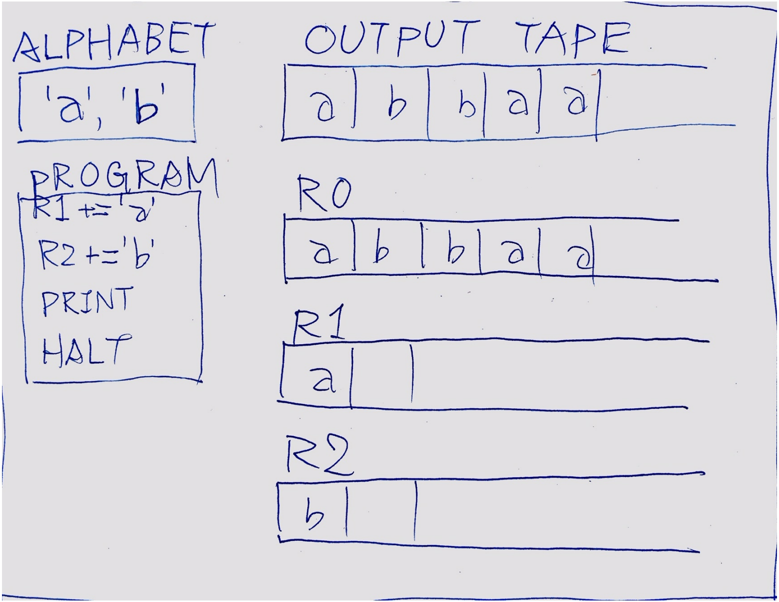
Definition

A program is a finite sequence $\langle I_0, I_1, \ldots, I_n \rangle$ of instructions:

Add:	LET $\mathrm{R}i \mathrel{+}= c$
Sub:	LET $\mathrm{R}i \mathrel{-}= c$
Jump:	IFEMPTY $\mathrm{R}i$ THEN $\mathrm{L}\epsilon$ ELSE $\mathrm{L}0$ OR $\mathrm{L}1$ $\ldots$OR $\mathrm{L}r$
Print:	PRINT
Halt:	HALT

- Only the last instruction, I_n, is a halt.
- The instructions in a program are executed in the same order as they are in the program, except that a jump instruction designates the next instruction to execute.
- Prior to execution of the program, every register contains the empty string except possibly $\mathrm{R}0$, which may contain the input string.

Visualization



```
R1 += 'a'
R2 += 'b'
PRINT
HALT
```

Add Instructions

Definition

An add instruction has the form:

$$\text{LET } \mathrm{R}i \mathrel{+}= c$$

where $\mathrm{R}i$ is the ith register and c is a symbol. This instruction appends c to the string stored in $\mathrm{R}i$.

Example

If the string in $\mathrm{R2}$ is “aab”, then it becomes “aaba” after execution of

$$\text{LET } \mathrm{R2} \mathrel{+}= \text{'a'}$$

Sub Instructions

Definition

A sub instruction has the form:

$$\text{LET } \mathrm{R}i -= c$$

where $\mathrm{R}i$ is the ith register and c is a symbol. This instruction removes it if the string stored in $\mathrm{R}i$ ends with c, otherwise leaves the string unchanged.

Example

If the string in $\mathrm{R}2$ is “aaba” then it remains unchanged after execution of

$$\text{LET } \mathrm{R}2 -= \text{'b'}$$

and becomes “aab” after execution of

$$\text{LET } \mathrm{R}2 -= \text{'a'}$$

Jump Instructions

Definition

A jump instruction has the form:

$$\text{IFEMPTY } \mathrm{R}i \text{ THEN } \mathrm{L}\epsilon \text{ ELSE } \mathrm{L}0 \text{ OR } \mathrm{L}1 \text{ OR } \ldots \text{OR } \mathrm{L}r$$

provided that the underlying alphabet $\mathcal{A} = \{a_0, a_1, \ldots, a_r\}$. The numbers $\mathrm{L}\epsilon, \mathrm{L}0, \mathrm{L}1, \ldots, \mathrm{L}r$ are integers within 0 and n (inclusive) if the program is $\langle I_0, I_1, \ldots, I_n\rangle$.

This instruction checks the string s in $\mathrm{R}i$:

- If $s = \epsilon$, then the instruction $I_{\mathrm{L}\epsilon}$ is to be executed next.
- If s ends with a_k, then the instruction $I_{\mathrm{L}k}$ is to be executed next.

Jump Instructions (Cont.)

Example

Suppose $\mathcal{A} = \{\text{'a'}, \text{'b'}, \text{'c'}\}$, I_4 is

$$\text{LET } \mathrm{R3} \mathrel{+}= \text{'a'}$$

and I_5 is

$$\text{IFEMPTY } \mathrm{R3} \text{ THEN 8 ELSE 4 OR 5 OR 0}$$

then the next instruction to execute after I_5 is I_4

- The string in $\mathrm{R3}$ just before the execution of I_5 ends with 'a'

Print Instruction

Definition

The print instruction

PRINT

copies the current string in $\mathrm{R0}$ into the output tape.

Example

If the content in the output tape is

"aabba"

and the string in $\mathrm{R0}$ is "bba" prior to the print instruction, then after its execution the content in the output tape becomes

"aabbabba"

and the string in $\mathrm{R0}$ is still "bba", unchanged.

Halt Instruction

The halt instruction

HALT

only occurs at the end of a program, i.e. it is the last instruction. A program halts immediately after it reaches a halt instruction.

- A program may never halt, due to a jump instruction:

  ```
  LET R3 += 'a'
  IFEMPTY R3 THEN 0 ELSE 0 OR 0
  HALT
  ```

  where $\mathcal{A} = \{\text{'a'}, \text{'b'}\}$.

A Sample Program: String Reversing

Let $\mathcal{A} = \{\text{‘a’}, \text{‘b’}\}$, the following program reverses the input string (in $\mathrm{R0}$) and prints it on the output tape.

- Algorithm: Move and reverse $\mathrm{R0}$ to $\mathrm{R1}$, move and reverse $\mathrm{R1}$ to $\mathrm{R2}$, move and reverse $\mathrm{R2}$ back to $\mathrm{R0}$, and finally print and halt.

```
0   IFEMPTY R0 THEN 7 ELSE 1 OR 4
1   LET R0 −= ‘a’
2   LET R1 += ‘a’
3   IFEMPTY R0 THEN 7 ELSE 1 OR 4
4   LET R0 −= ‘b’
5   LET R1 += ‘b’
6   IFEMPTY R0 THEN 7 ELSE 1 OR 4
7   IFEMPTY R1 THEN 14 ELSE 8 OR 11
8   LET R1 −= ‘a’
9   LET R2 += ‘a’
```

A Sample Program: String Reversing (Cont.)

```
10  IFEMPTY R1 THEN 14 ELSE 8 OR 11
11  LET R1 −= ‘b’
12  LET R2 += ‘b’
13  IFEMPTY R1 THEN 14 ELSE 8 OR 11
14  IFEMPTY R2 THEN 21 ELSE 15 OR 18
15  LET R2 −= ‘a’
16  LET R0 += ‘a’
17  IFEMPTY R2 THEN 21 ELSE 15 OR 18
18  LET R2 −= ‘b’
19  LET R0 += ‘b’
20  IFEMPTY R2 THEN 21 ELSE 15 OR 18
21  PRINT
22  HALT
```

Demo

Demo time!

Summary and Future Work

Workflow

- Read in user's register machine program in text file
- Parse the program and convert it into ASTs
- Interpret the program by evaluating the ASTs and display the output

Improvement

- Allow user to specify the alphabet
- Eliminate the use of semicolon (;) as the instruction separator in text file
- Allow single-step execution