Advanced Topics in Theoretical Computer Science

Part 2: Register machines

10.11.2021

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: sofronie@uni-koblenz.de
Contents

• Recapitulation: Turing machines and Turing computability
• Register machines (LOOP, WHILE, GOTO)
• Recursive functions
• The Church-Turing Thesis
• Computability and (Un-)decidability
• Complexity
Contents

- Recapitulation: Turing machines and Turing computability
- Register machines (LOOP, WHILE, GOTO)
- Recursive functions
- The Church-Turing Thesis
- Computability and (Un-)decidability
- Complexity
2. Register Machines

- Register machines (Random access machines)
- LOOP Programs
- WHILE Programs
- GOTO Programs
- Relationships between LOOP, WHILE, GOTO
- Relationships between register machines and Turing machines
2. Register Machines

- Register machines (Random access machines)
- LOOP Programs
- WHILE Programs
- GOTO Programs
- Relationships between LOOP, WHILE, GOTO
- Relationships between register machines and Turing machines
Register Machines

The register machine gets its name from its one or more “registers”:

In place of a Turing machine’s tape and head (or tapes and heads) the model uses multiple, uniquely-addressed registers, each of which holds a single positive integer.
Register Machines

In comparison to Turing machines:

- equally powerful fundament for computability theory
- **Advantage:** Programs are easier to understand
Register Machines

In comparison to Turing machines:

- equally powerful fundament for computability theory
- **Advantage:** Programs are easier to understand

similar to ...

the imperative kernel of programming languages

pseudo-code
Register Machines

Computation of $a \mod b$ (pseudocode)

\[ r := a; \]
\[ \text{while } r \geq b \text{ do} \]
\[ \quad r := r - b \]
\[ \text{end;} \]
\[ \text{return } r \]
Register Machines

Definition: Questions

Which instructions (if, while, goto?)
Register Machines

Definition: Questions

Which instructions (if, while, goto?)

Which data types? (integers? strings?)
Register Machines

Definition: Questions

Which instructions (if, while, goto?)

Which data types? (integers? strings?)

Which data structures? (arrays?)
Register Machines

Definition: Questions

Which instructions (if, while, goto?)

Which data types? (integers? strings?)

Which data structures? (arrays?)

Which atomic instructions?
Register Machines

Definition: Questions

Which instructions (if, while, goto?)

Which data types? (integers? strings?)

Which data structures? (arrays?)

Which atomic instructions?

Which Input/Output?
Register Machines

Settings (Informally)

- **Instruction set:**
  - Various variants:
    - loop or while or if + goto
Register Machines

**Settings** (Informally)

- **Instruction set:**
  - Various variants:
    - loop or while or if + goto

- **Data types:**
  - The natural numbers.
    - This is the only difference to normal computers
Register Machines

Settings (Informally)

- **Instruction set:**
  - Various variants:
    - loop or while or if + goto

- **Data types:**
  - The natural numbers.
    - This is the only difference to normal computers

- **Data structures**
  - Unbounded but finite number of registers denoted $x_1, x_2, x_3 \ldots, x_n$;
    - each register contains a natural number
    - (no arrays, objects, ...)
Register Machines

Settings (Informally)

- Atomic instructions:
  - Increment/Decrement a register
Register Machines

Settings (Informally)

- **Atomic instructions:**
  - Increment/Decrement a register

- **Input/Output**
  - **Input:** $n$ input values in the first $n$ registers
  - All the other registers are 0 at the beginning.
  - **Output:** In register $n+1$. 
Example: LOOP Programs

Syntax

Definition

• **Atomic programs:** For each register $x_i$:
  
  - $x_i := x_i + 1$
  - $x_i := x_i - 1$

  are LOOP instructions and also LOOP programs.
Syntax

Definition

- **Atomic programs**: For each register $x_i$:
  - $x_i := x_i + 1$
  - $x_i := x_i - 1$

  are LOOP instructions and also LOOP programs.

- If $P_1, P_2$ are LOOP programs then
  - $P_1; P_2$ is a LOOP program
Example: LOOP Programs

Syntax

Definition

- **Atomic programs:** For each register $x_i$:
  - $x_i := x_i + 1$
  - $x_i := x_i - 1$

  are LOOP instructions and also LOOP programs.

- If $P_1, P_2$ are LOOP programs then
  - $P_1; P_2$ is a LOOP program

- If $P$ is a LOOP program then
  - loop $x_i$ do $P$ end is a LOOP instruction and a LOOP program.
Example: LOOP Programs

Syntax

Definition

- **Atomic programs:** For each register $x_i$:
  - $x_i := x_i + 1$
  - $x_i := x_i - 1$

  are **LOOP** instructions and also **LOOP** programs.

- If $P_1$, $P_2$ are **LOOP** programs then
  - $P_1; P_2$ is a **LOOP** program

- If $P$ is a **LOOP** program then
  - loop $x_i$ do $P$ end is a **LOOP** program (and a **LOOP** instruction)
Example: **WHILE** Programs

**Syntax**

**Definition**

- **Atomic programs**: For each register $x_i$:
  - $x_i := x_i + 1$
  - $x_i := x_i - 1$

  are **WHILE** instructions and also **WHILE** programs.

- If $P_1, P_2$ are **WHILE** programs then
  - $P_1; P_2$ is a **WHILE** program

- If $P$ is a **WHILE** program then
  - while $x_i \neq 0$ do $P$ end is a **WHILE** program (and a **WHILE** instruction)
**Example: GOTO Programs**

**Syntax** Indexes (numbers for the lines in the program) \( j \geq 0 \)

**Definition**

- **Atomic programs:**
  - \( x_i := x_i + 1 \)
  - \( x_i := x_i - 1 \)

  are GOTO instructions for each register \( x_i \).

- If \( x_i \) is a register and \( j \) is an index then
  - if \( x_i = 0 \) goto \( j \) is a GOTO instruction.

- If \( l_1, \ldots, l_k \) are GOTO instructions and \( j_1, \ldots, j_k \) are indices then
  - \( j_1 : l_1; \ldots; j_k : l_k \) is a GOTO program
Definition
A register machine is a machine consisting of the following elements:

- A finite (but unbounded) number of registers $x_1, x_2, x_3, \ldots, x_n$; each register contains a natural number.
- A LOOP-, WHILE- or GOTO-program.
Definition (State of a register machine)

The state $s$ of a register machine is a map:

$$s : \{x_i \mid i \in \mathbb{N}\} \rightarrow \mathbb{N}$$

which associates with every register a natural number as value.
Register Machines: State

Definition (Initial state; Input)

Let $m_1, \ldots, m_k \in \mathbb{N}$ be given as input to a register machine. In the input state $s_0$ we have

- $s_0(x_i) = m_i$ for all $1 \leq i \leq k$
- $s_0(x_i) = 0$ for all $i > k$
**Register Machines: State**

**Definition (Initial state; Input)**
Let \( m_1, \ldots, m_k \in \mathbb{N} \) be given as input to a register machine. In the input state \( s_0 \) we have
- \( s_0(x_i) = m_i \) for all \( 1 \leq i \leq k \)
- \( s_0(x_i) = 0 \) for all \( i > k \)

**Definition (Output)**
If a register machine started with the input \( m_1, \ldots, m_k \in \mathbb{N} \) halts in a state \( s_{\text{term}} \) then:

\[
s_{\text{term}}(x_{k+1})
\]

is the output of the machine.