Automata Learning
Reading Group

Different types of Automata

Jari Stenman
2013-11-26
Introduction

• Purpose:
  • Introduce (review?) some automata models
  • Show a simple but very useful model for undecidability results
Overview

• Turing Machines
• Linear Bounded Automata
• Pushdown Automata
• Counter Machines
Overview

• Turing Machines
• Linear Bounded Automata
• Pushdown Automata
• Counter Machines
Turing Machines

Mandatory Picture
Turing Machines

- Finite control
- Infinite storage tape
Turing Machines

- Finite control
- Infinite storage tape
Turing Machines

- Finite control
- Infinite storage tape
Turing Machines

- Finite control
- Infinite storage tape

... # b b a b # ...

start

b/b,R a/b,R

#/#,L
Turing Machines

- Finite control
- Infinite storage tape
Turing Machines

- Finite control
- Infinite storage tape
Turing Machines

- Finite control
- Infinite storage tape
Turing Machines

- Finite control
- Infinite storage tape
Turing Machines

- Finite control
- Infinite storage tape
- Capture the intuitive notion of computation
- “Everything” undecidable
<table>
<thead>
<tr>
<th>Year</th>
<th>Inventor(s)</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1936</td>
<td>Turing</td>
<td>Turing Machines</td>
</tr>
<tr>
<td>1943</td>
<td>McCulloch &amp; Pitts</td>
<td>Finite Automata</td>
</tr>
<tr>
<td>1955, 1956</td>
<td>Mealy, Moore</td>
<td>Finite State Transducers</td>
</tr>
<tr>
<td>1956</td>
<td>Chomsky</td>
<td>Chomsky Hierarchy</td>
</tr>
<tr>
<td>1960-1964</td>
<td>Myhill, Landweber, Kuroda</td>
<td>Linear Bounded Automata</td>
</tr>
</tbody>
</table>
Chomsky Hierarchy

- Turing Machines
- Linear Bounded Automata
- Pushdown Automata
- Finite Automata
Overview

• Turing Machines
• Linear Bounded Automata
• Pushdown Automata
• Counter Machines
Linear Bounded Automata

- Turing Machines with bounded tape

  ... # L a b a b R # ...

- Nondet. LBA accept exactly the context-sensitive languages

Tuesday 26 November 13
Linear Bounded Automata

- Turing Machines with bounded tape

```
... # L a b a b R # ...
```

- Nondet. LBA accept exactly the context-sensitive languages

- **Open problem:** Are det. LBA and nondet. LBA equivalent?
Overview

• Turing Machines
• Linear Bounded Automata
• Pushdown Automata
• Counter Machines
Pushdown Automata

- Finite control
- Unbounded stack

Input: abab

Diagram:
- Start state
- Transition: push(#) from start
- Transition: a, push(a)
- Transition: b, pop(a)
- Transition: pop(#)

Tuesday 26 November 13
Pushdown Automata

- Finite control
- Unbounded stack

Input: \texttt{abab}

Diagram:
- Start state
- Transition to \texttt{push(\#)}
- Transition to \texttt{a,push(a)}
- Transition to \texttt{b, pop(a)}
- Transition to \texttt{pop(\#)}
Pushdown Automata

- Finite control
- Unbounded stack

Input: bab

Diagram:
- Start
- Push (#)
- a, push(a)
- b, pop(a)
- pop(#)

Stack:
- ...
Pushdown Automata

- Finite control
- Unbounded stack

Input:

Tuesday 26 November 13
Pushdown Automata

- Finite control
- Unbounded stack

Input:

```
+---+---+---+
|   |   |   |
|   | a |   |
|   | # |   |
+---+---+---+
```

```
+---+---+---+
|   |   |   |
|   | # |   |
+---+---+---+
```
Pushdown Automata

- Finite control
- Unbounded stack

Input: Tuesday 26 November 13
Pushdown Automata

- Finite control
- Unbounded stack

Input:

```
start
push(#)
```

```
a,push(a)
```

```
b,pop(a)
```

```
pop(#)
```

```
```
Pushdown Automata

- Finite control
- Unbounded stack
- Nondet. PDA accept exactly the context-free languages
Pushdown Automata

• Finite control
• Unbounded stack
• Nondet. PDA accept exactly the context-free languages
  • What happens if we have several stacks?
Pushdown Automata

2 stacks
Pushdown Automata

2 stacks
Pushdown Automata

2 stacks
Pushdown Automata

2 stacks
Pushdown Automata

2 stacks

... # a a c # ...

Tuesday 26 November 13
Pushdown Automata

2 stacks

... # a c c # ...

Tuesday 26 November 13
Pushdown Automata

2 stacks

2-stack PDA are Turing-powerful!
Overview

- Turing Machines
- Linear Bounded Automata
- Pushdown Automata
- Counter Machines
Counter Machines

- Finite control
- Counters storing natural numbers

\[ a, x_1+ \]
\[ b, x_2+ \]
\[ x_1=0? \]
\[ x_2=0? \]
Counter Machines

• Finite control
• Counters storing natural numbers

How powerful is this model?
Simulating a Stack

- A 2-CM can simulate a 1-stack PDA:
- Encode stack as natural number
- Implement push and pop for each stack symbol as operations on this number
Simulating a Stack

Example: $\Sigma = \{a, b\}$

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Push $a$</td>
<td>Double $x_1$ and add 1</td>
</tr>
<tr>
<td>Push $b$</td>
<td>Double $x_1$ and add 2</td>
</tr>
<tr>
<td>Pop $a$</td>
<td>Check if $x_1-1 \equiv 0 \text{ mod } 2$</td>
</tr>
<tr>
<td>Pop $b$</td>
<td>Check if $x_1-2 \equiv 0 \text{ mod } 2$</td>
</tr>
</tbody>
</table>

Stack:  

Counter: 0
Simulating a Stack

Example: $\Sigma = \{a, b\}$

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push $a$</td>
<td>Double $x_1$ and add 1</td>
</tr>
<tr>
<td>Push $b$</td>
<td>Double $x_1$ and add 2</td>
</tr>
<tr>
<td>Pop $a$</td>
<td>Check if $x_1 - 1 \equiv 0 \text{ mod } 2$</td>
</tr>
<tr>
<td>Pop $b$</td>
<td>Check if $x_1 - 2 \equiv 0 \text{ mod } 2$</td>
</tr>
</tbody>
</table>

Stack: 

Counter: 0 1

Tuesday 26 November 13
Simulating a Stack

Example: $\Sigma = \{a, b\}$

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push $a$</td>
<td>Double $x_1$ and add 1</td>
</tr>
<tr>
<td>Push $b$</td>
<td>Double $x_1$ and add 2</td>
</tr>
<tr>
<td>Pop $a$</td>
<td>Check if $x_1 - 1 \equiv 0 \mod 2$</td>
</tr>
<tr>
<td>Pop $b$</td>
<td>Check if $x_1 - 2 \equiv 0 \mod 2$</td>
</tr>
</tbody>
</table>

Stack: 

Counter: 0 1 3

Tuesday 26 November 13
Simulating a Stack

Example: $\Sigma = \{a, b\}$

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push $a$</td>
<td>Double $x_1$ and add 1</td>
</tr>
<tr>
<td>Push $b$</td>
<td>Double $x_1$ and add 2</td>
</tr>
<tr>
<td>Pop $a$</td>
<td>Check if $x_1 - 1 \equiv 0 \mod 2$</td>
</tr>
<tr>
<td>Pop $b$</td>
<td>Check if $x_1 - 2 \equiv 0 \mod 2$</td>
</tr>
</tbody>
</table>

Decrement $x_1$ while adding 2 to $x_2$, and finally add 1 to $x_2$. Copy $x_2$ to $x_1$ and reset $x_2$. 

Stack: \[ \text{Stack: } \]

Counter: \[ \text{Counter: } 0 \quad 1 \quad 3 \]
Simulating a Stack

Example: $\Sigma = \{a, b\}$

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push $a$</td>
<td>Double $x_1$ and add 1</td>
</tr>
<tr>
<td>Push $b$</td>
<td>Double $x_1$ and add 2</td>
</tr>
<tr>
<td>Pop $a$</td>
<td>Check if $x_1 - 1 \equiv 0 \mod 2$</td>
</tr>
<tr>
<td>Pop $b$</td>
<td>Check if $x_1 - 2 \equiv 0 \mod 2$</td>
</tr>
</tbody>
</table>

Stack: $\square \rightarrow a \rightarrow a \rightarrow a \rightarrow a \rightarrow a$

Counter: $0 \rightarrow 1 \rightarrow 3 \rightarrow 8$
Simulating a Stack

Example: $\Sigma = \{a, b\}$

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push $a$</td>
<td>Double $x_1$ and add 1</td>
</tr>
<tr>
<td>Push $b$</td>
<td>Double $x_1$ and add 2</td>
</tr>
<tr>
<td>Pop $a$</td>
<td>Check if $x_1 - 1 \equiv 0 \mod 2$</td>
</tr>
<tr>
<td>Pop $b$</td>
<td>Check if $x_1 - 2 \equiv 0 \mod 2$</td>
</tr>
</tbody>
</table>

First, decrement by 1. Decrement $x_1$ by 2 while adding 1 to $x_2$, until no longer possible. Now, $x_1$ contains remainder and $x_2$ quotient. If $x_1$ is 0, set $x_1$ to $x_2$. Otherwise, abort.

Stack:  
Counter: 0 1 3 8 3
Simulating a Stack

Example: $\Sigma = \{a, b\}$

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push $a$</td>
<td>Double $x_1$ and add 1</td>
</tr>
<tr>
<td>Push $b$</td>
<td>Double $x_1$ and add 2</td>
</tr>
<tr>
<td>Pop $a$</td>
<td>Check if $x_1 - 1 \equiv 0 \mod 2$</td>
</tr>
<tr>
<td>Pop $b$</td>
<td>Check if $x_1 - 2 \equiv 0 \mod 2$</td>
</tr>
</tbody>
</table>

Stack:  

Counter: 0 1 3 8 3

Tuesday 26 November 13
Counter Machines

- A 2-CM can simulate a 1-stack PDA.
- A 4-CM can simulate a 2-stack PDA.
- So 4-CM are Turing powerful!
Counter Machines

- But there is more:
  - A 2-CM can simulate an n-CM for any n.
  - Encode $k_1,k_2,k_3,k_4$ as $2^{k_1}3^{k_2}5^{k_3}7^{k_4}$
Counter Machines

• But there is more:
  • A 2-CM can simulate an n-CM for any n.
  • Encode $k_1, k_2, k_3, k_4$ as $2^{k_1}3^{k_2}5^{k_3}7^{k_4}$
  • 2-CM are Turing powerful!
Counter Machines

• Easy way to prove undecidability: show that if your problem is decidable, you can solve some undecidable problem for 2-CM.

• Simulate 3 operations (for each counter):
  • Check for 0
  • Increment counter
  • Decrement counter
Thank You!