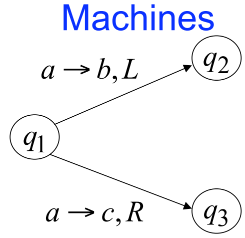
CS 301 - Lecture 22
Non-Determinism, Universal Turing Machine and Linear Bounded Automata

Fall 2008

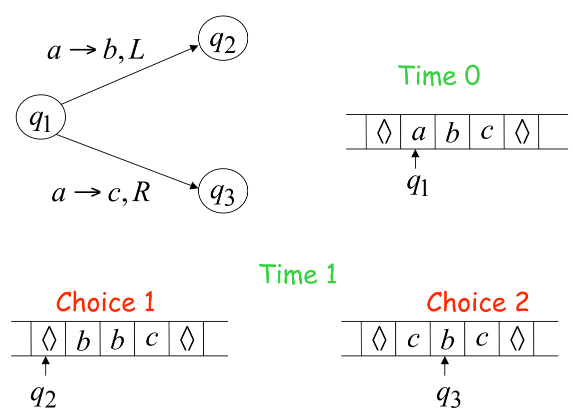
Review

- Languages and Grammars
  - Alphabets, strings, languages
- Regular Languages
  - Deterministic Finite and Nondeterministic Automata
  - Equivalence of NFA and DFA and Minimizing a DFA
  - Regular Expressions
  - Regular Grammars
  - Properties of Regular Languages
  - Languages that are not regular and the pumping lemma
- Context Free Languages
  - Context Free Grammars
  - Derivations: leftmost, rightmost and derivation trees
  - Parsing and ambiguity
  - Simplifications and Normal Forms
  - Nondeterministic Pushdown Automata
  - Pushdown Automata and Context Free Grammars
  - Deterministic Pushdown Automata
  - Pumping Lemma for context free grammars
  - Properties of Context Free Grammars
- Turing Machines
  - Definition, Accepting Languages, and Computing Functions
  - Combining Turing Machines and Turing's Thesis
  - Turing Machine Variations
  - Today: Non-Determinism, Universal Turing Machine and Linear Bounded Automata

NonDeterministic Turing Machines



Non Deterministic Choice



Input string w is accepted if
this a possible computation

$$q_0 w \overset{*}{\succ} x \, q_f \, y$$

Initial configuration

Final Configuration

Final state

NonDeterministic Machines simulate
Standard (deterministic) Machines:

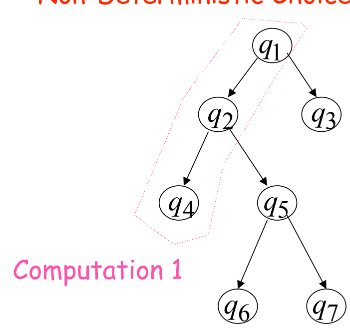
Every deterministic machine
is also a nondeterministic machine

Deterministic machines simulate
NonDeterministic machines:

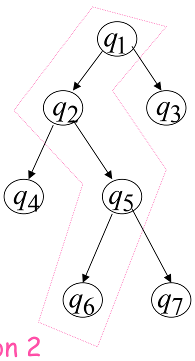
Deterministic machine:

Keeps track of all possible computations

Non-Deterministic Choices



Non-Deterministic Choices



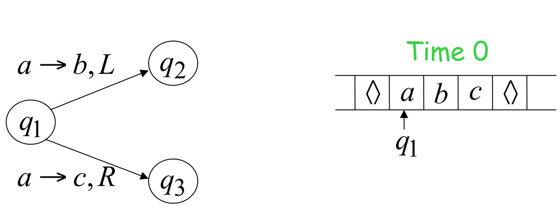
Computation 2

Simulation

Deterministic machine:

- Keeps track of all possible computations
- Stores computations in a two-dimensional tape

NonDeterministic machine

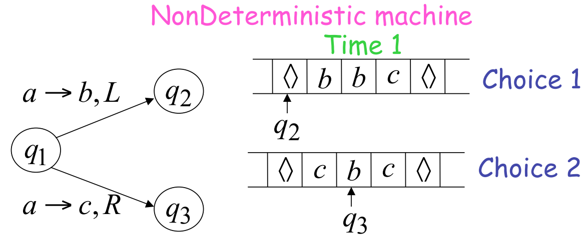


Deterministic machine

	#	#	#	#	#	#	
	#	a	b	c	#		
	#	q_1			#		
	#	#	#	#	#		

Computation 1

NonDeterministic machine



Deterministic machine

	#	#	#	#	#	#	
#		b	b	c	#		
#	q_2				#		
#		c	b	c	#		
#			q_3		#		

Computation 1

Computation 2

Repeat

- Execute a step in each computation:
- If there are two or more choices in current computation:
    1. Replicate configuration
    2. Change the state in the replica

Theorem: NonDeterministic Machines have the same power with Deterministic machines

Remark:
The simulation in the Deterministic machine takes time exponential time compared to the NonDeterministic machine

A Universal Turing Machine

A limitation of Turing Machines:

Turing Machines are "hardwired"

they execute
only one program

Real Computers are re-programmable

Solution: Universal Turing Machine

Attributes:

- Reprogrammable machine
- Simulates any other Turing Machine

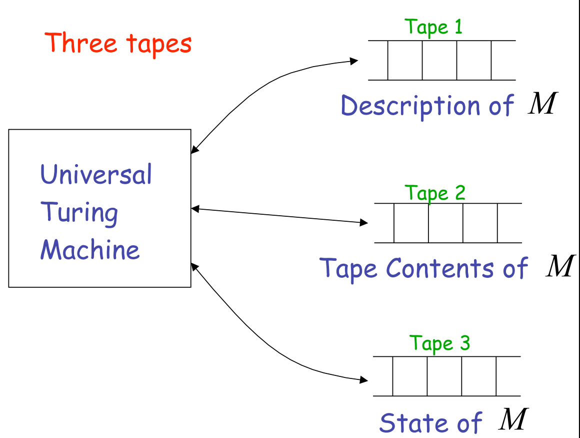
Universal Turing Machine
simulates any other Turing Machine M

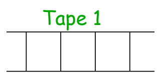
Input of Universal Turing Machine:

Description of transitions of M

Initial tape contents of M

Three tapes





Description of M

We describe Turing machine M
as a string of symbols:

We encode M as a string of symbols

Alphabet Encoding

Symbols:	a	b	c	d	$\cdots$
Encoding:	1	11	111	1111	

State Encoding

States:	q_1	q_2	q_3	q_4	$\cdots$
Encoding:	1	11	111	1111	

Head Move Encoding

Move:	L	R
Encoding:	1	11

Transition Encoding

Transition: $\delta(q_1, a) = (q_2, b, L)$

Encoding: 10101101101

separator

Machine Encoding

Transitions:

$\delta(q_1, a) = (q_2, b, L)$ $\delta(q_2, b) = (q_3, c, R)$

Encoding:

1 0 1 0 11 0 11 0 1 00 11 0 11 0 111 0 111 0 11

separator

Tape 1 contents of Universal Turing Machine:

encoding of the simulated machine M
as a binary string of 0's and 1's

A Turing Machine is described
with a binary string of 0's and 1's

Therefore:

The set of Turing machines forms a language:

each string of the language is
the binary encoding of a Turing Machine

Language of Turing Machines

L = { 010100101, (Turing Machine 1)

00100100101111, (Turing Machine 2)

111010011110010101, ……

…… }

How Many Turing Machines Are There?

We now have a language L:

Each string in L is a Turing Machine!

How big is this language?

Equivalently…

how many Turing machines are there?

how many valid Java programs are there?

Not finite…. Are there degrees of infinity?

Countable Sets

Infinite sets are either: Countable

or

Uncountable

Countable set:

Any finite set

or

Any *Countably infinite* set:

There is a one to one correspondence

between

elements of the set

and

Natural numbers

Example: The set of even integers is countable

Even integers: $0,\ 2,\ 4,\ 6,\ \ldots$

Correspondence: $\downarrow\ \downarrow\ \downarrow\ \downarrow$

Positive integers: $1,\ 2,\ 3,\ 4,\ \ldots$

$2n$ corresponds to $n+1$

Example: The set of rational numbers is countable

Rational numbers: $\frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \ldots$

Naïve Proof

Rational numbers: $\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \ldots$

Correspondence: $\downarrow\ \downarrow\ \downarrow$

Positive integers: $1,\ 2,\ 3,\ \ldots$

Doesn't work:

we will never count numbers with nominator 2: $\frac{2}{1}, \frac{2}{2}, \frac{2}{3}, \ldots$

Better Approach

$$
\begin{array}{ccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\
\frac{3}{1} & \frac{3}{2} & \cdots & & \\
\frac{4}{1} & \cdots & & &
\end{array}
$$

$$\begin{array}{ccccc} \frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\ \frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\ \frac{3}{1} & \frac{3}{2} & \cdots & & \\ \frac{4}{1} & \cdots & & & \end{array}$$

$$\begin{array}{ccccc} \frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\ \frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\ \frac{3}{1} & \frac{3}{2} & \cdots & & \\ \frac{4}{1} & \cdots & & & \end{array}$$

$$\begin{array}{ccccc} \frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\ \frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\ \frac{3}{1} & \frac{3}{2} & \cdots & & \\ \frac{4}{1} & \cdots & & & \end{array}$$

$$\begin{array}{ccccc} \frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\ \frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\ \frac{3}{1} & \frac{3}{2} & \cdots & & \\ \frac{4}{1} & \cdots & & & \end{array}$$

$$\begin{array}{cccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\
\frac{3}{1} & \frac{3}{2} & \cdots & & \\
\frac{4}{1} & \cdots & & &
\end{array}$$

Rational Numbers: $\frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{1}{3}, \frac{2}{2}, \ldots$

Correspondence:

Positive Integers: $1, 2, 3, 4, 5, \ldots$

We proved:

the set of rational numbers is countable by describing an enumeration procedure

Definition

Let S be a set of strings

An **enumeration procedure** for S is a Turing Machine that generates all strings of S one by one

and

Each string is generated in finite time

strings $s_1, s_2, s_3, \ldots \in S$

Enumeration Machine for S $\xrightarrow{\text{output (on tape)}}$ $s_1, s_2, s_3, \ldots$

Finite time: $t_1, t_2, t_3, \ldots$

Observation:

If for a set there is an enumeration procedure, then the set is countable

Example:

The set of all strings $\{a,b,c\}^+$ is countable

Proof:

We will describe an enumeration procedure

Naive procedure:

Produce the strings in lexicographic order:

a
aa
aaa
$aaaa$
......

Doesn't work:

strings starting with b will never be produced

Better procedure: **Proper Order**

1. Produce all strings of length 1

2. Produce all strings of length 2

3. Produce all strings of length 3

4. Produce all strings of length 4

....

Produce strings in **Proper Order**:

$\left.\begin{array}{l} a \\ b \\ c \end{array}\right\}$ length 1

$\left.\begin{array}{l} aa \\ ab \\ ac \\ ba \\ bb \\ bc \\ ca \\ cb \\ cc \end{array}\right\}$ length 2

$\left.\begin{array}{l} aaa \\ aab \\ aac \\ \ldots\ldots \end{array}\right\}$ length 3

Theorem: The set of all Turing Machines is countable

Proof: Any Turing Machine can be encoded with a binary string of 0's and 1's

Find an enumeration procedure for the set of Turing Machine strings

Enumeration Procedure:

Repeat

1. Generate the next binary string of 0's and 1's in proper order

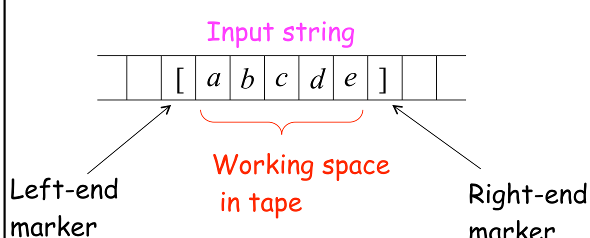
2. Check if the string describes a Turing Machine
   if **YES:** print string on output tape
   if **NO:** ignore string

Linear Bounded
Automata
LBAs

Linear Bounded Automata (LBAs)
are the same as Turing Machines
with one difference:

The input string tape space
is the only tape space allowed to use

Linear Bounded Automaton (LBA)



All computation is done between end markers

We define LBA's as NonDeterministic

Open Problem:

NonDeterministic LBA's
have same power with
Deterministic LBA's ?

Example languages accepted by LBAs:

$$L = \{a^n b^n c^n\}$$

$$L = \{a^{n!}\}$$

LBA's have more power than NPDA's

LBA's have also less power than Turing Machines

What's Next

- Read
  - Linz Chapter 1,2.1, 2.2, 2.3, (skip 2.4), 3, 4, 5, 6.1, 6.2, (skip 6.3), 7.1, 7.2, 7.3, (skip 7.4), 8, 9, 10, 11.1, and 11.2
  - JFLAP Chapter 1, 2.1, (skip 2.2), 3, 4, 5, 6, 7, (skip 8), 9, (skip 10), 11.1
- Next Lecture Topics From 11.1
  - Recursive Languages and Recursively Enumerable Languages
- Quiz 3 in Recitation on Wednesday 11/12
  - Covers Linz 7.1, 7.2, 7.3, (skip 7.4), 8, and JFLAP 5,6,7
  - Closed book, but you may bring one sheet of 8.5 x 11 inch paper with any notes you like.
  - Quiz will take the full hour
- Homework
  - Homework Due Today
  - New Homework Available by Friday Morning
  - New Homework Due Next Thursday