CS21
Decidability and Tractability

Lecture 4

January 12, 2015

Outline

- Pumping Lemma
- Pushdown Automata
- Context-Free Grammars and Languages

Regular expressions and FA

- **Theorem**: a language L is recognized by a FA iff L is described by a regular expr.
- Languages recognized by a FA are called regular languages.
- Rephrasing what we know so far:
  - regular languages closed under 3 operations
  - NFA recognize exactly the regular languages
  - regular expressions describe exactly the regular languages

Limits on the power of FA

- Is *every* language describable by a sufficiently complex regular expression?
- If someone asks you to design a FA for a language that seems hard, how do you know when to give up?
- Is this language regular?

{w : w has an equal # of "01" and "10" substrings}

Limits on the power of FA

- Intuition:
  - FA can only remember finite amount of information. They cannot count
  - languages that "entail counting" should be non-regular…
- Intuition not enough:

{w : w has an equal # of "01" and "10" substrings}

$= 0\Sigma^*0 \cup 1\Sigma^*1$

but {w: w has an equal # of "0" and "1" substrings} is not regular!

Limits on the power of FA

How do you *prove* that there is *no* Finite Automaton recognizing a given language?

Non-regular languages

Pumping Lemma: Let L be a regular language. There exists an integer p ("pumping length") for which every $w \in L$ with $|w| \geq p$ can be written as

$$w = xyz \quad \text{such that}$$

1. for every $i \geq 0$, $xy^iz \in L$, and
2. $|y| > 0$, and
3. $|xy| \leq p$.

Non-regular languages

- Using the Pumping Lemma to prove L is not regular:
  - assume L is regular
  - then there exists a pumping length p
  - select a string $w \in L$ of length at least p
  - argue that for every way of writing w = xyz that satisfies (2) and (3) of the Lemma, pumping on y yields a string not in L.
  - contradiction.

Pumping Lemma Examples

- Theorem: $L = \{0^n1^n : n \geq 0\}$ is not regular.
- Proof:
  - let p be the pumping length for L
  - choose $w = 0^p1^p$

$$w = \underbrace{000000000\ldots0}_{p}\underbrace{111111111\ldots1}_{p}$$

  - $w = xyz$, with $|y| > 0$ and $|xy| \leq p$.

Pumping Lemma Examples

- 3 possibilities:

$$w = \underbrace{00000}_{x}\underbrace{0000}_{y}\underbrace{0\ldots0111111111\ldots1}_{z}$$

$$w = \underbrace{000000000\ldots011}_{x}\underbrace{11111}_{y}\underbrace{11\ldots1}_{z}$$

$$w = \underbrace{0000000}_{x}\underbrace{00\ldots011111}_{y}\underbrace{1111\ldots1}_{z}$$

- in each case, pumping on y gives a string not in language L.

Pumping Lemma Examples

- Theorem: L = {w: w has an equal # of 0s and 1s} is not regular.
- Proof:
  - let p be the pumping length for L
  - choose $w = 0^p1^p$

$$w = \underbrace{000000000\ldots0}_{p}\underbrace{111111111\ldots1}_{p}$$

  - $w = xyz$, with $|y| > 0$ and $|xy| \leq p$.

Pumping Lemma Examples

- 3 possibilities:

$$w = \underbrace{00000}_{x}\underbrace{0000}_{y}\underbrace{0\ldots0111111111\ldots1}_{z}$$

$$w = \underbrace{000000000\ldots011}_{x}\underbrace{11111}_{y}\underbrace{11\ldots1}_{z}$$

$$w = \underbrace{000000000\ldots}_{x}\underbrace{01}_{y}\underbrace{11111111\ldots1}_{z}$$

- first 2 cases, pumping on y gives a string not in language L; 3rd case a problem!

Pumping Lemma Examples

– recall condition 3: $|xy| \leq p$

– since $w = 0^p1^p$ we know more about how it can be divided, and this case cannot arise:

$$w = \underbrace{000000000\ldots}_{x}\underbrace{01}_{y}\underbrace{111111111\ldots1}_{z}$$

– so we do get a contradiction.

– conclude that L is not regular.

Pumping Lemma Examples

- Theorem: $L = \{0^i1^j : i > j\}$ is not regular.
- Proof:
  - let p be the pumping length for L
  - choose $w = 0^{p+1}1^p$

$$w = \underbrace{000000000\ldots0}_{p+1}\underbrace{1111111\ldots1}_{p}$$

  - $w = xyz$, with $|y| > 0$ and $|xy| \leq p$.

Pumping Lemma Examples

– 1 possibility:

$$w = \underbrace{00000}_{x}\underbrace{0000}_{y}\underbrace{0\ldots0111111111\ldots1}_{z}$$

– pumping on y gives strings in the language (?)

– this seems like a problem…

– Lemma states that for every $i \geq 0$, $xy^iz \in L$

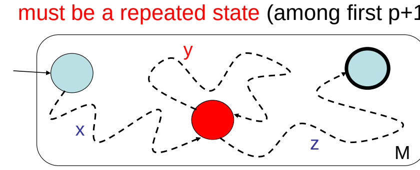
– xy^0z not in L. So L not regular.

Proof of the Pumping Lemma

– Let M be a FA that recognizes L.

– Set p = number of states of M.

– Consider $w \in L$ with $|w| \geq p$. On input w, M must go through *at least* p+1 states. There must be a repeated state (among first p+1).



FA Summary

- A “problem” is a language
- A “computation” receives an input and either accepts, rejects, or loops forever.
- A “computation” recognizes a language (it may also decide the language).
- Finite Automata perform simple computations that read the input from left to right and employ a finite memory.

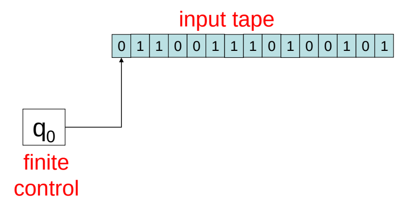
FA Summary

- The languages recognized by FA are the regular languages.
- The regular languages are closed under union, concatenation, and star.
- Nondeterministic Finite Automata may have several choices at each step.
- NFAs recognize exactly the same languages that FAs do.

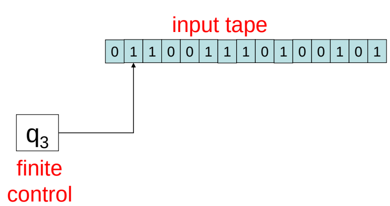
FA Summary

- Regular expressions are languages built up from the operations union, concatenation, and star.
- Regular expressions describe exactly the same languages that FAs (and NFAs) recognize.
- Some languages are not regular. This can be proved using the Pumping Lemma.

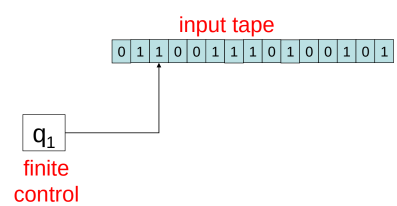
Machine view of FA



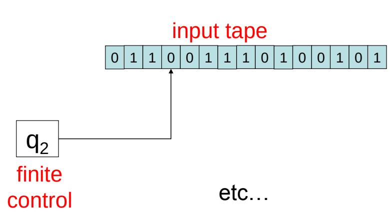
Machine view of FA



Machine view of FA



Machine view of FA

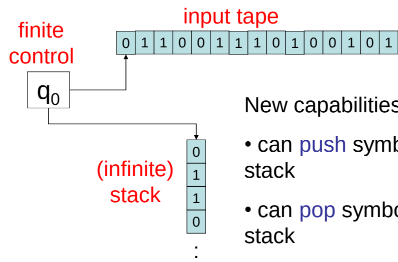


etc…

A more powerful machine

- limitation of FA related to fact that they can only “remember” a bounded amount of information
- What is the simplest alteration that adds unbounded “memory” to our machine?
- Should be able to recognize, e.g., $\{0^n1^n: n \geq 0\}$

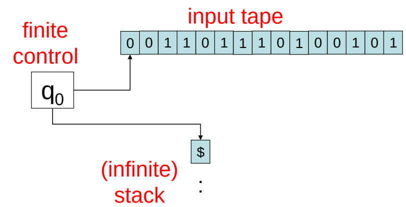
Pushdown Automata



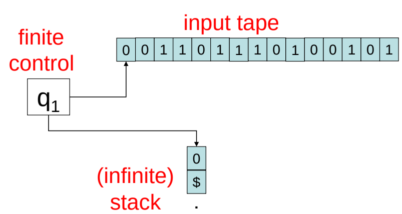
New capabilities:

- can push symbol onto stack
- can pop symbol off of stack

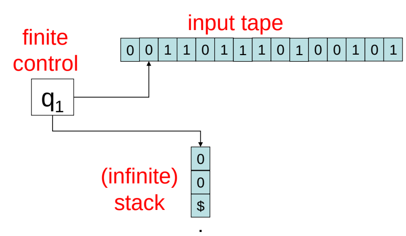
Pushdown Automata



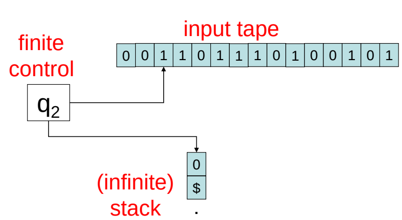
Pushdown Automata



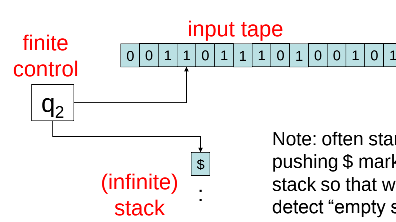
Pushdown Automata



Pushdown Automata



Pushdown Automata



Note: often start by pushing \$ marker onto stack so that we can detect “empty stack”

Pushdown Automata (PDA)

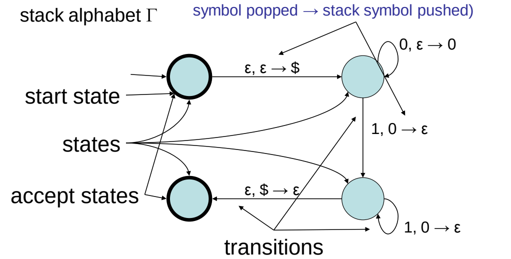
- We will define nondeterministic pushdown automata immediately
  - potentially several choices of “next step”
- Deterministic PDA defined later
  - weaker than NPDA
- Two ways to describe NPDA
  - diagram
  - formal definition

NPDA diagram

tape alphabet Σ

stack alphabet Γ

transition label: (tape symbol read, stack symbol popped $\rightarrow$ stack symbol pushed)



NPDA operation

- Taking a transition labeled:

$$a, b \rightarrow c$$

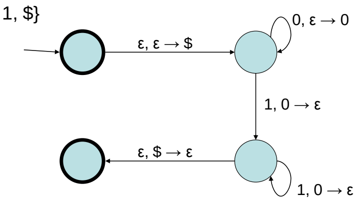
  - $a \in (\Sigma \cup \{\varepsilon\})$
  - $b, c \in (\Gamma \cup \{\varepsilon\})$

  - read a from tape, or don’t read from tape if $a = \varepsilon$
  - pop b from stack, or don’t pop from stack if $b = \varepsilon$
  - push c onto stack, or don’t push onto stack if $c = \varepsilon$

Example NPDA

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \$\}$

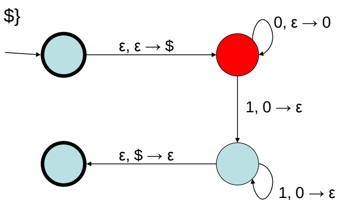


- tape: 0 0 1 1 Stack contents: $

Example NPDA

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \$\}$

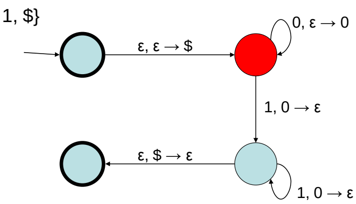


- tape: 0 0 1 1 Stack contents: 0 $

Example NPDA

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \$\}$

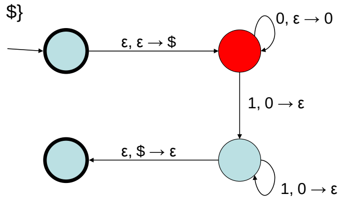


- tape: 0 0 1 1 Stack contents: 0 0 $

Example NPDA

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \$\}$

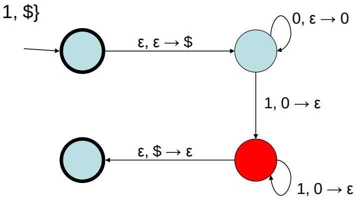


- tape: 0 0 1 1 Stack contents: 0 0 $

Example NPDA

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \$\}$

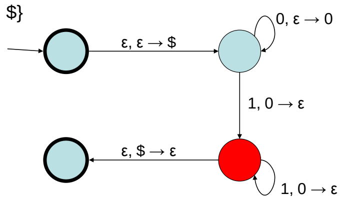


- tape: 0 0 1 1 Stack contents: 0 $

Example NPDA

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \$\}$

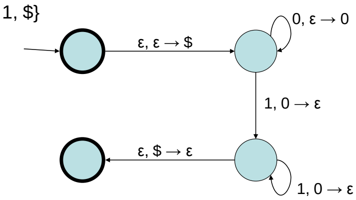


- tape: 0 0 1 1 Stack contents: $
  accepted

Example NPDA

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \$\}$

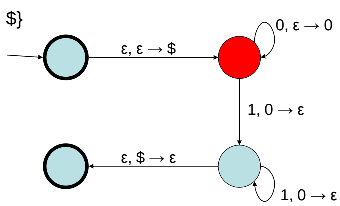


- tape: 0 0 1 Stack contents: $

Example NPDA

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \$\}$

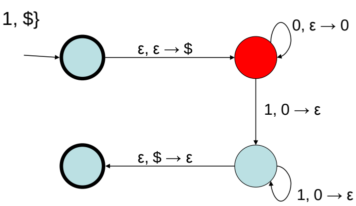


- tape: 0 0 1 Stack contents: 0 $

Example NPDA

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \$\}$

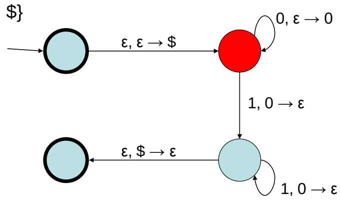


- tape: 0 0 1 Stack contents: 0 0 $

Example NPDA

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \$\}$

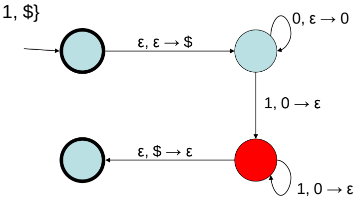


- tape: 0 0 1 Stack contents: 0 0 $

Example NPDA

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \$\}$

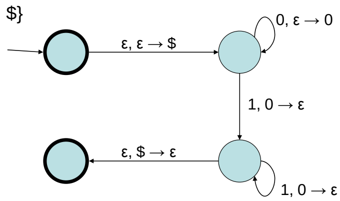


- tape: 0 0 1 Stack contents: 0 $
  not accepted

Example NPDA

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \$\}$



- What language does this NPDA accept?

Formal definition of NPDA

- A NPDA is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where:
  - Q is a finite set called the states
  - Σ is a finite set called the tape alphabet
  - Γ is a finite set called the stack alphabet
  - $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \wp(Q \times (\Gamma \cup \{\varepsilon\}))$ is a function called the transition function
  - q_0 is an element of Q called the start state
  - F is a subset of Q called the accept states

Formal definition of NPDA

- NPDA M = $(Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts string $w \in \Sigma^*$ if w can be written as

  $w_1w_2w_3\ldots w_m \in (\Sigma \cup \{\varepsilon\})^*$, and
- there exist states $r_0, r_1, r_2, \ldots, r_m$, and
- there exist strings $s_0, s_1, \ldots, s_m$ in $(\Gamma\cup\{\varepsilon\})^*$
  - $r_0 = q_0$ and $s_0 = \varepsilon$
  - $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, where $s_i = at$, $s_{i+1} = bt$ for some $t \in \Gamma^*$
  - $r_m \in F$