Outline

- Non-regular languages: Pumping Lemma
- Pushdown Automata
- Context-Free Grammars and Languages

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 $w = xyz$ such that

1. for every $i \geq 0$, $xy^i z \in L$
2. $|y| > 0$
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: $L = \{0^i 1^j : i > j\}$ is not regular.
- Proof:
  - let p be the pumping length for L
  - choose $w = 0^{p+1}1^p$
  - $w = 000000000...011111111...1$
  - $w = xyz$, with $|y| > 0$ and $|xy| \leq p$.

Pumping Lemma Examples

- 1 possibility:
  - $w = 000000000...01111111111...1$
  - pumping on y gives strings in the language (?)
  - this seems like a problem...
  - Lemma states that for every $i \geq 0$, $xyz \in L$
  - $xy^2 z$ 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 ∈ L with |w| ≥ 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

input tape

finite control

Machine view of FA

input tape

finite control
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., \( \{a^n : n \geq 0\} \)

Pushdown Automata

- New capabilities:
  - Can push symbol onto stack
  - Can pop symbol off of stack
Pushdown Automata

Finite control
Input tape
Stack

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 \( \Sigma \)
Stack alphabet \( \Gamma \)

Transition label: (Tape symbol read, stack symbol popped \( \rightarrow \) stack symbol pushed)

NPDA operation

- Taking a transition labeled:
  \[ a, b \rightarrow c \]
  - \( a \in (\Sigma \cup \{\epsilon\}) \)
  - \( b,c \in (\Gamma' \cup \{\epsilon\}) \)
  - read \( a \) from tape, or don't read from tape if \( a = \epsilon \)
  - pop \( b \) from stack, or don't pop from stack if \( b = \epsilon \)
  - push \( c \) onto stack, or don't push onto stack if \( c = \epsilon \)
Example NPDA

\[ \Sigma = \{0, 1\} \]
\[ \Gamma = \{0, 1, \$\} \]

* tape: 0 0 1 1
Stack contents: \$

\[ \varepsilon, \varepsilon \rightarrow \$ \]
\[ 0, \varepsilon \rightarrow 0 \]
\[ 1, 0 \rightarrow \varepsilon \]

\[ \varepsilon, \$ \rightarrow \varepsilon \]
\[ 1, 0 \rightarrow \varepsilon \]

\[ \Sigma = \{0, 1\} \]
\[ \Gamma = \{0, 1, \$\} \]

* tape: 0 0 1 1
Stack contents: 0 $

\[ \varepsilon, \varepsilon \rightarrow \$ \]
\[ 0, \varepsilon \rightarrow 0 \]
\[ 1, 0 \rightarrow \varepsilon \]

\[ \varepsilon, \$ \rightarrow \varepsilon \]
\[ 1, 0 \rightarrow \varepsilon \]

\[ \Sigma = \{0, 1\} \]
\[ \Gamma = \{0, 1, \$\} \]

* tape: 0 0 1 1
Stack contents: 0 0 $

\[ \varepsilon, \varepsilon \rightarrow \$ \]
\[ 0, \varepsilon \rightarrow 0 \]
\[ 1, 0 \rightarrow \varepsilon \]

\[ \varepsilon, \$ \rightarrow \varepsilon \]
\[ 1, 0 \rightarrow \varepsilon \]

\[ \Sigma = \{0, 1\} \]
\[ \Gamma = \{0, 1, \$\} \]

* tape: 0 0 1 1
Stack contents: accepted

\[ \varepsilon, \varepsilon \rightarrow \$ \]
\[ 0, \varepsilon \rightarrow 0 \]
\[ 1, 0 \rightarrow \varepsilon \]

\[ \varepsilon, \$ \rightarrow \varepsilon \]
\[ 1, 0 \rightarrow \varepsilon \]
Example NPDA

\[ \Sigma = \{0, 1\} \]
\[ \Gamma = \{0, 1, \$\} \]

• tape: 0 0 1  Stack contents: $ $

$ \xrightarrow{0, \$ \rightarrow \epsilon} \epsilon $  
$ \xrightarrow{1, \$ \rightarrow \epsilon} \epsilon $  

\[ \Sigma = \{0, 1\} \]
\[ \Gamma = \{0, 1, \$\} \]

• tape: 0 0 1  Stack contents: 0 $ $

$ \xrightarrow{0, \$ \rightarrow \epsilon} \epsilon $  
$ \xrightarrow{1, \$ \rightarrow \epsilon} \epsilon $  

Example NPDA

\[ \Sigma = \{0, 1\} \]
\[ \Gamma = \{0, 1, \$\} \]

• tape: 0 0 1  Stack contents: 0 0 $ $

$ \xrightarrow{0, \$ \rightarrow \epsilon} \epsilon $  
$ \xrightarrow{1, \$ \rightarrow \epsilon} \epsilon $  

Example NPDA

\[ \Sigma = \{0, 1\} \]
\[ \Gamma = \{0, 1, \$\} \]

• tape: 0 0 1  Stack contents: 0 $ $

$ \xrightarrow{0, \$ \rightarrow \epsilon} \epsilon $  
$ \xrightarrow{1, \$ \rightarrow \epsilon} \epsilon $  

Example NPDA

\[ \Sigma = \{0, 1\} \]
\[ \Gamma = \{0, 1, \$\} \]

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

$ \xrightarrow{0, \$ \rightarrow \epsilon} \epsilon $  
$ \xrightarrow{1, \$ \rightarrow \epsilon} \epsilon $  

Example NPDA

\[ \Sigma = \{0, 1\} \]
\[ \Gamma = \{0, 1, \$\} \]

• tape: 0 0 1  Stack contents: 0 $ $

$ \xrightarrow{0, \$ \rightarrow \epsilon} \epsilon $  
$ \xrightarrow{1, \$ \rightarrow \epsilon} \epsilon $  

Example NPDA

\[ \Sigma = \{0, 1\} \]
\[ \Gamma = \{0, 1, \$\} \]

• What language does this NPDA accept?

$ \xrightarrow{0, \$ \rightarrow \epsilon} \epsilon $  
$ \xrightarrow{1, \$ \rightarrow \epsilon} \epsilon $  

6
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
  – \(\Sigma\) is a finite set called the tape alphabet
  – \(\Gamma\) is a finite set called the stack alphabet
  – \(\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow P(Q \times (\Gamma \cup \{\varepsilon\}))\) 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 \(t \in \Gamma^*\)
    – \(r_m \in F\)

Example of formal definition

- \(Q = \{q_0, q_1, q_2, q_3\}\)
- \(\Sigma = \{0, 1\}\)
- \(\Gamma = \{0, 1, \$\}\)
- \(F = \{q_0, q_3\}\)
- \(\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \$)\}\)
- \(\delta(q_1, 0, 0) = \{(q_1, 0)\}\)
- \(\delta(q_1, 0, 1) = \{(q_2, \varepsilon)\}\)
- \(\delta(q_1, 1, 0) = \{(q_2, \varepsilon)\}\)
- \(\delta(q_2, \varepsilon, \$) = \{(q_3, \varepsilon)\}\)
- Other values of \(\delta(\cdot, \cdot, \cdot)\) equal \(\emptyset\)

Exercise

Design a NPDA for the language

\(\{a^ib^jc^k : i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}\)

Context-free grammars and languages

• languages recognized by a (N)FA are exactly the languages described by regular expressions, and they are called the regular languages
• languages recognized by a NPDA are exactly the languages described by context-free grammars, and they are called the context-free languages