Expressive Power of Automata

For which of the following languages can you find an automaton or regular expression:

- Sequence of open or closed parentheses of even length? E.g. (), ((,)),)()))(, ...
- as many digits before as after decimal point?
- Sequence of balanced parentheses
  ((()) ()) - balanced
  ( )) (( ) - not balanced
- Comments from // until LF
- Nested comments like /* ... /* */ ... */

Expressive Power of Automata

For which of the following languages can you find an automaton or regular expression:

- Sequence of open or closed parentheses of even length? E.g. (), ((,)),)()))(, ... yes
- as many digits before as after decimal point? No
- Sequence of balanced parentheses No

  ((()) ()) - balanced

  ( )) (( ) - not balanced
- Comments from // until LF Yes
- Nested comments like /* ... /* */ ... */ No

Automaton that Claims to Recognize $\{ a^n b^n \mid n >= 0 \}$

Make the automaton deterministic

Let the resulting DFA have K states, $|Q|=K$

Feed it a, aa, aaa, …. Let q_i be state after reading a^i

$q_0, q_1, q_2, \dots, q_K$

This sequence has length K+1 -> a state must repeat

$q_i = q_{i+p}$ $\quad\quad p > 0$

Then the automaton should accept $a^{i+p}b^{i+p}$.

But then it must also accept

$$a^i\, b^{i+p}$$

because it is in state after reading a^i as after a^{i+p}.

So it does not accept the given language.

Limitations of Regular Languages

- Every automaton can be made deterministic
- Automaton has finite memory, cannot count
- Deterministic automaton from a given state behaves always the same
- If a string is too long, deterministic automaton will repeat its behavior

Pumping Lemma

If L is a regular language, then there exists a positive integer p (the pumping length) such that every string $s \in$ L for which $|s| \geq p$, can be partitioned into three pieces, $s = x\ y\ z$, such that

- $|y| > 0$
- $|xy| \leq p$
- $\forall i \geq 0.\ xy^i z \in$ L

Let's try again: $\{\ a^n b^n \mid n >= 0\ \}$

Finite State Automata are Limited

Let us use (context-free) **grammars**!

Context Free Grammar for $a^n b^n$

S ::= ε - first rule of this grammar

S ::= a S b - second rule of this grammar.

Example of a derivation

S => aSb => a aSb b => aa aSb bb => aaabbb

Parse tree: leaves give us the result

Context-Free Grammars

$G = (A, N, S, R)$

- A - terminals (alphabet for generated words $w \in A^*$)
- N - non-terminals – symbols with (recursive) definitions
- Grammar rules in R are pairs (n,v), written

  $n ::= v$ where

  $n \in N$ is a non-terminal

  $v \in (A \cup N)^*$ - sequence of terminals and non-terminals

A derivation in G starts from the starting symbol S

- Each step replaces a non-terminal with one of its right hand sides

Example from before: $G = (\{a,b\}, \{S\}, S, \{(S,\varepsilon), (S,aSb)\})$

Parse Tree

Given a grammar $G = (A, N, S, R)$, t is a **parse tree** of G
iff t is a node-labelled tree with ordered children that satisfies:

- root is labeled by S
- leaves are labelled by elements of A
- each non-leaf node is labelled by an element of N
- for each non-leaf node labelled by n whose children left to right are labelled by $p_1 \ldots p_n$, we have a rule $(n ::= p_1 \ldots p_n) \in R$

Yield of a parse tree t is the unique word in A^* obtained by reading the leaves of t from left to right

Language of a grammar G = words of all yields of parse trees of G

$L(G) = \{\text{yield}(t) \mid \text{isParseTree}(G,t)\}$
$w \in L(G) \quad \Leftrightarrow \quad \exists t.\ w = \text{yield}(t) \wedge \text{isParseTree}(G,t)$

isParseTree - **easy** to check condition, given t

Harder: know *if for a word there **exists** a parse tree*

Grammar Derivation

A **derivation** for G is any sequence of words $p_i \in (A \cup N)^*$,whose:

- first word is S
- each subsequent word is obtained from the previous one by replacing one of its letters by right-hand side of a rule in R :
  $p_i = unv$, $(n::=q) \in R$,
  $p_{i+1} = uqv$
- Last word has only letters from A

Each parse tree of a grammar has one or more derivations, which result in expanding tree gradually from S

- Different orders of expanding non-terminals may generate the same tree
- Leftmost derivation: always expands leftmost non-terminal
  - Rightmost derivation: always expands rightmost non-terminal

Remark

We abbreviate

$S ::= p$

$S ::= q$

as

$S ::= p \mid q$

Example: Parse Tree vs Derivation

Consider this grammar G = ({a,b}, {S,P,Q}, S, R) where R is:

S ::= PQ

P ::= a | aP

Q ::= ε | aQb

Show a parse tree for aaaabb

Show at least two derivations that correspond to that tree.

Balanced Parentheses Grammar

Consider the language L consisting of precisely those words consisting of parentheses “**(**“ and “**)**” that are balanced (each parenthesis has the matching one)

- Example sequence of parentheses

((()) ()) - balanced, belongs to the language

( )) (( )   - not balanced, does not belong

Exercise: give the grammar and example derivation for the first string.

Balanced Parentheses Grammar

G_1 $\quad S ::= \varepsilon \mid S(S)S$

G_2 $\quad S ::= \varepsilon \mid (S)S$

G_3 $\quad S ::= \varepsilon \mid S(S)$

G_4 $\quad S ::= \varepsilon \mid S\ S \mid (S)$

These all define the same language, the language of balanced parentheses.