Revision Lecture

Özlem Çetinoğlu

CA215 - Languages and Computability

December 14th, 2009

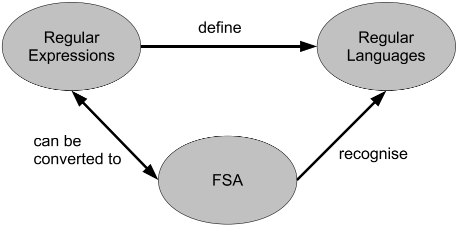
Chomsky Hiearchy

Type	Grammar	Language	Automata
3	Finite State	Regular	Finite
2	Context-Free	C-F	Pushdown
1	Context-Sensitive	C-S	Linear-Bounded
0	General Rewrite	Unrestricted	Turing Machines

Type 3 $\subset$ Type 2 $\subset$ Type 1 $\subset$ Type 0

Chomsky Hiearchy

Type	Grammar	Language	Automata
3	Finite State	Regular	Finite
2	Context-Free	C-F	Pushdown
1	Context-Sensitive	C-S	Linear-Bounded
0	General Rewrite	Unrestricted	Turing Machines



Chomsky Hiearchy

Type	Grammar	Language	Automata
3	Finite State	Regular	Finite
2	Context-Free	C-F	Pushdown
1	Context-Sensitive	C-S	Linear-Bounded
0	General Rewrite	Unrestricted	Turing Machines

$L = \{a^n b^m : n, m \geq 0\}$

Chomsky Hiearchy

Type	Grammar	Language	Automata
3	Finite State	Regular	Finite
2	Context-Free	C-F	Pushdown
1	Context-Sensitive	C-S	Linear-Bounded
0	General Rewrite	Unrestricted	Turing Machines

$L = \{a^n b^n : n \geq 0\}$

Chomsky Hiearchy

Type	Grammar	Language	Automata
3	Finite State	Regular	Finite
2	Context-Free	C-F	Pushdown
1	Context-Sensitive	C-S	Linear-Bounded
0	General Rewrite	Unrestricted	Turing Machines

$L = \{a^n b^n c^n : n \geq 0\}$

Chomsky Hiearchy

Type	Grammar	Language	Automata
3	Finite State	Regular	Finite
2	Context-Free	C-F	Pushdown
1	Context-Sensitive	C-S	Linear-Bounded
0	General Rewrite	Unrestricted	Turing Machines

Grammars

Formally, a grammar G is a quadruple (T, N, S, P), where:

- T is the set of *terminal* symbols,
- N is the set of *nonterminals*,
- S is the *start symbol*, $S \in N$,
- P is a finite set of *productions* of the form
  FSG: $A \longrightarrow a, A \rightarrow aB$ (or $A \rightarrow Ba$ but not both) , or $A \rightarrow \epsilon$ if A is not on the RHS of any rule.
  CFG: A $\rightarrow \alpha$, where A $\in$ N and $\alpha \in (T \cup N)^*$ Each production must have a single non-terminal on its left hand side.
  CSG: $\alpha \rightarrow \beta$, where $|\alpha| \leq |\beta|$ and $\alpha, \beta \in (T \cup N)^*$
  OR $A \rightarrow \epsilon$ if $A \in N$ is not on the RHS of a rule.
  general rewrite grammar: $\alpha \rightarrow \beta$, where $\alpha, \beta \in (T \cup N)^*$ and α contains at least one nonterminal

Machines

PDA: FSA with a stack

Turing Machine: LBA with an infinite tape

FSA Definition

A finite state automaton is a 5-tuple $M = (Q, \Sigma, q_0, F, \delta)$, where:

- Q is a finite set of *states*,
- Σ is an input alphabet,
- $q_0 \in Q$ is the *initial state*,
- $F \subseteq Q$ is the set of *final states*,
- δ is the *transition function*, $Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \wp(Q)$.

PDA Definition

A pushdown automaton is a 6-tuple $M = (Q, \Sigma, \Gamma, q_0, F, \delta)$, where:

Q is a finite set of *states*,

Σ is an alphabet (*input symbols*),

Γ is an alphabet (*stack symbols*),

$q_0 \in Q$ is the *initial state*,

$F \subseteq Q$ is the set of *final states*,

δ is the *transition function*, $Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \to Q \times (\Gamma \cup \{\epsilon\})$.

If $((q, a, \alpha), (q', \beta)) \in \delta$, then when in state q with α at the top of the stack, M may

- read a from input
- replace α with β on the top of the stack
- enter state q'

LBA Definition

A linear bounded automaton is a 5-tuple $M = (Q, \Sigma, \Gamma, q_0, \delta)$, where:

- Q is a finite set of *states*,
- Σ is an *input alphabet*,
- Γ is a *tape alphabet*,
- $q_0 \in Q$ is the *initial state*,
- δ is the *transition function* $Q \times (\Gamma \cup \{<, >\}) \to Q \times (\Gamma \cup \{<, >\}) \times \mathcal{A}$.

If $((q, a), (q', b, action)) \in \delta$, then when in state q with a at the current read position on the tape, M may replace a with b on the tape, perform the specified *action*, and enter state q'.

M accepts $w \in \Sigma^*$ iff it starts with configuration $(q_0, \underline{<} w >)$ and the action Y is taken.

TM Definition

A Turing machine is a 5-tuple $M = (Q, \Sigma, \Gamma, q_0, \delta)$, where:

- Q is a finite set of *states*,
- Σ is an *input alphabet*,
- Γ is a *tape alphabet*,
- $q_0 \in Q$ is the *initial state*,
- δ is the *transition function* $Q \times (\Gamma \cup \{<\}) \rightarrow Q \times (\Gamma \cup \{<\}) \times \mathcal{A}$.

If $((q, a), (q', b, action)) \in \delta$, then when in state q with a at the current read position on the tape, M may replace a with b on the tape, perform the specified *action*, and enter state q'.

The symbol "$\#$" is used to denote a blank tape square.

M accepts $w \in \Sigma^*$ iff it starts with configuration $(q_0, \underline{w}\#)$ and the action Y is taken.

Question

$M_1 = (Q, \Sigma, q_0, \delta, F)$ where:

- $Q = \{q_0, q_1\}$
- $\Sigma = \{a, b, c\}$
- $q_0 = q_0$
- $F = \{q_1\}$
- $\delta = \begin{array}{l} ((q_0, a), q_0) \\ ((q_0, b), q_0) \\ ((q_0, a), q_1) \\ ((q_0, c), q_1) \end{array}$

- Describe in English the language accepted by this NFA.
- Define the regular expression of the language accepted by this NFA.
- Convert this NFA into the equivalent DFA.

Other Questions

Last year's exam
Downloadable from
$http://www.computing.dcu.ie/\ away/CA215/Exams/0809sem1.pdf$