Computational Linguistics II: Parsing

Formal Languages: Context Free Languages II

Frank Richter & Jan-Philipp Söhn

fr@sfs.uni-tuebingen.de, jp.soehn@uni-tuebingen.de

November 8th, 2006

Once Again: The Big Picture

hierarchy	grammar	machine	other
type 3	reg. grammar	DFA	reg. expressions
		NFA	
det. cf.	LR(k) grammar	DPDA	
type 2	CFG	PDA	
type 1	CSG	LBA	
type 0	unrestricted grammar	Turing machine	

DFA: Deterministic finite state automaton
(D)PDA: (Deterministic) Pushdown automaton
CFG: Context-free grammar
CSG: Context-sensitive grammar

LBA: Linear bounded automaton

Defining the Pushdown-Automaton

Definition 1 (NPDA) A nondeterministic pushdown-automaton is a septuple $(\Sigma, Q, \Gamma, q_0, Z, F, \delta)$ where

Σ is a finite set called *the input alphabet*,

Q is a finite set of *states*,

Γ is a finite set called *the stack alphabet*,

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

$Z \in \Gamma$ is the *start symbol* on the stack,

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

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

Formal Basics I

Definition (State)

Given an NPDA $M = (\Sigma, Q, \Gamma, q_0, Z, F, \delta)$,
each $k \in Z \times \Sigma^* \times \Gamma^*$ is a state of M.

Formal Basics II

Definition (directly derives)

Given an NPDA $M = (\Sigma, Q, \Gamma, q_0, Z, F, \delta)$,
a state $k_1 = (z, a_1 \dots a_n, A_1 \dots A_m) \in Z \times \Sigma^* \times \Gamma^*$ directly derives state k_2 iff

1. $k_2 = (z', a_2 \dots a_n, B_1 \dots B_i A_2 \dots A_m)$ and $\delta(z, a_1, A_1) \ni (z', B_1 \dots B_i)$, or
2. $k_2 = (z', a_1 a_2 \dots a_n, B_1 \dots B_i A_2 \dots A_m)$ and $\delta(z, \epsilon, A_1) \ni (z', B_1 \dots B_i)$.

We write $k_1 \vdash k_2$.

Definition (derives)

Given an NPDA $M = (\Sigma, Q, \Gamma, q_0, Z, F, \delta)$,
a state k_1 derives state k_n iff there is a sequence $k_1 \vdash k_2 \dots k_n$.
We write $k_1 \vdash^* k_n$.
($\vdash^*$ is the reflexive transitive closure of $\vdash$.)

Formal Basics III

Definition (Acceptance)
Given an NPDA $M = (\Sigma, Q, \Gamma, q_0, Z, F, \delta)$ and a string $x \in \Sigma^*$,
M accepts x iff
there is a $q \in F$ such that $(q_0, x, Z) \vdash^* (q, \epsilon, \epsilon)$.

Definition (Language accepted by M)
Given an NPDA $M = (\Sigma, Q, \Gamma, q_0, Z, F, \delta)$,
the language L(M) accepted by M is the set of strings accepted by M,
L(M) $= \{x \in \Sigma^* | (q_0, x, Z) \vdash^* (q, \epsilon, \epsilon)$ for some $q \in F\}$.

Example of a CFG

The grammar we saw last time:

$$S \longrightarrow A\ B$$

$$A \longrightarrow aAb$$

$$A \longrightarrow ab$$

$$B \longrightarrow cB$$

$$B \longrightarrow c$$

Bad example: left recursion

$$B \longrightarrow Bc$$

$$B \longrightarrow c$$

Example of a CFG II — Bracketing

Proc → WhProc | IfProc

WhProc → while Cond do Proc

IfProc → if Cond then Proc

Cond → ...

S → [NP VP]

VP → [vb (NP)]

NP → [det AP n]

AP → [adj | adj AP]

[[det adj n] [vb [det adj n]]]

Closure Properties I

Union

- Type 3 languages are closed $(\alpha|\beta)$
- Det. cf. languages are not closed $(L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}})$
- Type 2 languages are closed

Concatenation

- Type 3 languages are closed $(\alpha\beta)$
- Det. cf. languages are not closed $\{a^n\$b^n + b^m\$c^m\}$
- Type 2 languages are closed

Complementation

- Type 3 languages are closed (FSA: final states $\leftrightarrow$ non-final states)
- Det. cf. languages are closed
- Type 2 languages are not closed

Closure Properties II

Kleene star

- Type 3 languages are closed (α^*)
- Det. cf. languages are not closed $\{a^n\$a^n + a^n\$a^n\}$
- Type 2 languages are closed

Intersection

- Type 3 languages are closed ($L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$)
- Det. cf. languages are not closed
$$\{a^nb^nc^m\} \cap \{a^nb^mc^m\} = \{a^nb^nc^n\}$$
- Type 2 languages are not closed
- The intersection of a det. cf. language with a regular language is also det. cf.

Decision Properties I

Word problem

- Type 3 languages: decidable (FSA: final state reached?)
- Type 2 languages: decidable (CYK algorithm)

Emptiness problem

- Type 3 languages: decidable (FSA: path from initial to final state?)
- Type 2 languages: decidable (marking of symbols in grammar)

Finiteness problem

- Type 3 languages: decidable (FSA: path from initial state to cycle?)
- Type 2 languages: decidable (cycles in grammar-graph)

Decision Properties II

Equivalence problem

- Type 3 languages: decidable (compare minimized DFAs of $L(G_1)$ and $L(G_2)$)
- Det. cf. languages: decidable (proved 1997)
- Type 2 languages: not decidable

Intersection problem

- Type 3 languages: decidable (Emptiness of $L(G)=L(G_1) \cap L(G_2)$?)
- Det. cf. languages: not decidable (not closed unter intersection)
- Type 2 languages: not decidable (not closed unter intersection)

Again: Pumping Lemma for Regular Languages

uvw theorem:

For each regular language L there is an integer n such that for each $x \in L$ with $|x| \geq n$ there are u, v, w with $x = uvw$ such that

1. $|v| \geq 1$,
2. $|uv| \leq n$,
3. for all $i \in \mathbb{N}_0$: $uv^iw \in L$.

Pumping Lemma for Context Free Languages

uvxyz theorem:

For each context free language L there is an integer n such that for each $a \in L$ with $|a| \geq n$ there are u, v, x, y, z with $a = uvxyz$ such that

1. $|vy| \geq 1$,
2. $|vxy| \leq n$,
3. for all $i \in \mathbb{N}_0$: $uv^ixy^iz \in L$.