Lec 10. Ambiguity and Pushdown Automata

Eunjung Kim

Ambiguity in grammars and languages

In the grammar G_{ari}

1. $E \rightarrow I \quad | \quad E + E \quad | \quad E \times E \quad | \quad (E)$
2. $I \rightarrow a \quad | \quad b \quad | \quad Ia \quad | \quad Ib \quad | \quad I0 \quad | \quad I1$

How many parse trees that yield the string "$E + E \times E$"?

How many parse trees that yield the string "$a + b$"?

Ambiguity in grammars and languages

Ambiguous grammar

- A grammar is ambiguous if there is a string $w \in \Sigma^*$ such that there are (at least two) parse trees, in each of which the root is labelled by the start variable S and w is the yield.
- Equivalently, a grammar is ambiguous if a string has two leftmost derivations.
- A grammar is unambiguous if every string has at most one parse tree in the grammar.

Inherent ambiguity

Inherently Ambiguous CFL

- A context-free language L is inherently ambiguous if any grammar G which generates L (i.e. $L(G) = L$) is ambiguous.
- A CFL L is unambiguous if there is an unambiguous grammar G which generates L.

Eliminating ambiguity

- There is no algorithm which decides whether a given CFG is ambiguous or not ("undecidable problem").
- There are inherently ambiguous languages: e.g. $\{a^n b^n c^m d^m : n, m \geq 1\} \cup \{a^n b^m c^m d^n : n, m \geq 1\}$
- Showing if a language is inherently ambiguous or unambiguous is not easy (in terms of proof...)
- BUT, many CFL's we care are unambiguous, and there are techniques to modify the grammar to eliminate ambiguity.

Eliminating ambiguity: example 1

The grammar G_{ari} is ambiguous: e.g. $a + a \times a$ and $a + a + a$

1. $E \to I \mid E + E \mid E \times E \mid (E)$
2. $I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Goal: we want the grammar to

- respect the priority of operators, and
- generate a sequence of identical operations in a unique way, e.g. grouped from left to right.

Eliminating ambiguity: example 1

The grammar G_{ari} is ambiguous: e.g. $a + a \times a$ and $a + a + a$

1. $E \to I \quad | \quad E + E \quad | \quad E \times E \quad | \quad (E)$
2. $I \to a \quad | \quad b \quad | \quad Ia \quad | \quad Ib \quad | \quad I0 \quad | \quad I1$

Arithmetic expression with $+, \times$ are written like

$$(a0 + b1) \times a0 + b0 \times b \times b_0 + b11 \times (a1 + b0) = term + term + term$$

where each term is factored into

$$(a0 + b1) - a0,\ b_0\text{-}b\text{-}b_0 \text{ and } b11 - (a11 + b0).$$

Eliminating ambiguity: example 1

We introduce intermediary variables which represent the following.

- "Identifier": the existing variable. *I* already represents them.
- "Factor": the operands of $\times$. Identifiers or an expression surrounded by () in the expressions of G_{ari}.
- "Term": those separated by $+$ in an arithmetic expression. Either an identifier or have multiple factors.
- "Expression": any expression generated by G_{ari}. Either a term or "+" of ≥ 2 terms.

Eliminating ambiguity: example 1

We introduce intermediary variables which represent the following.

- "Identifier": the existing variable. I already represents them.
- "Factor": the operands of $\times$. Identifiers or an expression surrounded by () in the expressions of G_{ari}.
- "Term": those separated by $+$ in an arithmetic expression. Either an identifier or have multiple factors.
- "Expression": any expression generated by G_{ari}. Either a term or "+" of ≥ 2 terms.

Let us construct a CFG with the additional variables as above (Start E).

- $I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
- $F \to I \mid (E)$
- $T \to F \mid T \times F$
- $E \to T \mid E + T$

Eliminating ambiguity: example 1

New CFG generating $L(G_{ari})$ (Start E).

- $I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
- $F \rightarrow I \mid (E)$
- $T \rightarrow F \mid T \times F$
- $E \rightarrow T \mid E + T$

Parse tree for $a + a \times a$.

Eliminating ambiguity: example 1

New CFG generating $L(G_{ari})$ (Start E).

- $I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
- $F \rightarrow I \mid (E)$
- $T \rightarrow F \mid T \times F$
- $E \rightarrow T \mid E + T$

Parse tree for $a + a \times a$.

The new CFG is an unambiguous grammar generating the same language.

Eliminating ambiguity: example 2

CFG generating a well-formed parenthesis.

$$S \rightarrow SS \quad | \quad (S) \quad | \quad \epsilon$$

How many parse trees for ()()()?

Eliminating ambiguity: example 2

CFG generating a well-formed parenthesis.

$$S \rightarrow SS \quad | \quad (S) \quad | \quad \epsilon$$

How many parse trees for ()()()?

Ambiguity of the grammar arises from that a concatenation of well-formed parenthesis can be expressed by multiple parse trees.

We use the same principle for eliminating ambiguity as for the arithmetic expression.

Pushdown automata

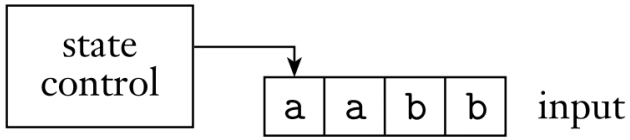

Figure 2.11, Sipser 2012

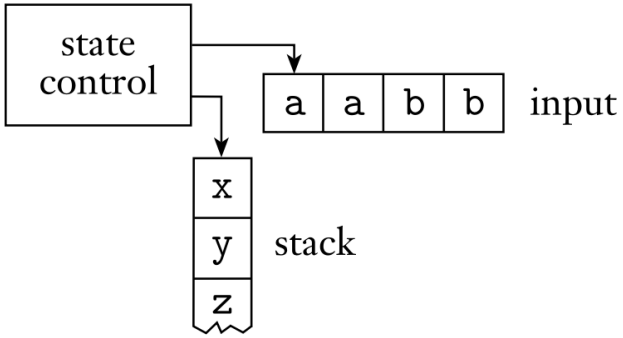

Figure 2.12, Sipser 2012

Pushdown automata

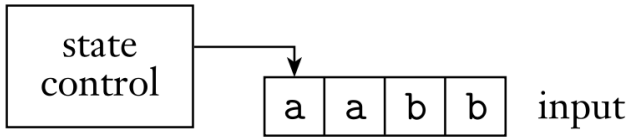

Figure 2.11, Sipser 2012

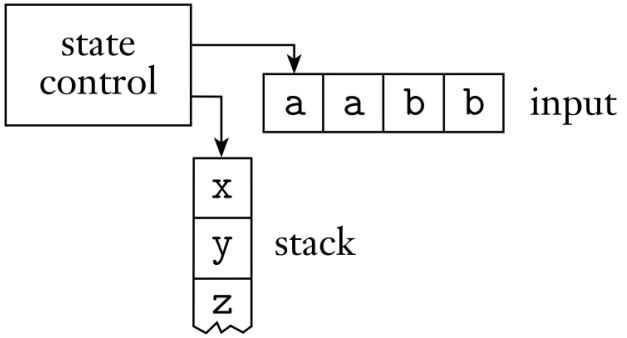

Figure 2.12, Sipser 2012

PDA: informal description

A pushdown automata has three components: an input tape, a state (control), and a stack. At each step, PDA does the following

- current state: q.
- read from input: the first symbol a of the input tape; possibly ϵ.
- pops off stack: the first symbol x at (the top of) the stack; possibly ϵ.
- transition: depending on (q, a, x), PDA
  1. updates the current state q to a new state q'
  2. pushes a symbol y to the stack; possibly ϵ.

PDA FOR $L = \{w \cdot w^R : w \in \{0,1\}^*\}$

(INFORMAL) PDA RECOGNIZING A PALINDROME

- It has three states: q_{start}, q_{push}, $q_{pop\&match}$, q_{accept}.
- Starting at q_{start}, it pushes \$ to stack and updates to q_{push}.
- Stays in q_{push} while: it reads the symbol b from input and pushes b to stack.
- "Guess" the middle of the word; implemented as an update from q_{push} to $q_{pop\&match}$
- Stays in $q_{pop\&match}$ while: it reads the symbol b from input, pops the symbol b; they match.
- Moves $q_{pop\&match}$ to q_{accept} if: there is nothing to read in the input when \$ is popped.
- In all other cases (e.g. "mismatch" between the input content and stack symbol): "dies"

Formal definition of PDA

A pushdown automata is

a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where all entries except for q_0 is a finite set:

- Q is the set of states.
- Σ is the input alphabet.
- Γ is the stack alphabet.
- $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ is the transition function.
- $q_0 \in Q$ is the start state.
- $F \subseteq Q$ is the set of accept states.

Language recognized by a PDA

PDA accepting a string

A pushdown automata $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts an input string $w \in \Sigma^*$ if

I. w can be written as $w_1 w_2 \cdots w_n$, where $w_i \in \Sigma_\epsilon$,

II. there is a sequence $r_0, r_1, \ldots, r_n$ of states, and

III. a sequence of strings $s_0, s_1, \ldots, s_n$ of strings over Γ^*,

such that the following holds:

1. $r_0 = q_0$ and $s_0 = \epsilon$,
2. $s_i = xt$, $s_{i+1} = yt$ and $(r_{i+1}, y) \in \delta(r_i, w_{i+1}, x)$ hold for some $x, y \in \Gamma_\epsilon$ and $t \in \Gamma^*$
3. $r_n \in F$.

Instantaneous description (a.k.a configuration)

Configuration of PDA

For a pushdown automata $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$, a configuration of P (also called an instantaneous description (ID)) is a triple $(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$. Essentially

- q is the current state.
- w is the remaining input string.
- γ is the string in the stack,read from top to bottom.

If $(q', y) \in \delta(q, a, x)$, then we write

$$(q, aw, x\beta) \vdash (q', w, y\beta).$$

This means that one can reach the configuration $(q', w, y\beta)$ from $(q, aw, x\beta)$ in a single step (transition). The notation $\vdash^*$ is used when one is reach from the other in $n \geq 0$ steps.

Language recognized by a PDA

Language recognized by PDA

For a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$, the language recognized by PDA P is defined as

- the set of all strings in Σ^* accepted by P, or equivalently
- the set of all strings $w \in \Sigma^*$ such that $(q_0, w, \epsilon) \vdash^* (q, \epsilon, \gamma)$ for some $q \in F$ and $\gamma \in \Gamma^*$.

We write as $L(P)$ the language recognized by PDA P.

PDA FOR $L = \{0^n 1^n : n \geq 0\}$

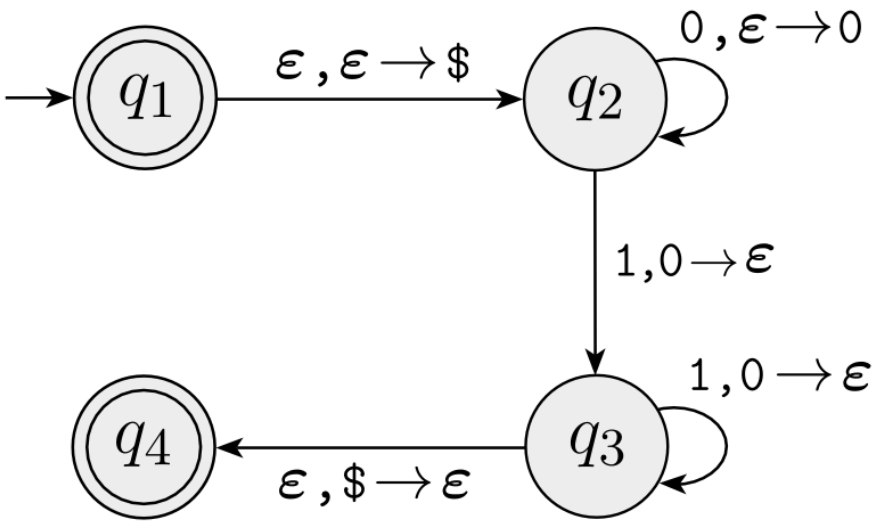

Figure 2.15, Sipser 2012

PDA FOR $L = \{0^n 1^n : n \geq 0\}$

Formal description of PDA recognizing L: it is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_1, F)$ as follows.

- $Q = \{q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, \$\}$
- the transition function δ is given as the transition table below.
- Start state is q_1
- $F = \{q_1, q_4\}$

Input:	0			1			ϵ		
Stack:	0	\$	ϵ	0	\$	ϵ	0	\$	ϵ
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \epsilon)\}$					
q_3				$\{(q_3, \epsilon)\}$				$\{(q_4, \epsilon)\}$	
q_4									

From example 2.14, Sipser 2012

PDA FOR $L = \{w \cdot w^R : w \in \{0,1\}^*\}$

PDA FOR $L = \{w \in 0^n 1^m : n \geq m\}$

PDA FOR $L = \{w \in \{a, b\}^* : |w|_a = |w|_b\}$

PDA FOR

$$L = \{w \in a^i b^j c^k : i = j \textbf{ OR } i = k\}$$