Determinism and Parsing

CENG 280

Course outline

- Preliminaries: Alphabets and languages
- Regular languages DFA NFA =
- Context-free languages
  - Context-free grammars
  - Parse trees
  - Push-down automaton
  - Push-down automaton - context-free languages
  - Languages that are and that are not context-free, Pumping lemma
  - Deterministic CFLs
- Turing-machines

Determinism and Parsing

PL,

string $w \to \boxed{\text{parser}}$

G

$w \in L(G)$?

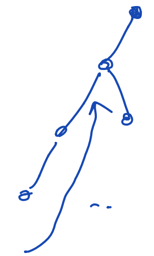
if yes constructs a parse tree for w

3.6 CYK

d_1.

d_2

$(q, w', z) \vdash \ldots \vdash \ldots$



- Parsing for programming languages
- Parsing with PDAs
- Deterministic CFLs with deterministic PDAs
- Heuristic rules for converting grammar to get a deterministic PDA

Deterministic Context-free Languages

A pushdown automaton M is **deterministic** if for each configuration there is at most one configuration that can succeed it in a computation by M.

Deterministic Context-free Languages

- Two strings are said to be **consistent** if one of them is the prefix of the other.
  - $e - aa$, $a - aa$, $aa - a$, $e - e$, $a - e$, $b - ba$, ~~$a - ba$~~
- Two transitions $((q, a, \beta), (p, \gamma))$ and $((q, a', \beta'), (p', \gamma'))$ are said to be **compatible** if both a and a' are consistent, and β and β' are also consistent.
  $\alpha - \alpha$, $e - \alpha$, $e - e$
- If a PDA M has compatible transitions, then there can be situations where both transitions applicable.
  - $((q, a, e), (p, \gamma)) - ((q, a, a), (p', \gamma'))$ $(q, ab, abb) \vdash$
  - $((q, e, ab), (p, \gamma)) - ((q, e, a), (p', \gamma'))$ $q,$
- A PDA is **deterministic** if it does not have compatible transitions, i.e., no non-deterministic choices.

Deterministic-Nondeterministic PDA Examples

Example

$L = \{wcw^R\}$, $w \in \{a, b\}^*$

$S \to c \mid aSa \mid bSb$ 1-2, 2-3, 1-3

1.$((s, a, e), (s, a))$ not
2.$((s, b, e), (s, b))$ compatible
3.$((s, c, e), (f, e))$
4.$((f, a, a), (f, e))$ not
5.$((f, b, b), (f, e))$ comp.

deterministic PDA

Deterministic-Nondeterministic PDA Examples

Example

$L = \{wcw^R\}$,
$S \to c \mid aSa \mid bSb$

1. $((s, a, e), (s, a))$
2. $((s, b, e), (s, b))$
3. $((s, c, e), (f, e))$
4. $((f, a, a), (f, e))$
5. $((f, b, b), (f, e))$

Example

$L = \{\underline{ww}^R\}$, 1–3 are compatible
$S \to aSa \mid bSb \mid e$ 2–3 are compatible

1. $((\underline{s}, \underline{a}, \underline{e}), (s, a))$
2. $((\underline{s}, \underline{b}, \underline{e}), (s, b))$
3. $((\underline{s}, \underline{e}, \underline{e}), (f, e))$
4. $((\underline{f}, \underline{a}, a), (f, e))$ not
5. $((\underline{f}, b, b), (f, e))$ compatible

not deterministic

Deterministic Context-free Languages

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

$L\$ = \{ \$, ab\$, aabb\$, \ldots \}$

Definition

A language $L \subseteq \Sigma^*$ is a deterministic context-free language if $L\$ = L(M)$ for some deterministic pushdown automaton M (M senses the end of the input).

- Why \$ is necessary? Consider $L = \{a^i \mid i \geq 0\} \cup \{a^n b^n \mid n \geq 0\}$
- Every deterministic context-free language is a context-free language. Why? $L\$ = L(M)$ M' $L = L(M')$
- Is every CFL deterministic?

NO

Deterministic Context-free Languages

Yes L is CF, write G, s.t. L(G) = L

Example

Consider $L = \{a^n b^m c^p \mid m, n, p \geq 0, m \neq n \text{ or } n \neq p\}$. Is L context-free, if yes, is it deterministic? L$
Consider the complement of L. It's complement is not CFL (in a few minutes). Deterministic CFL's are closed under complementation.

$\rightsquigarrow$ L is D FCL, $\Sigma^* - L$ is also D-CFL

Example

Consider $L = \{a^n b^m c^p \mid m, n, p \geq 0, m \neq n \text{ or } n \neq p\}$. Is L context-free, if yes, is it deterministic?
Consider the complement of L. It's complement is not CFL (in a few minutes). Deterministic CFL's are closed under complementation.

Theorem

The class of deterministic context-free languages is closed under complementation. if L is a D-CFL, then $\bar{L} = \Sigma^* - L$ is D-CFL

Proof: Read the book. M, $L\$ = L(M)$, construct M''

M'' is deterministic

$L(M'') = \bar{L}$

Theorem

The class of deterministic context-free languages is closed under complementation. if $w \in L(M')$, $w \notin L(M'')$
if $w \notin L(M')$, $w \in L(M'')$
M' is also deterministic

- First convert M to a simple $M' = (K, \Sigma, \Gamma, \Delta, S, F)$ (it only depends on input symbol, top of the stack, does not change the deterministic property). $((q, \alpha, \gamma), (p, \beta))$ $|\gamma| \leq 1$ $|\beta| \leq 2$ M''
- Consider the acceptance condition, simple switch of final/non-final states are not sufficient. $F'' = K \setminus F \cup \{f\}$
- R If the computation ends in F and the stack is empty, then REJECT (no issue, $K \setminus F$ is the set of final states). $w \in L(M')$
- A If the computation ends in $K \setminus F$, then accept (need to empty the stack) \$
- A If the computation ends in F and the stack is not empty, then empty the stack and ACCEPT. $\{f\}$, empty the stack
- A If the computation ends in a dead-end, no transitions-stack operations is possible. Read the rest, empty the stack and accept the word. $L(M'') = \overline{L}\$$

Deterministic Context-free Languages

Theorem

The class of deterministic context-free languages is closed under complementation.

Corollary

The class of deterministic context-free languages is a proper subset of the class of context-free languages.

Proof?

Example

Consider $L = \{a^n b^m c^p \mid m, n, p \geq 0, m \neq n \text{ or } n \neq p\}$.

$\bar{L} = \{a^n b^n c^n \mid n \geq 0\} \cup$ ..... the order changes

$\bar{L}$ is not CF $\quad \{b,c\}^* a \{a,b,c\}^* \cup \{a,c\}^* b \{a,b,c\}^*$

$\underbrace{\bar{L}}_{CF} \cap \underbrace{(a^* b^* c^*)}_{R} = \underbrace{\{a^n b^n c^n \mid n \geq 0\}}_{CF}$ not

$\bar{L}$ not CF

L is not Det-CF.

Deterministic Context-free Languages

Theorem

The class of deterministic context-free languages is closed under complementation.

Corollary

The class of deterministic context-free languages is a proper subset of the class of context-free languages.

Proof?

Example

Consider $L = \{a^n b^m c^p \mid m, n, p \geq 0, m \neq n \text{ or } n \neq p\}$.
$\bar{L} = \{a^n b^n c^n \mid n \geq 0\} \cup$ ..... the order changes
$\bar{L} \cap a^\star b^\star c^\star = \{a^n b^n c^n \mid n \geq 0\}$

Nondeterminism is more powerful than determinism in the context of pushdown automata.

Parsing

$w \in L(G)$

L_1
L_2 } deterministic,

$L_1 \cup L_2$ is not det.

- Deterministic CFLs are not closed under union. Proof?
- Only deterministic CFL can be recognized by a deterministic PDA.
- Given a grammar G, can we construct a deterministic PDA M with $L(G) = L(M)$?
- This question is undecidable. There is no algorithm to answer the question for an arbitrary grammar.
- Deterministic context-free languages are never inherently ambiguous. Proof?
- There are some heuristic approaches to eliminate grammar rules that result in compatible transitions, so that the resulting automaton will be deterministic (some examples in the book).

G PDA M

Top-Down Parsing

Definition (Top-down parser)

A deterministic pushdown automaton M is considered to be a topdown parser when its stack operations along a computation reconstructs the parse tree in a top-down left-to right fashion.

Top-Down Parsing

Definition (Top-down parser)

A deterministic pushdown automaton M is considered to be a topdown parser when its stack operations along a computation reconstructs the parse tree in a top-down left-to right fashion.

Example

$L = \{a^n b^n \mid n \geq 0\}$, $S \rightarrow e \mid aSb$

G M $L(G) = L(M)$

1.$((s, e, e), (q, S))$

2.$((q, e, S), (q, aSb))$

3.$((q, e, S), (q, e))$

4.$((q, a, a), (q, e))$

5.$((q, b, b), (q, e))$

Top-Down Parsing

Definition (Top-down parser)

A deterministic pushdown automaton M is considered to be a topdown parser when its stack operations along a computation reconstructs the parse tree in a top-down left-to right fashion.

Example

$L = \{a^n b^n \mid n \geq 0\}$, $S \rightarrow e \mid aSb$

1.$((s, e, e), (q, S))$
2.$((q, e, S), (q, aSb))$
3.$((q, e, S), (q, e))$
4.$((q, a, a), (q, e))$
5.$((q, b, b), (q, e))$

1.$((s, e, e), (q, S))$
2.$((q, a, e), (q_a, e))$
3.$((q_a, e, a), (q, e))$
4.$((q_a, e, S), (q_a, aSb))$
5.$((q, b, e), (q_b, e))$
6.$((q_b, e, b), (q, e))$
7.$((q_b, e, S), (q_b, e))$
8.$((q, \$, e), (q_f, e))$

Bottom up parsing

Definition

Given $G = (V, \Sigma, R, S)$, the bottom up push-down automaton $M = (K, \Sigma, \Gamma, \Delta, p, F)$ is defined as follows: $K = \{p, q\}$, $\Gamma = V$, $F = \{q\}$, and Δ:

1. $((p, a, e), (p, a))$ for each $a \in \Sigma$
2. $((p, e, \alpha^R), (p, A))$ for each rule $A \rightarrow \alpha$ in R
3. $((p, e, S), (q, e))$

Bottom up parsing

Definition

Given $G = (V, \Sigma, R, S)$, the bottom up push-down automaton $M = (K, \Sigma, \Gamma, \Delta, p, F)$ is defined as follows: $K = \{p, q\}$, $\Gamma = V$, $F = \{q\}$, and Δ:

1. $((p, \underline{a, e}), (p, a))$ for each $a \in \Sigma$
2. $((p, e, \underline{\alpha^R}), (p, \underline{A}))$ for each rule $A \rightarrow \alpha$ in R
3. $((p, e, S), (q, e))$

Example

Construct a bottom up parser for $G = (V, \Sigma, R, S)$, with rules

aabb $\rightarrow$ $S \rightarrow aSa \mid bSb \mid e$