Determinism and Parsing

CENG 280

Course outline

- Preliminaries: Alphabets and languages
- Regular languages
- 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

- 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.
  - $e - aa$, $a - aa$, $aa - a$
- 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.
- If a PDA M has compatible transitions, then there can be situations where both transitions applicable.
  - $((q, a, e), (p, \gamma)) - ((q, a, a), (p', \gamma'))$
  - $((q, e, ab), (p, \gamma)) - ((q, e, a), (p', \gamma'))$
- A PDA is **deterministic** if it does not have compatible transitions, i.e., no non-deterministic choices.

Deterministic-Nondeterministic PDA Examples

Example

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

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

Deterministic-Nondeterministic PDA Examples

Example

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

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

Example

$L = \{ww^R\}$,
$S \rightarrow aSa \mid bSb \mid e$

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

Deterministic Context-free Languages

Definition

A language $L \subseteq \Sigma^\star$ 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?
- Is every CFL deterministic?

Deterministic Context-free Languages

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.

Deterministic Context-free Languages

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.

Proof: Read the book.

Theorem

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

- 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).
- Consider the acceptance condition, simple switch of final/non-final states are not sufficient.
- 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).
- 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.
- 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.

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

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

- 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).

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

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

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

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

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:

$$1. ((p, a, e), (p, a)) \text{ for each } a \in \Sigma$$

$$2. ((p, e, \alpha^R), (p, A)) \text{ for each rule } A \to \alpha \text{ in } R$$

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

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

Example

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

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