INT201 Decision, Computation and Language

Lecture 7 – Context-Free Languages (2)

Dr Yushi Li and Dr Chunchuan Lyu

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Assistant Professor-Dr Chunchuan Lyu

- Graduated from The University of Edinburgh and XJTLU

    Studied computational semantics but moved to unsupervised reinforcement learning (what an agent should do if no moral gold standard is given?)

- Office hour: 14:00-16:00 Thursday at SD543 (or by appointment)
- Contact: chunchuan.lyu@xjtlu.edu.cn

Overall Study Tips

- Theory of Computation in 12 Hours by Easy Theory Youtuber

    Really clear explaination
- Theory of Computation 2020 by Michael Sipser MIT OCW

    We are following closely
- The Nature of Computation

    Good complementary book

Please come to office hour, if you are having difficulty or question about anything.

Noam Chomsky 1928-now

An American professor, father of modern linguistics

- Transformational Analysis (1955)
- Syntactic Structures (1957)
- Minimalist program (1995)

What is language?

Why does it have the properties it has?

Formal Basis of a Language Universal (2021 Miloš Stanojević, Mark Steedman)

Noam Chomsky 1928-now

A public intellectual

- Manufacturing Consent (1988 with Edward S. Herman)
- On Palestine (2015 with Ilan Pappé)
- Consequences of Capitalism (2021 with Marv Waterstone)

"one of the most notable contemporary champions of the people"

"pathological hatred of his own country"

Recap

- Regular languages are context-free
- Every context-free grammar has a Chomsky Normal Form

Today

- Closure property of context-free grammar
- Syntactic parsing (*optional)
- Pushdown Automata

Closure Properties of Context Free Language

上下文无关语言 并集

Theorem: If $L_1$and L_2 are context-free languages, their union $L_1 \cup L_2$ is also context free.

Example:

$$L_1 = \{a^n b^n c^m | m \geq 0, n \geq 0\}$$

$$L_2 = \{a^n b^m c^m | m \geq 0, n \geq 0\}$$

$$L_3 = L_1 \cup L_2 = \{a^i b^j c^k | i \geq 0, j \geq 0, k \geq 0, i = j \text{ } or \text{ } j = k\}$$

Proof idea:
For $L_1$and L_2 , there exists corresponding context free grammars $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$. Let $G_3 = (V_1 \cup V_2, \Sigma_1 \cup \Sigma_2, , R_1 \cup R_2 \ \cup \{S \to S_1 | S_2\}, S)$, clearly $\mathrm{L}(G_3) = L_1 \cup L_2$.

Closure Properties of Context Free Language

Theorem: If $L_1$and L_2 are context-free languages, their ~~union~~ $L_1 \circ L_2$ is also context free.

Concatenate

Example:

$$L_1 = \{a^n | n \geq 0\}$$

$$L_2 = \{b^n | n \geq 0\}$$

$$L_3 = L_1 \circ L_2 = \{a^i b^j | i \geq 0, j \geq 0\}$$

Proof idea:
For $L_1$and L_2 , there exists corresponding context free grammars $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$. Let $G_3 = (V_1 \cup V_2, \Sigma_1 \cup \Sigma_2, , R_1 \cup R_2 \ \cup \{S \to S_1 S_2\}, S)$, clearly $\mathrm{L}(G_3) = L_1 \circ L_2$.

Closure Properties of Context Free Language

Theorem: If L_1 is context-free languages, their Kleene closure L_1^* is also context free.

Example:

$$L_1 = \{a^n b^n | n \geq 0\}$$

$$\cancel{L_2 = L_1^* = \{(a^n b^n)^k \, | n \geq 0, k \geq 0\}}$$

$$L_2 = L_1^* = \{\circ_{i \in \{0,\dots,k-1\}} (a^{n_i} b^{n_i}) \mid n_i \geq 0, k \geq 0\}$$

Proof idea:
For L_1 , there exists corresponding context free grammars $G_1 = (V_1, \Sigma_1, R_1, S_1)$. Let $G_2 = (V_1, \Sigma_1, R_1 \cup \{S \rightarrow S_1 S | \epsilon\}, S)$, clearly L(G_2) =L_1^*.

Parsing Natural Language with Context-Free Grammar

Given CFG G = (V, Σ, R, S)
Variables V = {S, NP,VP,Det,Nominal,Noun,PP,Preposition,Verb}
Terminals Σ={The, spy, saw, cop, with, a , telescope}

Rules:

Grammar
- S → NP VP
- NP → Det Nominal 名义上的
- Nominal → Noun | Nominal PP
- VP → VP PP | Verb NP
- PP → Preposition NP

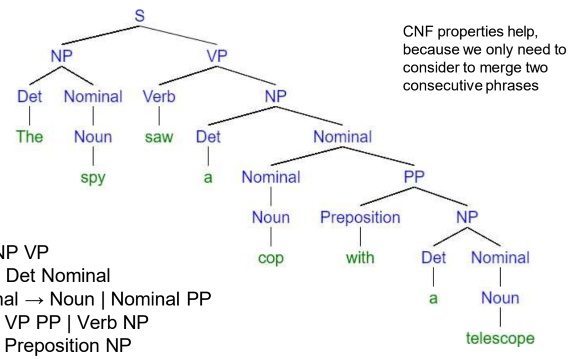
Lexicon 专门词汇
- 限定词 Det → The|a
- Noun → spy|cop|telescope
- Verb → saw
- Preposition → with

Is this CNF?

How to generate

Det Noun Verb Noun Det Noun
The spy saw a cop with a telescope
Det Preposition

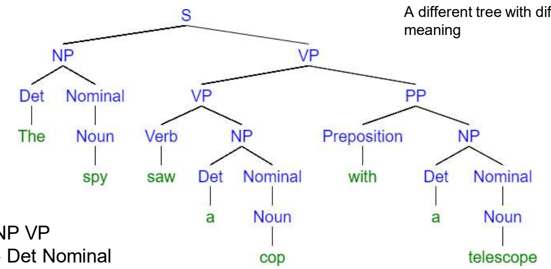
Parsing Natural Language with Context-Free Grammar



CNF properties help, because we only need to consider to merge two consecutive phrases

Rules:
S → NP VP
NP → Det Nominal
Nominal → Noun | Nominal PP
VP → VP PP | Verb NP
PP → Preposition NP

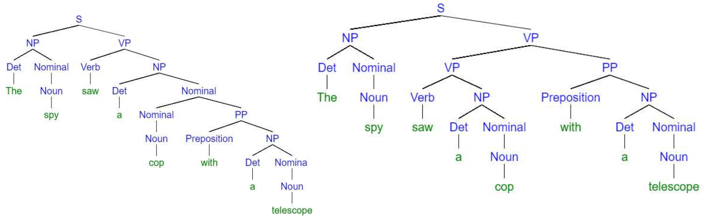
Parsing Natural Language with Context-Free Grammar



A different tree with different meaning

Rules:
S → NP VP
NP → Det Nominal
Nominal → Noun | Nominal PP
VP → VP PP | Verb NP
PP → Preposition NP

Parsing Natural Language with Context-Free Grammar



Different derivation corresponds to different meaning.

Liu, Alisa et al. "We're Afraid Language Models Aren't Modeling Ambiguity." *ArXiv* abs/2304.14399 (2023): n. pag.

下推自动机

Pushdown Automata (PDAs)

能被 pushdown automate 接受的语言就是上下文无关语言

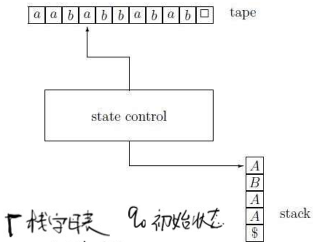
The class of languages that can be accepted by pushdown automata is exactly the class of context-free languages (finite automata are for regular languages).

- The input for a pushdown automaton is a string w in Σ^*. pda 的输入是 Σ^* 中的一个字符串
- PDA accepts or doesn't accept w. PDA 可以接受或不接受 w
- Different from finite automata, PDAs have a stack. 和有限自动机不同，PDA 有栈
- Stack have 2 different operations: 有两种不同的操作
  (1) push – adds item to top of stack 推入 将内容放入栈顶
  (2) pop – removes item from top of stack 推出 从栈顶移出元素

Pushdown Automata (PDAs)

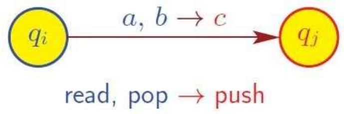
A PDA consists of: a tape, a stack and a state control

- **Tape**: divided into cells that store symbols belonging to $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.
- **Tape head**: move along the tape, one cell to the right per move.
- **Stack**: containing symbols from a finite set Γ, called the stack alphabet. This set contains a special symbol \$ (often mark bottom of stack).
- **Stack head**: reads the top symbol of the stack. This head can also pop the top symbol, and it can push symbols of Γ onto the stack.
- **State control**: can be in any one of a finite number of states. The set of states is denoted by Q. The set Q contains one special state q, called the start state.



Γ 栈字母表　q_0 初始状态
Q 非空有穷集合　Z_0 栈底符号
Σ 输入字母表　F 接受状态集合
δ 转移函数

PDA Transition



If PDA

- in state q_i
- reads $a \in \Sigma_\varepsilon$
- pops $b \in \Gamma_\varepsilon$ off the stack

If $a = \varepsilon$, then no input symbol is read.

If $b = \varepsilon$, then nothing is popped off stack.

then PDA

- moves to state q_j
- push $c \in \Gamma_\varepsilon$ onto top of stack

If $c = \varepsilon$, then b is popped from stack.

If $c = u_1u_2 \ldots u_k$ with $k \geq 1$ and $u_1, u_2, \ldots, u_k \in \Gamma$, then b is replaced by c, and u_k becomes the new top symbol of the stack .

PDA Definition

Definition

后进先出自动机

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

- Q is finite set of states
- Σ is (finite) input (tape) alphabet
- Γ is (finite) stack alphabet
- δ is the transition function: 状态 输入 栈 $Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow Q \times \Gamma_\varepsilon{}^*$
- q is start state, $q \in Q$
- F is set of accept states, $F \subseteq Q$, **PDA accepts as long as it is in F regardless of the stack.**

Γ 栈字母表　q_0 初始状态
Q 非空有穷集合　Z_0 栈底符号
Σ 输入字母表　F 接受状态集合
δ 转移函数

Let $r, r' \in Q$, $a \in \Sigma^*$ and $b, c \in \Gamma^*$

$$\delta(r, a, b) = (r', c).$$

In state r, PDA reads a on the tape and pop b from the stack, move to state r′ and push c to the stack. The tape head moves to the right.

PDA 从 tape 读取a, 把b从栈顶弹出, 移动到下一个状态 r′, 把c推入栈, 将状态移到下一个 tape.

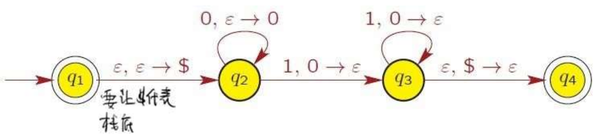
Example

Given a PDA $M = (Q, \Sigma, \Gamma, \delta, q_1, F)$

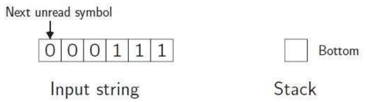
- $Q = \{q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, \$\}$
- q_1 is start state
- $F = \{q_1, q_4\}$
- δ: $Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow Q \times \Gamma_\varepsilon{}^*$

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

Example

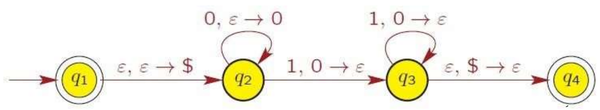


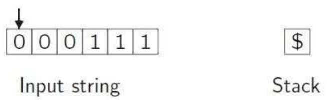
Process string 000111



- Start in start state q_1 with stack empty.
- No input symbols read so far.
- Next go to state q_2
  - reading nothing, popping nothing, and pushing \$ on stack.

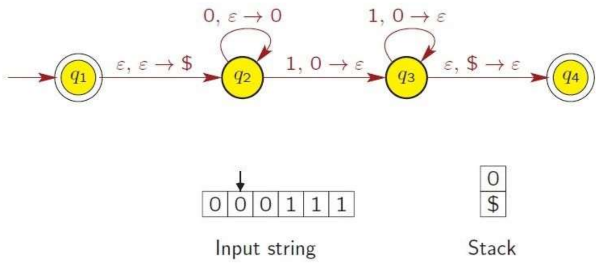
Example





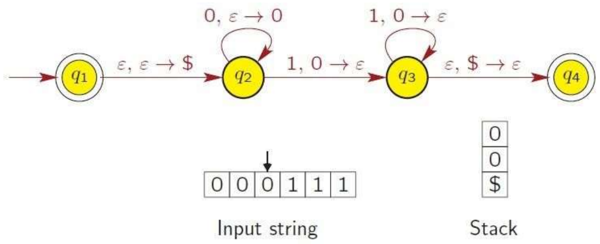
- Next return to state q_2
  - reading input symbol 0
  - popping nothing from stack
  - pushing 0 on stack.

Example



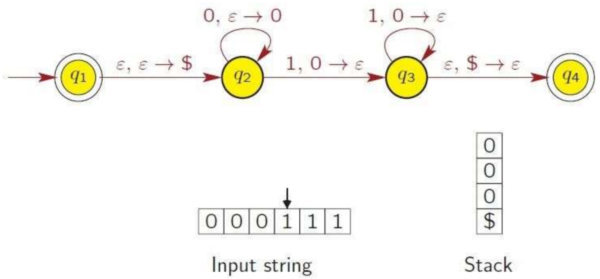
- Next return to state q_2
  - reading input symbol 0
  - popping nothing from stack
  - pushing 0 on stack.

Example



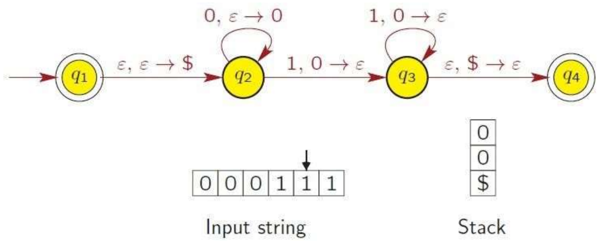
- Next return to state q_2
  - reading input symbol 0
  - popping nothing from stack
  - pushing 0 on stack.

Example



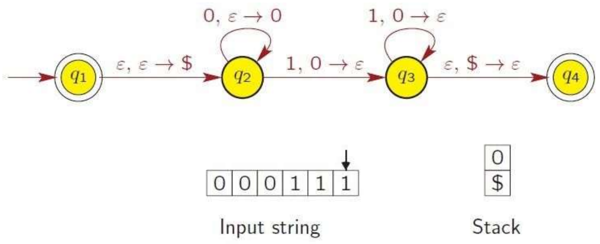
- Next go to state q_3
  - reading input symbol 1
  - popping 0 from stack
  - pushing nothing on stack.

Example



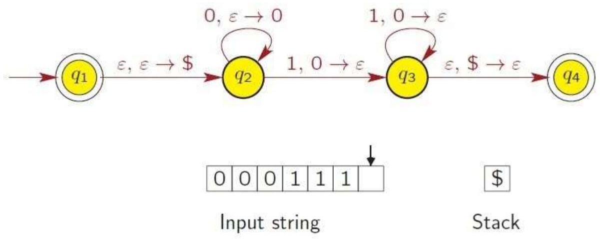
- Next return to state q_3
  - reading input symbol 1
  - popping 0 from stack
  - pushing nothing on stack.

Example



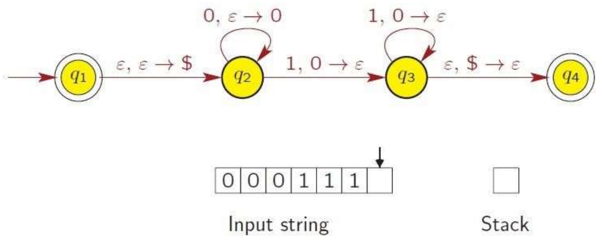
- Next return to state q_3
  - reading input symbol 1
  - popping 0 from stack
  - pushing nothing on stack.

Example



- Next go to state q_4
  - reading nothing
  - popping $ from stack
  - pushing nothing on stack.

Example

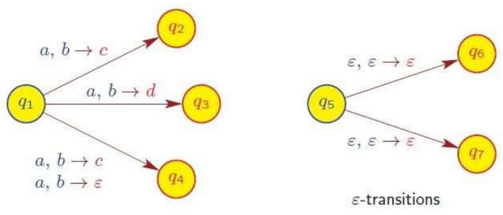


- String 000111 is **accepted** by PDA because
  - q_4 is an accept state and
  - PDA read the entire input string without crashing.

Nondeterministic PDA

PDA transition function allows for nondeterminism

$$\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to P(Q \times \Gamma_\varepsilon)$$



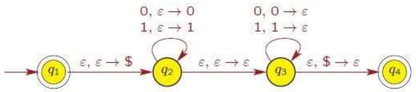
Language accepted by PDA

Definition

The set of all input strings that are accepted by PDA M is the language recognized by M and is denoted by L(M).

Example

PDA for language $\{ww^R \mid w \in \{0, 1\}^*\}$

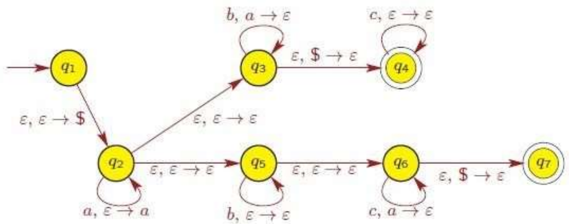


- $q_1 \to q_2$: First pushes \$ on stack to mark bottom
- $q_2 \to q_2$: Reads in first half w of string, pushing it onto stack
- $q_2 \to q_3$: Guesses that it has reached middle of string
- $q_3 \to q_3$: Reads second half $w^{\mathcal{R}}$ of string, matching symbols from first half in reverse order (recall: stack LIFO)
- $q_3 \to q_4$: Makes sure that no more input symbols on stack

Language accepted by PDA

Example

PDA for language $\{ a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k \}$



Quick review

- CFLs are closed under concatenation, union and Kleene closure
- CFLs/Natural language exhibits ambiguities (* optional)
- Pushdown automata has an additional stack to store information

Q&A

- Does the stack elements have any influence on the accepting condition of PDA?

  No, the acceptance is solely decidec by the state.

- Why we put the \$ at the beginning for some PDA?

  Combined with poping \$ before accepting, we make sure that all things being added later will be processed. Back to the first question, if a stack conditioned PDA is defined by accepting when its' state is accepting and stack elements match some criteria. We can add popping transitions to make an equivalent standard PDA.


THANK YOU
Xi'an Jiaotong-Liverpool University
西交利物浦大学
XJTLU
SCHOOL OF
FILM AND
TV ARTS