第 4 章　上下文无关语言

(Part 2 of 2)

王　鑫

wangx@tju.edu.cn

天津大学　智能与计算学部

Outline

1. Context-Free Grammars
2. Pushdown Automata
3. Non-Context-Free Languages

Context-Free Languages

regular languages 正则语言

- finite automata: DFA / NFA
- regular expressions

some simple languages, such as $\{0^n1^n \mid n \geq 0\}$, are ***not*** regular languages.

Context-Free Languages

regular languages 正则语言

- finite automata: DFA / NFA
- regular expressions

some simple languages, such as $\{0^n1^n \mid n \geq 0\}$, are ***not*** regular languages.

context-free languages 上下文无关语言

- pushdown automata 下推自动机

- first used in the study of human languages
- in the specification and compilation of programming languages
  - ***parser***
  - the construction of a parser from a context-free grammar

Outline

1. Context-Free Grammars
   - Formal Definition of a Context-Free Grammar
   - Examples of a Context-Free Grammar
   - Designing Context-Free Grammars
   - Ambiguity
   - Chomsky Normal Form
2. Pushdown Automata
3. Non-Context-Free Languages

Outline

1. Context-Free Grammars

2. Pushdown Automata
   - Formal Definition of a Pushdown Automaton
   - Examples of Pushdown Automata
   - Equivalence With Context-Free Grammars

3. Non-Context-Free Languages

Pushdown Automata 下推自动机

Pushdown Automata (PDA): we introduce a new type of computational model.

- like NFA but have an extra component called a ***stack***.

Pushdown Automata 下推自动机

Pushdown Automata (PDA): we introduce a new type of computational model.

- like NFA but have an extra component called a ***stack***.
- the stack provides additional memory beyond the finite amount available in the control.

Pushdown Automata 下推自动机

Pushdown Automata (PDA): we introduce a new type of computational model.

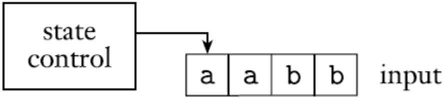
- like NFA but have an extra component called a ***stack***.
- the stack provides additional memory beyond the finite amount available in the control.
- the stack allows PDA to recognize some nonregular languages.

PDA are equivalent in power to CFG

- two options for proving that a language is context free
  - give either a CFG generating it (generator)
  - or a PDA recognizing it (recognizer)

Pushdown Automata 下推自动机

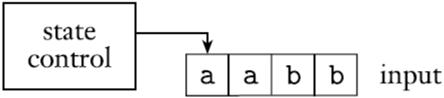
DFA/NFA vs. PDA



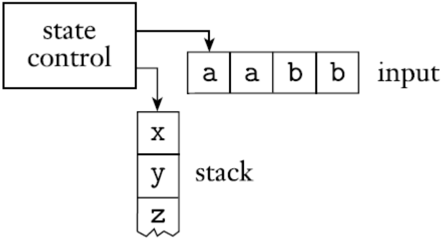
Schematic of DFA/NFA

Pushdown Automata 下推自动机

DFA/NFA vs. PDA



Schematic of DFA/NFA



Schematic of PDA

Pushdown Automata 下推自动机

PDA can write symbols on the stack and read them back later.

Writing a symbol ***"pushes down"*** all the other symbols on the stack.

- all access to the stack may be done only at the top: "last in, first out"
- ***pushing***: writing a symbol on the top of the stack
- ***popping***: removing a symbol on the top of the stack

A stack can hold an unlimited amount of information.

the language $\{0^n1^n \mid n \geq 0\}$

- a DFA/NFA is unable to recognize it.
- A PDA is able to recognize it.

Pushdown Automata 下推自动机

Deterministic and nondeterministic PDA are ***not*** equivalent in power.

- Nondeterministic PDA recognize certain languages that ***no*** deterministic PDA can recognize.
- Recall that DFA and NFA do recognize the same class of languages.
- So the pushdown automata situation is different.
- We focus on nondeterministic PDA because these automata are equivalent in power to CFG.

Formal Definition of a Pushdown Automaton

定义 (PDA (下推自动机))

Formal Definition of a Pushdown Automaton

定义 (PDA (下推自动机))

A ***pushdown automaton*** (PDA) is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where

Formal Definition of a Pushdown Automaton

定义 (PDA (下推自动机))

A ***pushdown automaton*** (PDA) is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where

1. Q is a finite set of ***states***,

Formal Definition of a Pushdown Automaton

定义 (PDA (下推自动机))

A ***pushdown automaton*** (PDA) is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where

1. Q is a finite set of ***states***,
2. Σ is a finite set called the ***input alphabet***,

Formal Definition of a Pushdown Automaton

定义 (PDA (下推自动机))

A ***pushdown automaton*** (PDA) is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where

1. Q is a finite set of ***states***,
2. Σ is a finite set called the ***input alphabet***,
3. Γ is a finite set called the ***stack alphabet***,

Formal Definition of a Pushdown Automaton

定义 (PDA (下推自动机))

A ***pushdown automaton*** (PDA) is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where

1. Q is a finite set of ***states***,
2. Σ is a finite set called the ***input alphabet***,
3. Γ is a finite set called the ***stack alphabet***,
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the ***transition function***,

Formal Definition of a Pushdown Automaton

定义 (PDA (下推自动机))

A ***pushdown automaton*** (PDA) is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where

1. Q is a finite set of ***states***,
2. Σ is a finite set called the ***input alphabet***,
3. Γ is a finite set called the ***stack alphabet***,
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the ***transition function***,
5. $q_0 \in Q$ is the ***start state***, and

Formal Definition of a Pushdown Automaton

定义 (PDA (下推自动机))

A ***pushdown automaton*** (PDA) is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where

1. Q is a finite set of ***states***,
2. Σ is a finite set called the ***input alphabet***,
3. Γ is a finite set called the ***stack alphabet***,
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the ***transition function***,
5. $q_0 \in Q$ is the ***start state***, and
6. $F \subseteq Q$ is the set of ***accept states***.

Formal Definition of Computation for a PDA

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ computes as follows.

- It accepts input w if w can be written as $w = a_1 a_2 \cdots a_m$, where $a_i \in \Sigma_\varepsilon$ and

Formal Definition of Computation for a PDA

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ computes as follows.

- It accepts input w if w can be written as $w = a_1 a_2 \cdots a_m$, where $a_i \in \Sigma_\varepsilon$ and
- sequences of states $r_0, r_1, \ldots, r_m \in Q$ and strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ exist that satisfy the following three conditions.

Formal Definition of Computation for a PDA

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ computes as follows.

- It accepts input w if w can be written as $w = a_1 a_2 \cdots a_m$, where $a_i \in \Sigma_\varepsilon$ and
- sequences of states $r_0, r_1, \ldots, r_m \in Q$ and strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ exist that satisfy the following three conditions.
- The strings s_i represent the sequence of stack contents that M has on the accepting branch of the computation.

Formal Definition of Computation for a PDA

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ computes as follows.

- It accepts input w if w can be written as $w = a_1 a_2 \cdots a_m$, where $a_i \in \Sigma_\varepsilon$ and
- sequences of states $r_0, r_1, \ldots, r_m \in Q$ and strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ exist that satisfy the following three conditions.
- The strings s_i represent the sequence of stack contents that M has on the accepting branch of the computation.

  1. $r_0 = q_0$ and $s_0 = \varepsilon$

Formal Definition of Computation for a PDA

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ computes as follows.

- It accepts input w if w can be written as $w = a_1 a_2 \cdots a_m$, where $a_i \in \Sigma_\varepsilon$ and
- sequences of states $r_0, r_1, \ldots, r_m \in Q$ and strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ exist that satisfy the following three conditions.
- The strings s_i represent the sequence of stack contents that M has on the accepting branch of the computation.

1. $r_0 = q_0$ and $s_0 = \varepsilon$
2. For $i = 0, \ldots, m-1$, we have $(r_{i+1}, b) \in \delta(r_i, a_{i+1}, a)$, where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$

Formal Definition of Computation for a PDA

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ computes as follows.

- It accepts input w if w can be written as $w = a_1 a_2 \cdots a_m$, where $a_i \in \Sigma_\varepsilon$ and
- sequences of states $r_0, r_1, \ldots, r_m \in Q$ and strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ exist that satisfy the following three conditions.
- The strings s_i represent the sequence of stack contents that M has on the accepting branch of the computation.
  1. $r_0 = q_0$ and $s_0 = \varepsilon$
  2. For $i = 0, \ldots, m-1$, we have $(r_{i+1}, b) \in \delta(r_i, a_{i+1}, a)$, where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$
  3. $r_m \in F$

Examples of Pushdown Automata

例 (PDA M_1 recognizes the language $\{0^n 1^n \mid n \geq 0\}$)

Let M_1 be $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

$Q = \{q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, \$\}$

$F = \{q_1, q_4\}$, and

δ is given by the following table, wherein blank entries signify $\emptyset$

Examples of Pushdown Automata

例 (PDA M_1 recognizes the language $\{0^n 1^n \mid n \geq 0\}$)

Let M_1 be $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

$Q = \{q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, \$\}$

$F = \{q_1, q_4\}$, and

δ is given by the following table, wherein blank entries signify $\emptyset$

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									

Examples of Pushdown Automata

例 (PDA M_1 recognizes the language $\{0^n1^n \mid n \geq 0\}$)

Let M_1 be $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

$Q = \{q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$ $\Gamma = \{0, \$\}$ $F = \{q_1, q_4\}$, and

State diagram for the PDA M_1

Examples of Pushdown Automata

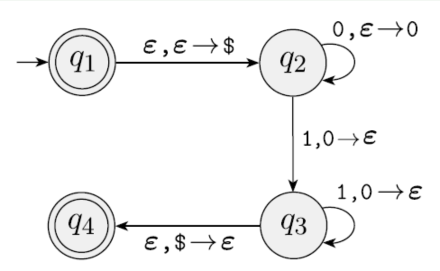
例 (PDA M_1 recognizes the language $\{0^n 1^n \mid n \geq 0\}$)

Let M_1 be $(Q, \Sigma, \Gamma, \delta, q_1, F)$, where

$Q = \{q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$ $\Gamma = \{0, \$\}$ $F = \{q_1, q_4\}$, and

State diagram for the PDA M_1



Examples of Pushdown Automata

例 (PDA M_2)

A pushdown automaton that recognizes the language

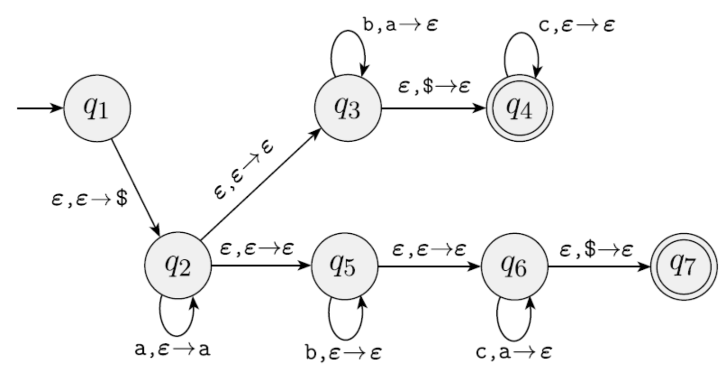
$\{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$

Examples of Pushdown Automata

例 (PDA M_2)

A pushdown automaton that recognizes the language

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



Examples of Pushdown Automata

例 (PDA M_3)

A pushdown automaton that recognizes the language

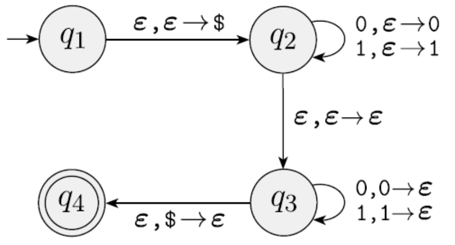
$\{ww^{\mathcal{R}} \mid w \in \{0,1\}^*\}$

Examples of Pushdown Automata

例 (PDA M_3)

A pushdown automaton that recognizes the language

$\{ww^{\mathcal{R}} \mid w \in \{0,1\}^*\}$



Equivalence With Context-Free Grammars

定理

A language is context free if and only if some pushdown automaton recognizes it.

Equivalence With Context-Free Grammars

定理

A language is context free if and only if some pushdown automaton recognizes it.

引理

If a language is context free, then some pushdown automaton recognizes it.

Equivalence With Context-Free Grammars

定理

A language is context free if and only if some pushdown automaton recognizes it.

引理

If a language is context free, then some pushdown automaton recognizes it.

引理

If a pushdown automaton recognizes some language, then it is context free.

Equivalence With Context-Free Grammars

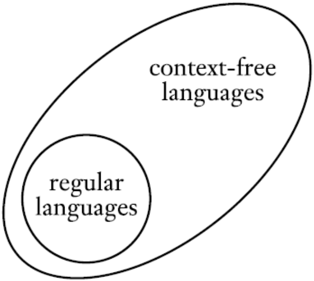
推论

Every regular language is context free.

Equivalence With Context-Free Grammars

推论

Every regular language is context free.



Outline

1 Context-Free Grammars

2 Pushdown Automata

3 Non-Context-Free Languages

- The Pumping Lemma for Context-Free Languages

The Pumping Lemma for Context-Free Languages

定理 (Pumping lemma for context-free languages 泵引理)

If A is a context-free language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into five pieces, $s = uvxyz$, satisfying the conditions:

The Pumping Lemma for Context-Free Languages

定理 (Pumping lemma for context-free languages 泵引理)

If A is a context-free language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into five pieces, $s = uvxyz$, satisfying the conditions:

1. for each $i \geq 0$, $uv^ixy^iz \in A$,

The Pumping Lemma for Context-Free Languages

定理 (Pumping lemma for context-free languages 泵引理)

If A is a context-free language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into five pieces, $s = uvxyz$, satisfying the conditions:

1. *for each* $i \geq 0$, $uv^ixy^iz \in A$,
2. $|vy| > 0$, *and*

The Pumping Lemma for Context-Free Languages

定理 (Pumping lemma for context-free languages 泵引理)

If A is a context-free language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into five pieces, $s = uvxyz$, satisfying the conditions:

1. *for each $i \geq 0$, $uv^ixy^iz \in A$,*
2. *$|vy| > 0$, and*
3. *$|vxy| \leq p$.*