Theoretical computer science

Lecture and tutorial - week 6

February 25, 2021

Agenda

- Pumping lemma: recap
- Pushdown Automaton (Deterministic PDA)
  - Notion
  - formal definition
  - configuration
  - transition
  - acceptance
- Examples

Administrative Information

When : Thursday, 4 March 2021 09:30-~11:30

Where : 108 (here) and possibly another room

+ online for abroad students only

What: Formal Languages, FSA, Pumping Lemma, PDA (only in part)

Topics for the midterm

- Finite State Automata
- Finite State Transducers
- Operations on FSA
- Regular Languages
- Pumping Lemma
- Pushdown Automata

Pumping lemma for regular languages

How Pumping lemma is useful?

We can use it to prove that a language is not regular. How?

Proof by contrapositive

$$R \implies P$$

$$\neg P \implies \neg R$$

Pumping lemma: formally

$$
\begin{array}{l}
\forall L \subseteq \Sigma^* \bullet \mathit{regular}(L) \implies \\
\quad (\exists m \in \mathbb{N} \bullet m \geq 1 \wedge \\
\quad\quad (\forall w \in L \bullet \mid w \mid \geq m \implies \\
\quad\quad\quad (\exists x, y, z \in \Sigma^* \bullet w = xyz \wedge (\mid y \mid \geq 1 \wedge \mid xy \mid \leq m \wedge \\
\quad\quad\quad\quad (\forall i \geq 0 \bullet xy^i z \in L))))
\end{array}
$$

Example 1

Let's consider language L_1

$$L_1 = \{a^n b^m \mid n \leq m\}$$

Is L_1 a regular language?

Example 1

Let's consider language L_1

$$L_1 = \{a^n b^m \mid n \leq m\}$$

Is L_1 a regular language?

$$\begin{array}{l} regular(L_1) \implies \\ \quad (\exists m \in \mathbb{N} \bullet m \geq 1 \land \\ \quad\quad (\forall w \in L_1 \bullet \mid w \mid \geq m \implies \\ \quad\quad\quad (\exists x, y, z \in \Sigma^* \bullet w = xyz \land \mid y \mid \geq 1 \land \mid xy \mid \leq m \land \\ \quad\quad\quad\quad (\forall i \geq 0 \bullet xy^i z \in L_1)))) \end{array}$$

Example 1

Let's consider language L_1

$$L_1 = \{a^n b^m \mid n \leq m\}$$

Is L_1 a regular language?

Which is equivalent to ...

$$\begin{aligned}
&\neg(\exists m \in \mathbb{N} \bullet m \geq 1 \wedge \\
&\quad (\forall w \in L_1 \bullet \mid w \mid \geq m \implies \\
&\qquad (\exists x, y, z \in \Sigma^* \bullet w = xyz \wedge \mid y \mid \geq 1 \wedge \mid xy \mid \leq m \wedge \\
&\qquad\quad (\forall i \geq 0 \bullet xy^i z \in L_1)))) \implies \neg regular(L_1)
\end{aligned}$$

Negation

The negation of a universal quantifier:

$$\neg(\forall x \bullet P(x)) \text{ is logically equivalent to } \exists x \bullet \neg P(x)$$

Negation

The negation of a universal quantifier:

$$\neg(\forall x \bullet P(x)) \text{ is logically equivalent to } \exists x \bullet \neg P(x)$$

The negation of a existential quantifier:

$$\neg(\exists x \bullet P(x)) \text{ is logically equivalent to } \forall x \bullet \neg P(x)$$

Negation

The negation of a universal quantifier:

$$\neg(\forall x \bullet P(x)) \text{ is logically equivalent to } \exists x \bullet \neg P(x)$$

The negation of a existential quantifier:

$$\neg(\exists x \bullet P(x)) \text{ is logically equivalent to } \forall x \bullet \neg P(x)$$

De Morgan's law:

$$\neg(P \land Q) \text{ is logically equivalent to } \neg P \lor \neg Q$$
$$\neg(P \lor Q) \text{ is logically equivalent to } \neg P \land \neg Q$$

Example 1

Let's consider language L_1

$$L_1 = \{a^n b^m \mid n \leq m\}$$

Is L_1 a regular language?

Which is equivalent to ...

$$\begin{aligned}
&(\forall m \in \mathbb{N} \bullet \neg(m \geq 1) \vee \\
&\quad \neg(\forall w \in L_1 \bullet \mid w \mid \geq m \implies \\
&\qquad (\exists x, y, z \in \Sigma^* \bullet w = xyz \wedge \mid y \mid \geq 1 \wedge \mid xy \mid \leq m \wedge \\
&\qquad\quad (\forall i \geq 0 \bullet xy^i z \notin L_1)))) \implies \neg regular(L_1)
\end{aligned}$$

Negation of an Implication

The negation of an implication is a conjunction:

$$\neg(P \implies Q) \text{ is logically equivalent to } P \wedge \neg Q$$

Example 1

Let's consider language L_1

$$L_1 = \{a^n b^m \mid n \leq m\}$$

Is L_1 a regular language?

Which is equivalent to ...

$$\begin{aligned}
&(\forall m \in \mathbb{N} \bullet \neg(m \geq 1) \vee \\
&\quad (\exists w \in L_1 \bullet \mid w \mid \geq m \wedge \\
&\qquad \neg(\exists x, y, z \in \Sigma^* \bullet w = xyz \wedge \mid y \mid \geq 1 \wedge \mid xy \mid \leq m \wedge \\
&\qquad\quad (\forall i \geq 0 \bullet xy^i z \in L_1)))) \implies \neg regular(L_1)
\end{aligned}$$

Example 1

Let's consider language L_1

$$L_1 = \{a^n b^m \mid n \leq m\}$$

Is L_1 a regular language?

Which is equivalent to ...

$$\begin{aligned}
&(\forall m \in \mathbb{N} \bullet \neg(m \geq 1) \vee \\
&\quad (\exists w \in L_1 \bullet \mid w \mid \geq m \wedge \\
&\qquad (\forall x, y, z \in \Sigma^* \bullet \neg(w = xyz) \vee \neg(\mid y \mid \geq 1) \vee \neg(\mid xy \mid \leq m) \vee \\
&\qquad\quad \neg(\forall i \geq 0 \bullet xy^i z \in L_1)))) \implies \neg regular(L_1)
\end{aligned}$$

Example 1

Let's consider language L_1

$$L_1 = \{a^n b^m \mid n \leq m\}$$

Is L_1 a regular language?

Which is equivalent to ...

$$\begin{aligned}
&(\forall m \in \mathbb{N} \bullet \neg(m \geq 1) \vee \\
&\quad (\exists w \in L_1 \bullet \mid w \mid \geq m \wedge \\
&\quad\quad (\forall x, y, z \in \Sigma^* \bullet \neg(w = xyz) \vee \neg(\mid y \mid \geq 1) \vee \neg(\mid xy \mid \leq m) \vee \\
&\quad\quad\quad (\exists i \geq 0 \bullet \neg(xy^i z \in L_1))))) \implies \neg regular(L_1)
\end{aligned}$$

Example 1

Let's consider language L_1

$$L_1 = \{a^n b^m \mid n \leq m\}$$

Is L_1 a regular language?

Which is equivalent to ...

$$\begin{array}{l}(\forall m \in \mathbb{N} \bullet m < 1 \vee \\ \quad (\exists w \in L_1 \bullet \mid w \mid \geq m \wedge \\ \quad\quad (\forall x, y, z \in \Sigma^* \bullet (w \neq xyz) \vee (\mid y \mid < 1) \vee (\mid xy \mid > m) \vee \\ \quad\quad\quad (\exists i \geq 0 \bullet xy^i z \notin L_1)))) \implies \neg regular(L_1)\end{array}$$

Disjunction elimination

But before eliminating $\neg$, let us eliminate $\vee$'s

$$P \vee Q \text{ is logically equivalent to } \neg P \implies Q$$

Or, more generally:

$$Q_1 \vee \cdots \vee Q_{n-1} \vee Q_n$$

is logically equivalent to

$$\neg Q_1 \implies (\cdots \implies (\neg Q_{n-1} \implies Q_n) \ldots)$$

Example 1

Let's consider language L_1

$$L_1 = \{a^n b^m \mid n \leq m\}$$

Is L_1 a regular language?

Which is equivalent to ...

$$
\begin{array}{l}
(\forall m \in \mathbb{N} \bullet \neg\neg(m \geq 1) \implies \\
\quad (\exists w \in L_1 \bullet \mid w \mid \geq m \wedge \\
\quad\quad (\forall x, y, z \in \Sigma^* \bullet \neg\neg(w = xyz) \implies \\
\quad\quad\quad (\neg\neg(\mid y \mid \geq 1) \implies (\neg\neg(\mid xy \mid \leq m) \implies \\
\quad\quad\quad\quad (\exists i \geq 0 \bullet \neg(xy^i z \in L_1))))))) \implies \neg regular(L_1)
\end{array}
$$

Example 1

Let's consider language L_1

$$L_1 = \{a^n b^m \mid n \leq m\}$$

Is L_1 a regular language?

Which is equivalent to ...

$$
\begin{array}{l}
(\forall m \in \mathbb{N} \bullet m \geq 1 \implies \\
\quad (\exists w \in L_1 \bullet \mid w \mid \geq m \wedge \\
\quad\quad (\forall x, y, z \in \Sigma^* \bullet w = xyz \implies \\
\quad\quad\quad (\mid y \mid \geq 1 \implies (\mid xy \mid \leq m \implies \\
\quad\quad\quad\quad (\exists i \geq 0 \bullet \neg(xy^i z \in L_1))))))) \implies \neg regular(L_1)
\end{array}
$$

Example 1

Let's consider language L_1

$$L_1 = \{a^n b^m \mid n \leq m\}$$

Is L_1 a regular language?

Which is equivalent to ...

$$
\begin{array}{l}
(\forall m \in \mathbb{N} \bullet m \geq 1 \implies \\
\quad (\exists w \in L_1 \bullet \mid w \mid \geq m \wedge \\
\quad\quad (\forall x, y, z \in \Sigma^* \bullet w = xyz \implies \\
\quad\quad\quad (\mid y \mid \geq 1 \implies (\mid xy \mid \leq m \implies \\
\quad\quad\quad\quad (\exists i \geq 0 \bullet xy^i z \notin L_1)))))) \implies \neg regular(L_1)
\end{array}
$$

Example 1

$L_1 = \{a^n b^k \mid n \leq k\}$

Is L_1 a regular language?

Proof

Let $m \in \mathbb{N}$.

Example 1

$L_1 = \{a^n b^k \mid n \leq k\}$

Is L_1 a regular language?

Proof
Let $m \in \mathbb{N}$. We set $w = a^m b^m$; notice that $w \in L_1$ and $|w| = 2m$ which is $|w| > m$.

Example 1

$L_1 = \{a^n b^k \mid n \leq k\}$

Is L_1 a regular language?

Proof

Let $m \in \mathbb{N}$. We set $w = a^m b^m$; notice that $w \in L_1$ and $|w|= 2m$ which is $|w|> m$. Let $x, y, z \in \{a, b\}^*$ such that $\mid y \mid \geq 1$, $|xy| \leq m$ and $w = xyz$.

Example 1

$L_1 = \{a^n b^k \mid n \leq k\}$

Is L_1 a regular language?

Proof

Let $m \in \mathbb{N}$. We set $w = a^m b^m$; notice that $w \in L_1$ and $|w|= 2m$ which is $|w|> m$. Let $x, y, z \in \{a, b\}^*$ such that $\mid y \mid \geq 1$, $|xy| \leq m$ and $w{=}xyz$. We have $y{=}a^l$ for some $l{\in}\{1, \ldots, m\}$, $x = a^{l'}$ for some $l'{\in}\{0, \ldots, m-l\}$ and $z = a^{m-l-l'} b^m$.

Example 1

$L_1 = \{a^n b^k \mid n \leq k\}$

Is L_1 a regular language?

Proof

Let $m \in \mathbb{N}$. We set $w = a^m b^m$; notice that $w \in L_1$ and $|w|= 2m$ which is $|w|> m$. Let $x, y, z \in \{a, b\}^*$ such that $| y |\geq 1$, $|xy|\leq m$ and $w=xyz$. We have $y=a^l$ for some $l\in\{1, \dots , m\}$, $x = a^{l'}$ for some $l'\in\{0, \dots , m - l\}$ and $z = a^{m-l-l'} b^m$. We set $i = 2$.

Example 1

$L_1 = \{a^n b^k \mid n \leq k\}$

Is L_1 a regular language?

Proof

Let $m \in \mathbb{N}$. We set $w = a^m b^m$; notice that $w \in L_1$ and $|w| = 2m$ which is $|w| > m$. Let $x, y, z \in \{a, b\}^*$ such that $|y| \geq 1$, $|xy| \leq m$ and $w = xyz$. We have $y = a^l$ for some $l \in \{1, \ldots, m\}$, $x = a^{l'}$ for some $l' \in \{0, \ldots, m - l\}$ and $z = a^{m-l-l'} b^m$. We set $i = 2$. We have $xy^2z = a^{m+l} b^m$ with $l \geq 1$, and thus xy^2z not in L_1.

Example 1

$L_1 = \{a^n b^k \mid n \leq k\}$

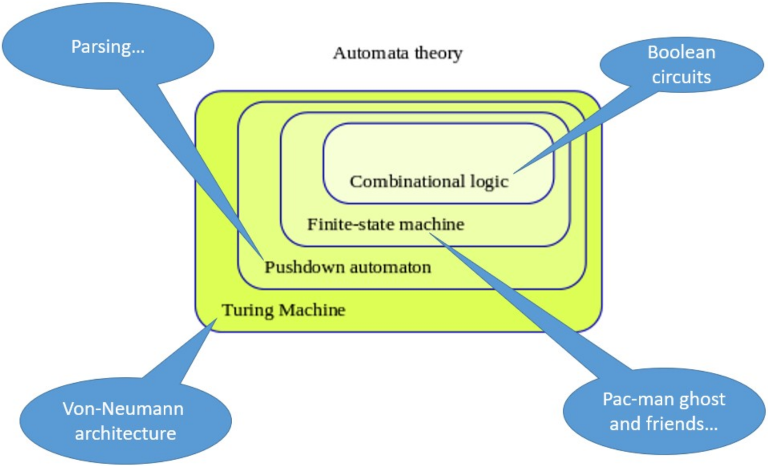
Is L_1 a regular language?

Proof
Let $m \in \mathbb{N}$. We set $w = a^m b^m$; notice that $w \in L_1$ and $|w|= 2m$ which is $|w|> m$. Let $x, y, z \in \{a, b\}^*$ such that $| y |\geq 1$, $|xy|\leq m$ and $w{=}xyz$. We have $y{=}a^l$ for some $l{\in}\{1, \dots, m\}$, $x = a^{l'}$ for some $l'{\in}\{0, \dots, m-l\}$ and $z = a^{m-l-l'} b^m$. We set $i = 2$. We have $xy^2z = a^{m+l} b^m$ with $l \geq 1$, and thus xy^2z not in L_1.

By applying the Pumping lemma for regular languages, we can conclude that that the language L_1 is not regular.

Deterministic Pushdown Automata

Administrative Information



PDA – Notion

- Build a complete FSA that recognises the following language:

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

- It is not possible (you already know how to prove it!)

PDA – Notion

- Build a complete FSA that recognises the following language:

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

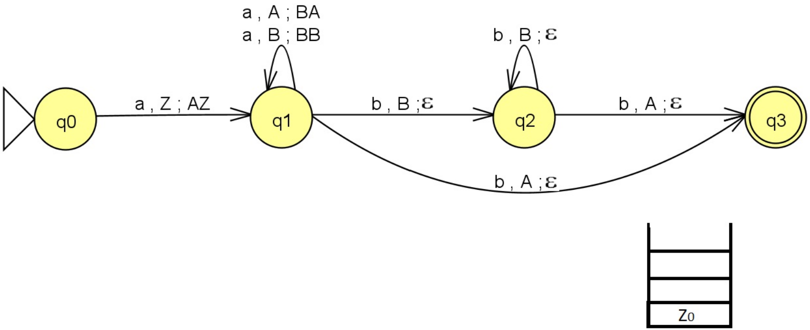
- It is not possible (you already know how to prove it!)
- PDAs are similar to FSA with an auxiliary memory: a stack.

PDA – Notion

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

- When a PDA reads an input symbol, it will be able to save it (or save other symbols) in its memory.
- For deciding if an input string is in the language $AnBn$, the PDA needs to remember the numbers of a's.
- Whenever the PDA reads the input symbol b, two things should happen:
  1. it should change states: from now on the only legal input symbols are b's.
  2. it should delete one a from its memory for every b it reads.

PDA (Graphical representation)



PDA – Notion (Moves)

A single move of a PDA will depend on:

- the current state,
- the next input (it could be no symbol: ϵ symbol), and
- the symbol currently on top of the stack.

PDA will be assumed to begin operation with an initial start symbol Z_0 on its stack and will not be permitted to move unless the stack contains at least one symbol;

- Z_0 is never removed and no additional copies of it are pushed onto the stack
- Z_0 is on top means that the stack is effectively empty.

PDA and compilers

- PDA are **at the heart of compilers**
- Stack memory has a LIFO policy
- LIFO is suitable to analyze **nested syntactic structures**
  - Arithmetical expressions
  - Begin/End
  - Activation records
  - Parenthesized strings
  - …

Balanced Parentheses

Intuitively, a string of parentheses is *balanced* if each left parenthesis has a matching right parenthesis and the matched pairs are well nested. The set PAREN of balanced strings of parentheses [ ] is the prototypical context-free language and plays a pivotal role in the theory of CFLs.

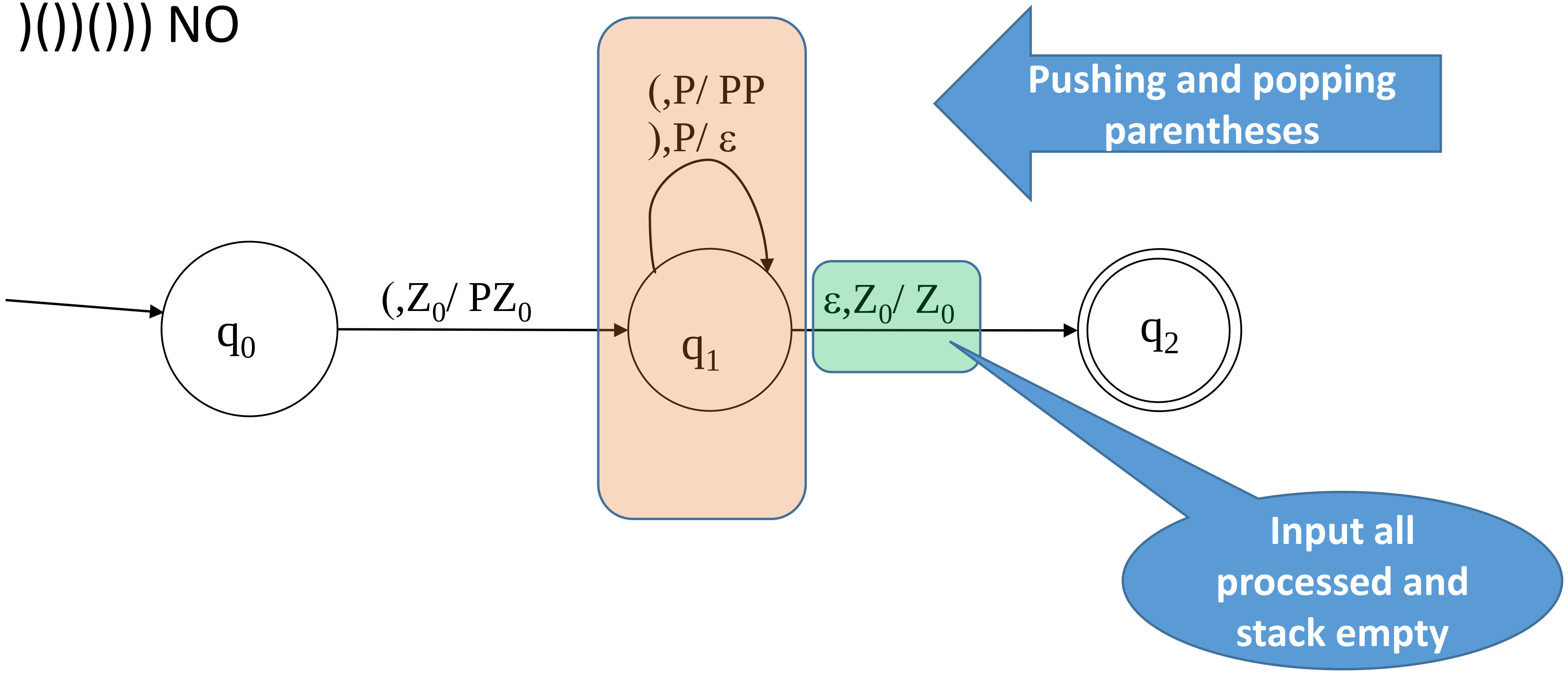
Historical Notes

The pivotal importance of balanced parentheses in the theory of context-free languages was recognized quite early on.

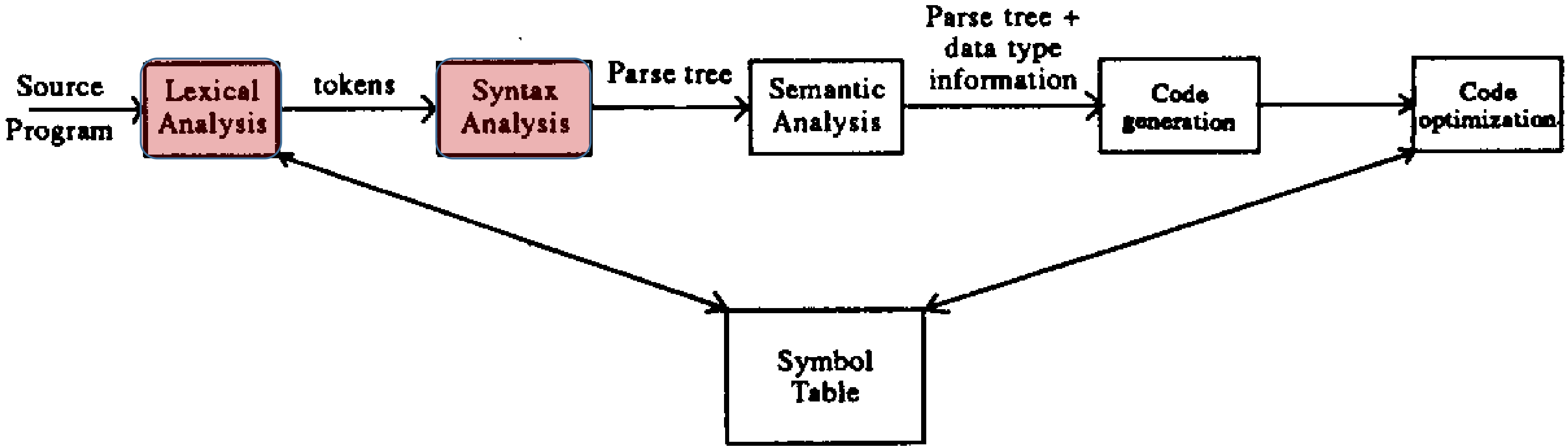
Dexter Kozen, *Automata and Computability*, Springer-Verlag , 1997

Balanced parentheses

- PDA to recognize well parenthesized strings
  - (()((()()))) OK
  -)()))())) NO



General Structure of a Compiler



You will study this in Compilers course

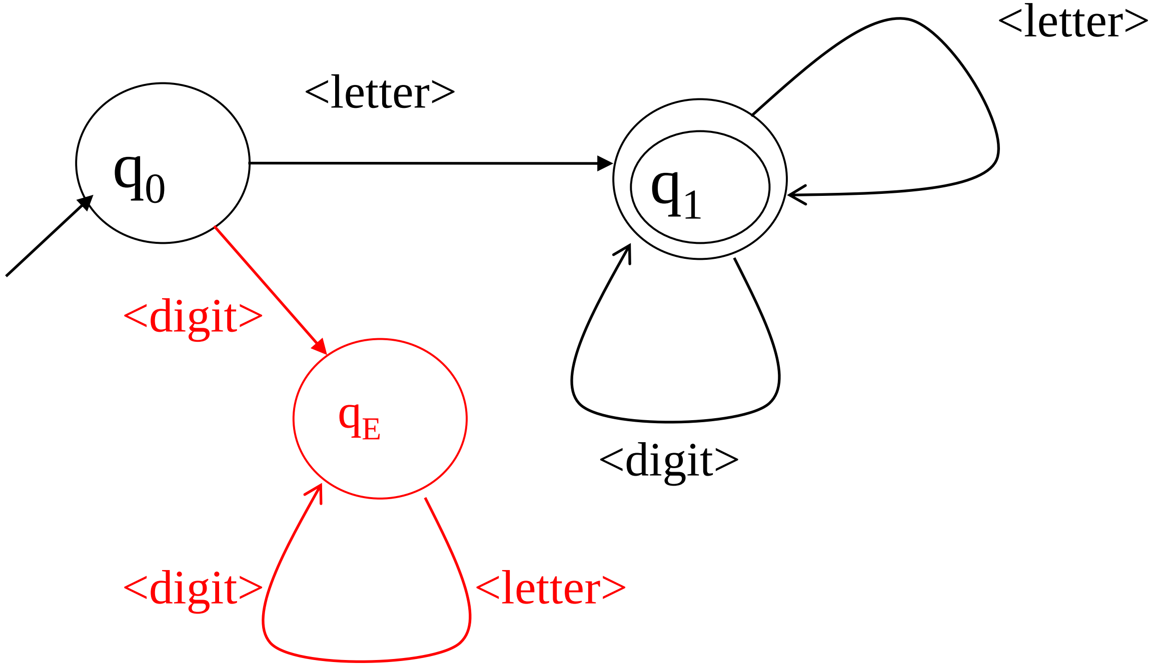
Lexical Analysis

- **Lexical analysis** (lexing/scanning) breaks the source code text into small pieces
  - *Tokens*
  - Single atomic units of the language
  - Keywords, identifiers ...

From Greek *lexikos* 'of words' (from *lexis* 'word')

- **The token syntax is typically a regular language**
  - **Finite State Automaton**, Regular expressions
  - This compiler part is called ***lexer***

Pascal identifiers



Syntax Analysis

- PDA is the most important class of automata between FSA and TM
- FSA cannot even recognize a simple language such as $a^n b^n$
- **Nested structures are the key of programming languages**
- Specific (nondeterministic) PDAs are used in ***Syntax Analysis/parsing***

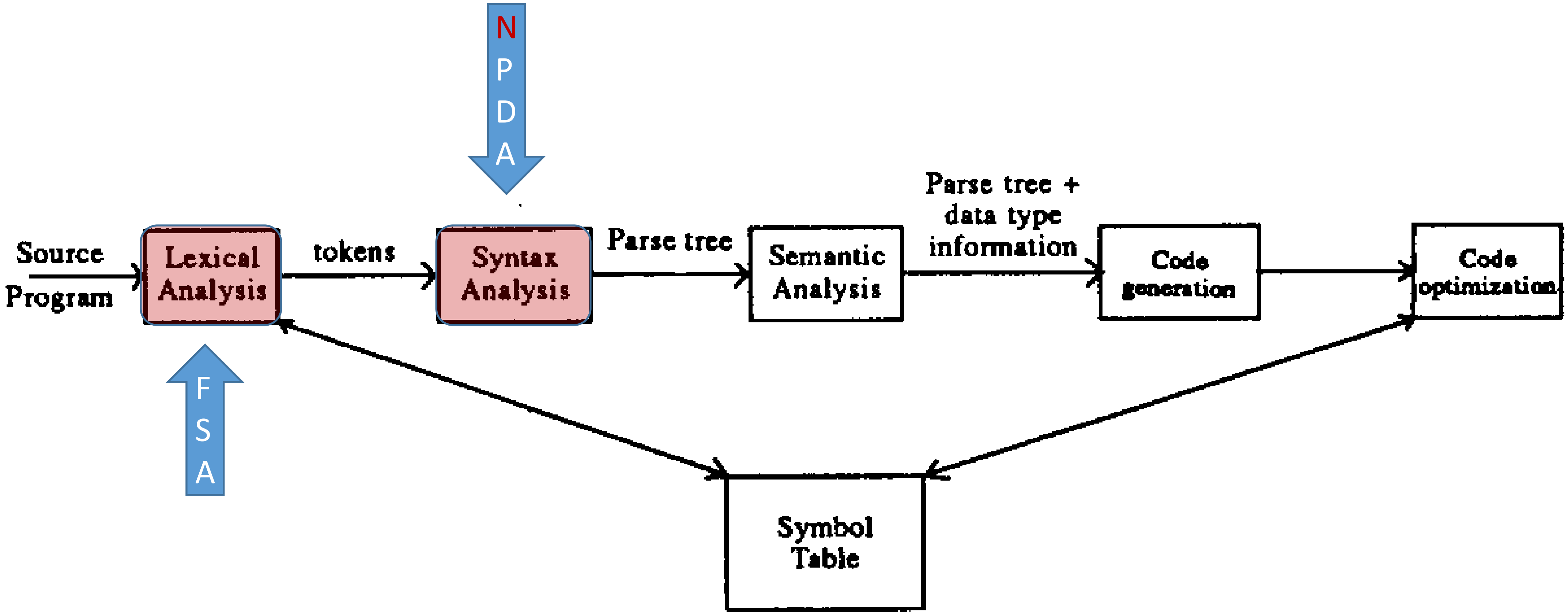
From Latin *pars* (*orationis*): part (of speech)

Context-free languages and PDA

Context-free grammars have played a central role in compiler technology since the 1960s .... There is an automaton-like notation, called the "pushdown automaton", that also describes all and only the context-free languages.

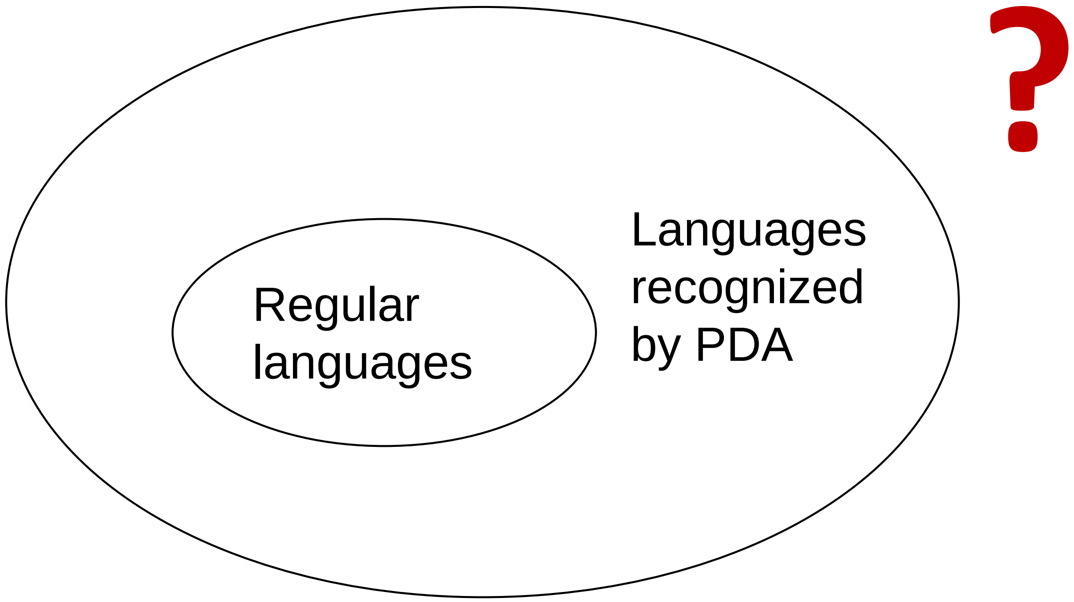
John E. Hopcroft, Rajeev Motwani
and Jeffrey D. Ullman

Very roughly…



You will study this in Compilers course

Everything seems under control...



Are there <u>languages that cannot be recognized by PDAs</u>?

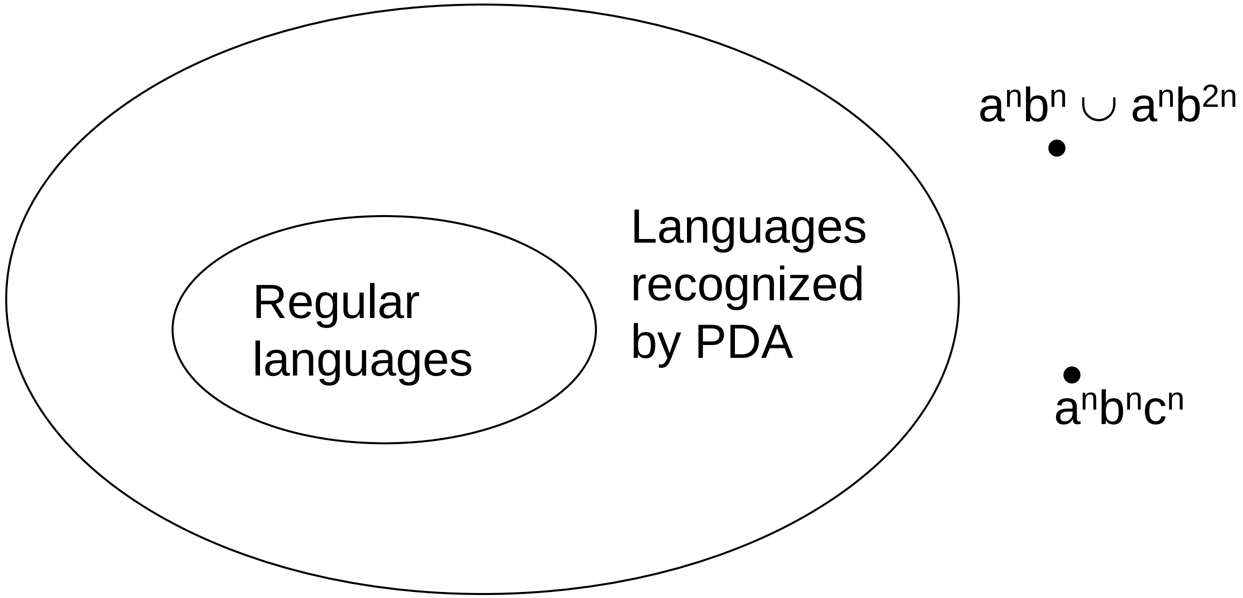
The short answer

- The short answer is yes:
  - there are languages that cannot be recognized by PDAs

- We will look into the details!

- We will also look into the details of PDA formalization
  - Configuration
  - Transitions
  - Transducers

Languages



What are the limits of PDAs?

Remarks

- The stack is a destructive memory
  - Once a symbol is read, it is destroyed
- The limitation of the stack can be proved formally through a generalization of the pumping lemma (lemma of Bar-Hillel)
- It is necessary to use **persistent memory**

  $\rightarrow$ **memory tapes and TM**

PDA – Formal Definition

A Pushdown Automaton

A PDA is a tuple $\langle Q, I, \Gamma, \delta, q_0, Z_0, F \rangle$ where

- Q is a finite set of states.
- I and Γ are finite sets, the input and stack alphabets.
- δ, the transition function, is a partial function from $Q \times (I \cup \{\epsilon\}) \times \Gamma$ to the set of finite subsets of $Q \times \Gamma^*$.
- $q_0 \in Q$, the initial state.
- $Z_0 \in \Gamma$, the initial stack symbol.
- $F \subseteq Q$, the set of accepting states.

Conditions on Z_0

Let $M = \langle Q, I, \Gamma, \delta, q_0, Z_0, F \rangle$ be a PDA. For $q \in Q$ and $i \in I$ and $A \in \Gamma$

- Z_0 is never removed:

$$\text{if } (q', \alpha) \in \delta(q, i, Z_0), \text{ then } \alpha = \alpha' Z_0 \text{ for some } \alpha'$$

- no additional copies of Z_0 are pushed onto the stack:

$$\text{if } (q', \alpha) \in \delta(q, i, A) \text{ and } A \neq Z_0, \text{ then } Z_0 \text{ does not occur in } \alpha$$

A Deterministic PDA – Formal Definition (the one seen in the lecture)

A Deterministic Pushdown Automaton (DPDA)

A PDA $M = \langle Q, I, \Gamma, \delta, q_0, Z_0, F \rangle$ is deterministic if it satisfies both of the following conditions.

1. For every $q \in Q$, every $x \in I \cup \{\epsilon\}$, and every $\gamma \in \Gamma$, the set $\delta(q, x, \gamma)$ has at most one element.
2. For every $q \in Q$, every $x \in I$, and every $\gamma \in \Gamma$, the two sets $\delta(q, x, \gamma)$ and $\delta(q, \epsilon, \gamma)$ cannot both be non-empty.

Configuration

A configuration is a generalization of the notion of state. It shows:

- the current state,
- the portion of the input string that has not yet been read, and
- the stack.

It is a snapshot of the PDA.

Configuration – Formal Definition

Configuration

A Configuration of the PDA $\langle Q, I, \Gamma, \delta, q_0, Z_0, F \rangle$ is a triple

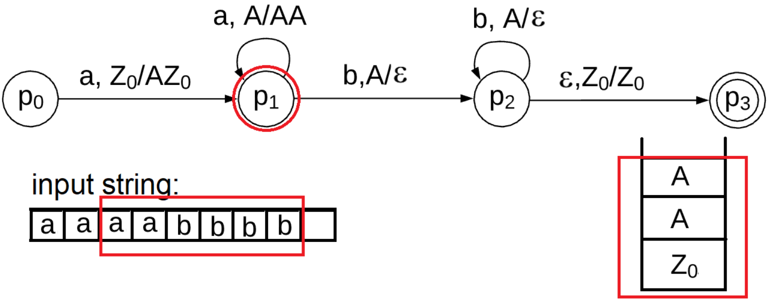
$$(q, x, \gamma)$$

where

- $q \in Q$, is the current state of the control device,
- $x \in I^*$, is the unread portion of the input string, and
- $\gamma \in \Gamma^*$, is the string of symbols in the stack.

Configuration (Graphical representation)

$$(q, x, \gamma)$$



Transition

Transitions between configurations ($\vdash$) depend on the transition function. It is the way to commute from a PDA snapshot to another.

There are 2 cases:

1. The transition function is defined for an input symbol.
2. The transition function is defined for an ϵ move.

Transition – Case 1

If $(q', \alpha) \in \delta(q, i, A)$ then

$$(q, x, \gamma) \vdash (q', x', \gamma')$$

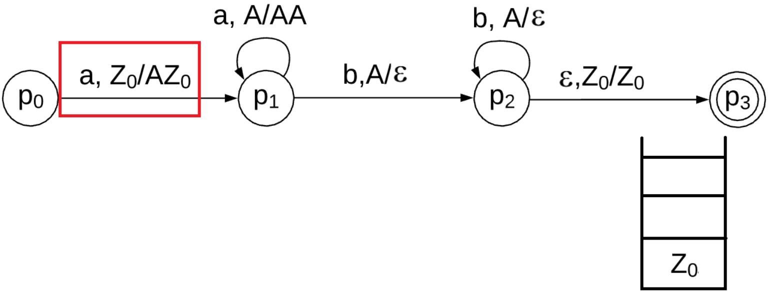
where (old snapshot)

- q is the current state
- $x = iy$
- $\gamma = A\beta$ (for some $\beta \in \Gamma^*$)

then (new snapshot)

- q' is the new state
- $x' = y$
- $\gamma' = \alpha\beta$

Transition – Case 1 (Graphical representation)



Transition – Case 2

If $(q', \alpha) \in \delta(q, \epsilon, A)$ then

$$(q, x, \gamma) \vdash (q', x', \gamma')$$

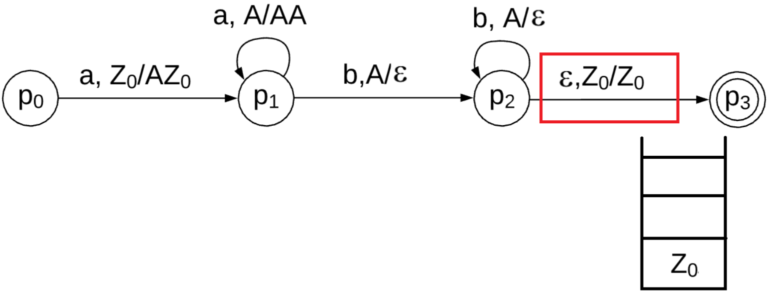
where (old snapshot)

- q is the current state
- $\gamma = A\beta$ (for some $\beta \in \Gamma^*$)

then (new snapshot)

- q' is the new state
- $x' = x$
- $\gamma' = \alpha\beta$

Transition – Case 2 (Graphical representation)



Acceptance – Informally

- A string x is accepted by a PDA if there is a path coherent with x on the PDA that goes from the initial state to the final state.
- The input string has to be read completely.

Acceptance – Formal Definition

Reflexive transitive closure of $\vdash$

Let M be the PDA $\langle Q, I, \Gamma, \delta, q_0, Z_0, F \rangle$, and $c_i = (q, x, \beta)$, $c_j = (q', x', \beta')$ be configurations of M:

$$c_i \vdash^* c_j$$

is the sequence of zero or more moves taking M from c_i to c_j

Acceptance by final state

Let M be the PDA $\langle Q, I, \Gamma, \delta, q_0, Z_0, F \rangle$, and $x \in I^*$. The string x is accepted by M if

$$(q_0, x, Z_0) \vdash^* (q, \epsilon, \gamma)$$

for some $\gamma \in \Gamma^*$ and some $q \in F$.

Acceptance – Formal Definition

Reflexive transitive closure of $\vdash$

Let M be the PDA $\langle Q, I, \Gamma, \delta, q_0, Z_0 \rangle$, and $c_i = (q, x, \beta)$, $c_j = (q', x', \beta')$ be configurations of M:

$$c_i \vdash^* c_j$$

is the sequence of zero or more moves taking M from c_i to c_j

Acceptance by empty stack

Let M be the PDA $\langle Q, I, \Gamma, \delta, q_0, Z_0 \rangle$, and $x \in I^*$. The string x is accepted by M if

$$(q_0, x, Z_0) \vdash^* (q, \epsilon, \epsilon)$$

Examples

Lets consider the following languages:

1. $A_n B_n = \{a^n b^n \mid n \geq 0\}$
2. $A_n B_m C_{n+m} = \{a^n b^m c^{n+m} \mid n, m \geq 0\}$
3. The language of well-parenthesised strings – the alphabet is $I = \{`(', `)'\}$
4. $Palindrom = \{xcx^R \mid x \in \{a, b\} \wedge |x| > 0\}$[1]

[1] where x^R is the reversed string x

Wrap up

- What have you learnt today?

Wrap up

- What have you learnt today?
- What for this could be useful?