Introduction to Computer Science

Géza Horváth

Seventh Lecture

Bar-Hillel lemma – application

Theorem

(Bar-Hillel lemma) *For each context-free language L there exists an integer $n \geq 1$ such that each string $p \in L$, $|p| \geq n$ can be written in a form $p = uvwxy$, where $|vwx| \leq n$, $|vx| \geq 1$ and $uv^i wx^i y \in L$ holds for each integer $i \geq 0$.*

If a language does not satisfy the conditions of the lemma, it is not context-free.

Bar-Hillel lemma – example 1.

The following classical example shows an application of the Bar-Hillel lemma. We are going to prove that language $L = \{a^j b^j c^j \mid j \geq 0\}$ is not context-free.

In order to do this suppose to the contrary that language L is context-free. Let $j \geq (n/3)$, then, by the Bar-Hillel lemma, $a^j b^j c^j$ can be written in a form $uv^i wx^i y$ such that $|vx| \geq 1$ and $uv^i wx^i y \in L$ holds for each integer $i \geq 0$. First, neither of v nor x should contain two or more different letters, because repeating them would change the form of the words in L. So v is a unary word (some power of a letter, e.g. $aa...a$) and x is a unary word as well. In this case, when we increase the integer i, we change the number of one or two different letters, but we cannot change the number of each letter, which is a contradiction.

Bar-Hillel lemma – example 2.

Let us consider the language $L = \{a^j b^k c^j d^k \mid j, k \geq 0\}$.

Suppose to the contrary that language L is context-free. Then, the word $a^n b^n c^n d^n \in L$ can be written in a form $uvwxy$ such that $uv^i wx^i y \in L$ for each $i \geq 0$. Suppose that the word v contains the letter a. In this case, the word x cannot contain the letter c, because $|vwx| \leq n$. Also, if the word v contains the letter b, then the word x cannot contain the letter d. In this case, we cannot increase the number of the letters a and c at the same time, and also we cannot increase the number of the letters b and d together, which means that this language does not satisfy the conditions of the Bar-Hillel lemma, consequently it is not context-free.

Bar-Hillel lemma – example 3.

In this example we consider the language $L = \{wcw \mid w \in \{a, b\}^*\}$, and we use the Bar-Hillel lemma to prove that this language is not context-free.

Suppose to the contrary that the language L is context-free, in this case the word $a^n b^n c a^n b^n$ can be written in a form $uvwxy$ such that $uv^i wx^i y \in L$ for each $i \geq 0$. The only possible solution is that the word v contains letters before the letter c and the word x contains letters after the letter c, because in other cases the number of the letters before and after c will not be the same. Now, $|vwx| \leq n$, so the word v can contain only letter b and the word x can contain only letter a, which is a contradiction.

Bar-Hillel lemma – example 4.

In this example we show that the language $L = \{a^p \mid p \text{ pime}\}$ is not context-free.

Suppose to the contrary that the language L is context-free. Let $p \geq n$, so a^p can be written in a form $uvwxy$ such that $|vx| \geq 1$ and $uv^i wx^i y \in L$ holds for each integer $i \geq 0$. Now, let $q = |vx|$, $r = |uwy|$, so $a^{r+i \cdot q} \in L$ holds for each integer $i \geq 0$. This means that $r + i \cdot q$ is a prime for each integer $i \geq 0$. Here $r \neq 1$, because $1 + i \cdot q = 1$ for $i = 0$, and 1 is not a prime number. Let $i = r$, then $r + r \cdot q$ should be a prime, but $r + r \cdot q = r \cdot (1 + q)$ is not a prime, so we have a contradiction.

Pushdown automata – introduction

Finite automata can accept regular languages, so we have to extend its definition so as it could accept context-free languages. The solution for this problem is to add a stack memory to a finite automaton, and the name of this solution is "pushdown automaton".

Pushdown automata – definition

Definition

A pushdown automaton (PDA) is the following 7-tuple:

$$PDA = (Q, T, Z, q_0, z_0, \delta, F)$$

where

- Q *is the finite nonempty set of the states,*
- T *is the set of the input letters (finite nonempty alphabet),*
- Z *is the set of the stack symbols (finite nonempty alphabet),*
- q_0 *is the initial state,* $q_0 \in Q$,
- z_0 *is the initial stack symbol,* $z_0 \in Z$,
- δ *is the transition function having a form* $Q \times \{T \cup \{\lambda\}\} \times Z \to 2^{Q \times Z^*}$, *and*
- F *is the set of the final states,* $F \subseteq Q$.

Pushdown automata – description

In order to understand the operating principle of the pushdown automaton, we have to understand the operations of finite automata and the stack memory. Finite automata were already introduced, and we studied them through many hours. The stack is a LIFO (last in first out) memory, which has two operations, PUSH and POP. When we use the POP operation, we read the top letter of the stack, and at the same time we delete it. When we use the PUSH operation, we add a word to the top of the stack.

The pushdown automaton accepts words over the alphabet T. At the beginning the PDA is in state q_0, we can read the first letter of the input word, and the stack contains only z_0. In each step, we use the transition function to change the state and the stack of the PDA. The PDA accepts the input word, if and only if it can read the whole word, and it is in a final state when the end of the input word is reached.

Pushdown automata – description

More formally, in each step, the pushdown automaton has a configuration – also called instantaneous description – (q, v, w), where $q \in Q$ is the current state, $v \in T^*$ is the unread part of the input word, and $w \in Z^*$ is the whole word contained by the stack. At the beginning, the pushdown automaton is in its initial configuration: (q_0, p, z_0), where p is the whole input word. In each step, the pushdown automaton changes its configuration, while using the transition function. There are two different kinds of steps, the first is the standard, the second is the so called λ-step.

1. The standard step is when the PDA reads its current state, current input letter, the top stack symbol, it finds an appropriate transition rule, it changes its state, it moves to the next input letter and changes the top symbol of the stack to the appropriate word. Formally, we can say the PDA can change its configuration from (q_1, av, zw) to (q_2, v, rw) in one step, if it has a transition rule $(q_2, r) \in \delta(q_1, a, z)$, where $q_1, q_2 \in Q$, $a \in T$, $z \in Z$, $v \in T^*$, $w \in Z^*$. Denote this transition $(q_1, av, zw) \vdash_{PDA} (q_2, v, rw)$.

Pushdown automata – description

2. The λ-step is when the PDA reads its current state, it does not read any input letters, it reads the top stack symbol, it finds an appropriate transition rule, and it changes its state, it does not move to the next input letter and it changes the top letter of the stack to the given word. Formally, we can say again that the PDA can change its configuration from (q_1, v, zw) to (q_2, v, rw) in one step, if it has a transition rule $(q_2, r) \in \delta(q_1, \lambda, z)$, where $q_1, q_2 \in Q$, $z \in Z$, $v \in T^*$, and $w \in Z^*$. Denote this transition $(q_1, v, zw) \vdash_{PDA} (q_2, v, rw)$.

We can say that the PDA can change its configuration from (q_1, v, w) to (q_2, x, y) in finite many steps, if there are configurations $C_0, C_1, ..., C_n$ such that $C_0 = (q_1, v, w)$, $C_n = (q_2, x, y)$, and $C_i \vdash_{PDA} C_{i+1}$ holds for each integer $0 \leq i < n$. Denote this transition $(q_1, v, w) \vdash^*_{PDA} (q_2, x, y)$.

The language accepted by the pushdown automaton

Finally, we can define the language accepted by the pushdown automaton:

Definition

$$L(PDA) = \{p \mid p \in T^*, (q_0, p, z_0) \vdash^*_{PDA} (q_f, \lambda, y), q_f \in F, y \in Z^*\}.$$

Pushdown automata – example

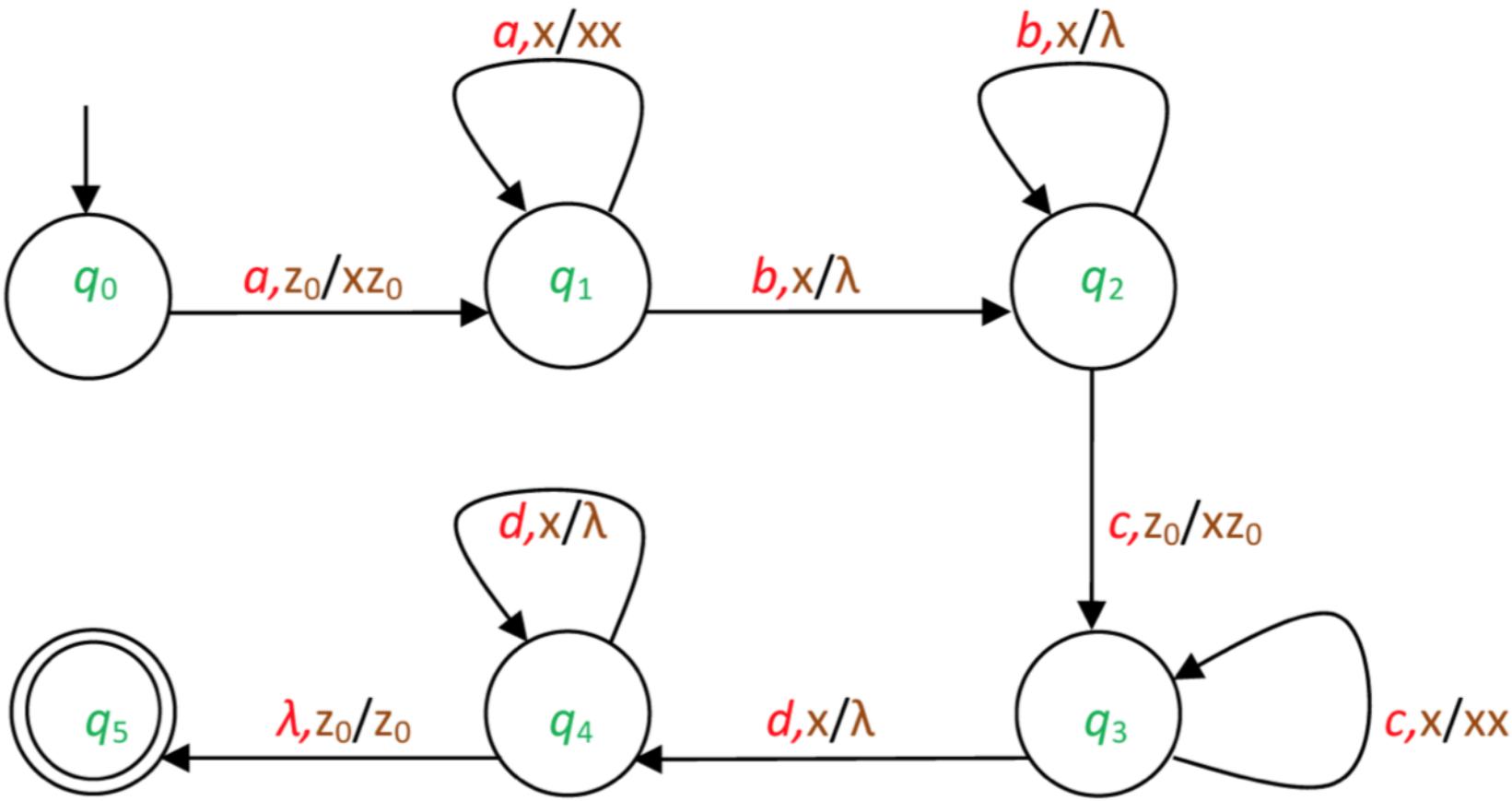
Example

This simple example shows the description of a pushdown automaton which accepts the language $L = \{a^i b^i c^j d^j \mid i, j \geq 1\}$.

$PDA = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b, c, d\}, \{x, z_0\}, q_0, z_0, \delta, \{q_5\})$,
$\delta(q_0, a, z_0) = \{(q_1, xz_0)\}$,
$\delta(q_1, a, x) = \{(q_1, xx)\}$,
$\delta(q_1, b, x) = \{(q_2, \lambda)\}$,
$\delta(q_2, b, x) = \{(q_2, \lambda)\}$,
$\delta(q_2, c, z_0) = \{(q_3, xz_0)\}$,
$\delta(q_3, c, x) = \{(q_3, xx)\}$,
$\delta(q_3, d, x) = \{(q_4, \lambda\}$,
$\delta(q_4, d, x) = \{(q_4, \lambda)\}$,
$\delta(q_4, \lambda, z_0) = \{(q_5, z_0)\}$.

Pushdown automata – graph representation

The below figure shows the graphical notation of the same pushdown automaton.



Acceptance by empty stack

There is another method for accepting words with a pushdown automaton. It is called "acceptance by empty stack". In this case, the automaton does not have any final states, and the word is accepted by the pushdown automaton if and only if it can read the whole word and the stack is empty when the end of the input word is reached. More formally:

Definition

The language accepted by automaton

$$PDA_e = (Q, T, Z, q_0, z_0, \delta)$$

by empty stack is

$$L(PDA_e) = \{p \mid p \in T^*, (q_0, p, z_0) \vdash^*_{PDA_e} (q, \lambda, \lambda), q \in Q\}.$$

Acceptance by empty stack – example

Example

This example shows the description of a pushdown automaton which accepts by empty stack the language

$L = \{a^i b^i c^j d^j \mid i, j \geq 1\}$.

$PDA_e = (\{q_0, q_1, q_2, q_3, q_4\}, \{a, b, c, d\}, \{x, z_0\}, q_0, z_0, \delta)$,

$\delta(q_0, a, z_0) = \{(q_1, xz_0)\}$,

$\delta(q_1, a, x) = \{(q_1, xx)\}$,

$\delta(q_1, b, x) = \{(q_2, \lambda)\}$,

$\delta(q_2, b, x) = \{(q_2, \lambda)\}$,

$\delta(q_2, c, z_0) = \{(q_3, xz_0)\}$,

$\delta(q_3, c, x) = \{(q_3, xx)\}$,

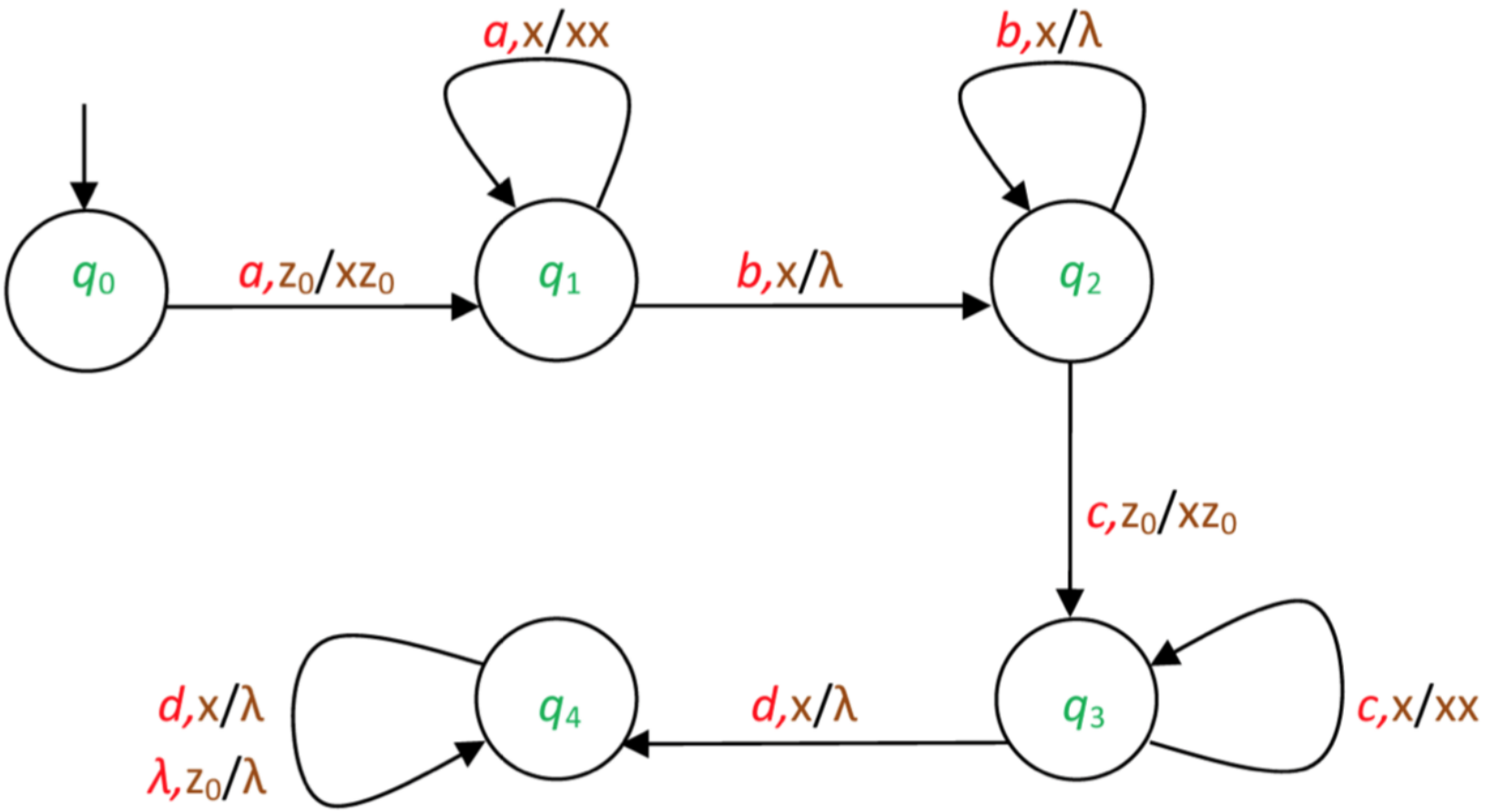
$\delta(q_3, d, x) = \{(q_4, \lambda\}$,

$\delta(q_4, d, x) = \{(q_4, \lambda)\}$,

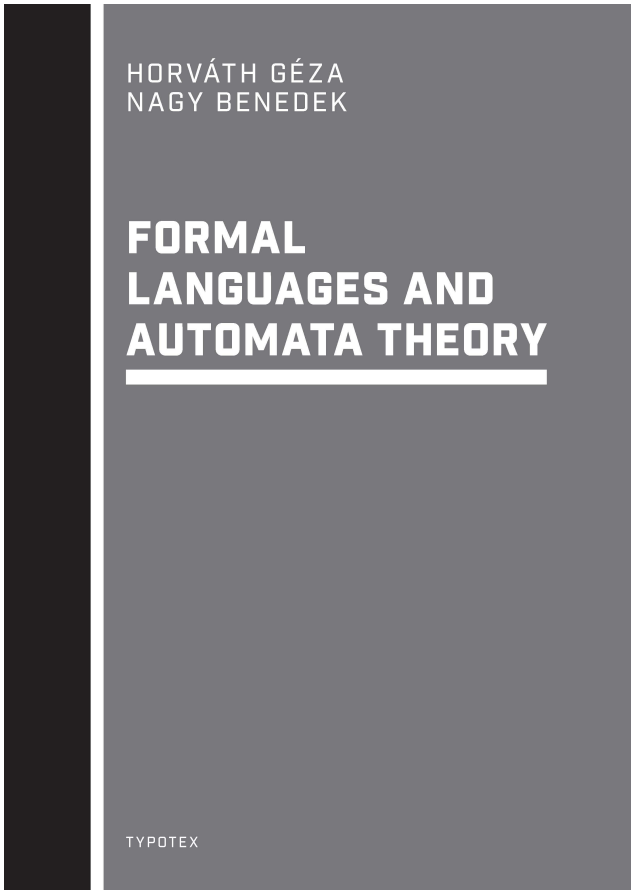
$\delta(q_4, \lambda, z_0) = \{(q_4, \lambda\}$.

Acceptance by empty stack – graph representation

The below figure shows the graphical notation of the same pushdown automaton.



References



This work was supported by the construction EFOP-3.4.3-16-2016-00021. The project was supported by the European Union, co-financed by the European Social Fund.