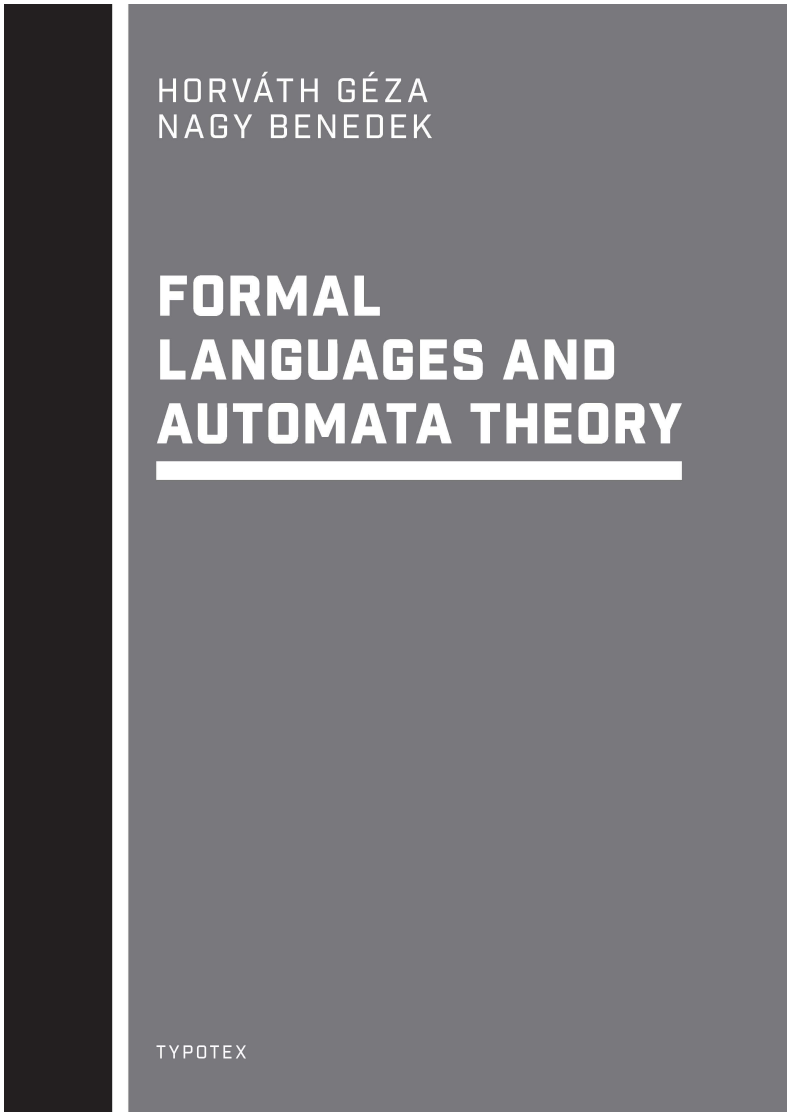
Introduction to Computer Science

Géza Horváth

First Class

References: lecture notes + book



Basic terms – alphabet

Definition

An alphabet is a finite nonempty set of symbols. We can use the union, the intersection and the relative complement set operations over the alphabet. The absolute complement operation can be used if a universe set is defined.

Example

Let the alphabet E be the English alphabet, the alphabet V contains the vowels, and the alphabet C is the set of the consonants. Then $V \cup C = E$, $V \cap C = \emptyset$ and $E \backslash V = \overline{V} = C$.

Basic terms – word

Definition

A word is a finite sequence of symbols of the alphabet. The length of the word is the number of symbols it is composed of, with repetitions. Two words are equal if they have the same length and they have the same letter in each position.

This might sound trivial, but let us see the following example:

Example

Let the alphabet $V = \{1, 2, +\}$ *and the words* $p = 1 + 1$, $q = 2$. *In this case* $1 + 1 \neq 2$, *i.e.* $p \neq q$, *because these two words have different lengths, and also different letters in the first position.*

Basic terms – concatenation

There is a special operation on words called concatenation, this is the operation of joining two words end to end.

Example

Let the alphabet E be the English alphabet. Let the word $p = railway$, and the word $q = station$ over the alphabet E. Then, the concatenation of p and q is $p \cdot q = railwaystation$. The length of p is $|p| = 7$ and the length of q is $|q| = 7$, as well.

If there is no danger of confusion we can omit the dot from between the parameters of the concatenation operation. It is obvious that the equation $|pq| = |p| + |q|$ holds for all words p and q.

Basic terms – empty word, prefix, suffix, subword

There is a special word called an empty word, whose length is 0 and denoted by λ.
The empty word is the identity element of the concatenation operation: $\lambda p = p\lambda = p$ holds for each word p over any alphabet.

Definition

The word u is a prefix of the word p if there exists a word w such that $p = uw$.

Definition

The word w is a suffix of the word p if there exists a word u such that $p = uw$.

Definition

The word v is a subword of word p if there exists words u, w such that $p = uvw$.

Basic terms – proper prefix, proper suffix, proper subword

Definition

The word u is a proper prefix of the word p, if it is a prefix of p, and the following properties hold: $u \neq \lambda$ and $u \neq p$.

Definition

The word w is a proper suffix of the word p, if it is a suffix of p, and $w \neq \lambda$, $w \neq p$.

Definition

The word v is a proper subword of the word p, if it is a subword of p, and $v \neq \lambda$, $v \neq p$.

As you can see, both the prefixes and suffixes are subwords, and both the proper prefixes and proper suffixes are proper subwords, as well.

Basic terms – prefix, suffix, subword

Example

Let the alphabet E be the English alphabet. Let the word $p = railwaystation$, and the words $u = rail$, $v = way$ and $w = station$. In this example, the word u is a prefix, the word w is a suffix and the word v is a subword of the word p. However, the word uv is also a prefix of the word p, and it is a subword of the word p, as well. The word uvw is a suffix of the word p, but it is not a proper suffix.

Basic terms – Kleene star, Kleene plus

We can use the exponentiation operation on a word in a classical way, as well.

Definition

$p^0 = \lambda$ by definition, $p^1 = p$, and $p^i = p^{i-1}p$, for each integer $i \geq 2$.

Definition

The union of p^i for each integer $i \geq 0$ is denoted by p^*, and the union of p^i for each integer $i \geq 1$ is denoted by p^+.

$$p^* = \bigcup_{i=0}^{\infty} p^i$$

$$p^+ = \bigcup_{i=1}^{\infty} p^i$$

These operations are called Kleene star and Kleene plus operations.

Basic terms – Kleene star, Kleene plus

$p^* = p^+ \cup \{\lambda\}$ holds for each word p, and if $p \neq \lambda$ then $p^+ = p^* \backslash \{\lambda\}$.

We can also use the Kleene star and Kleene plus operations on an alphabet.

Definition

For an alphabet V we denote the set of all words over the alphabet by V^*, and the set of all words, but the empty word by V^+.

Basic terms – Language, operations

A language over an alphabet is not necessarily a finite set of words, and it is usually denoted by L.

Definition

For a given alphabet V the language L over V is $L \subseteq V^*$.

We have a set again, so we can use the classical set operations, union, intersection, and the complement operation, if the operands are defined over the same alphabet.

Definition

For the absolute complement operation we use V^* for universe, so $\overline{L} = V^* \setminus L$.

Basic terms – Language, operations

We can also use the concatenation operation.

Definition

The concatenation of the languages L_1 and L_2 contains all words pq where $p \in L_1$ and $q \in L_2$.

The exponentiation operation is defined in a classical way, as well.

Definition

$L^0 = \{\lambda\}$ by definition, $L^1 = L$, and $L^i = L^{i-1} \cdot L$, for each integer $i \geq 2$.

The language L^0 contains exactly one word, the empty word. The empty language does not contain any words, $L_e = \emptyset$, and $L^0 \neq L_e$.

Basic terms – Language, operations

We can also use the Kleene star and Kleene plus closure for languages.

Definition

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

So $L^* = L^+$ if and only if $\lambda \in L$.

Basic terms – Algebraic approach

Definition

In abstract algebra, a monoid is an algebraic structure with a single associative binary operation and an identity element.

Definition

The free monoid on an alphabet V is the monoid whose elements are from V^*.

From this point of view both operations, concatenation (also called product), and the union operation (also called addition), create a free monoid on set V, because these operations are associative, and they both have an identity element.

Basic terms – Algebraic approach

From this point of view both operations, concatenation (also called product), and the union operation (also called addition), create a free monoid on set V, because these operations are associative, and they both have an identity element.

1. Associative:
   - $(L_1 \cdot L_2) \cdot L_3 = L_1 \cdot (L_2 \cdot L_3)$,
   - $(L_1 \cup L_2) \cup L_3 = L_1 \cup (L_2 \cup L_3)$,

   where $L_1, L_2, L_3 \subseteq V^*$.
2. Identity element:
   - $L^0 \cdot L_1 = L_1 \cdot L^0 = L_1$,
   - $L_e \cup L_1 = L_1 \cup L_e = L_1$,

   where $L_1 \subseteq V^*$, $L^0 = \{\lambda\}$, $L_e = \emptyset$.

The equation $L_e \cdot L_1 = L_1 \cdot L_e = L_e$ also holds for each $L_1 \subseteq V^*$.

How can we define languages?

How can we define languages?

The generative grammar is a universal method to define languages. It was introduced by Noam Chomsky in the 1950s.

Generative grammar – definition

Definition

The generative grammar (G) is the following quadruple:

$$G = (N, T, S, P)$$

where

- *N is the set of the nonterminal symbols, also called variables, (finite nonempty alphabet),*
- *T is the set of the terminal symbols or constants (finite nonempty alphabet),*
- *S is the start symbol, and*
- *P is the set of the production rules.*

The following properties hold: $N \cap T = \emptyset$ *and* $S \in N$.
Let us denote the union of the sets N and T by V ($V = N \cup T$).
Then, the form of the production rules is $V^*NV^* \rightarrow V^*$.

Generative grammar

Informally, we have two disjoint alphabets, the first alphabet contains the so called start symbol, and we also have a set of productions.
The productions have two sides, both sides are words over the joint alphabet, and the word on the left hand side must contain a nonterminal symbol.
We have a special notation for the production rules. Let the word p be the left hand side of the rule, and the word q be the right hand side of the rule, then we use the $p \to q \in P$ expression.

Generative grammar – example

Example

Let the generative grammar G *be* $G = (N, T, S, P)$, *where*
$N = \{S, A\}$,
$T = \{a, b\}$, *and*
$P = \{$
$S \to bAbS$,
$bAb \to bSab$,
$A \to \lambda$,
$S \to aa$
$\}$.

Generative grammar – derivation definition

In order to understand the operation of the generative grammar, we have to describe how we use the production rules to generate words. First of all, we should give the definition of the derivation.

Definition

Let $G = (N, T, S, P)$ be a generative grammar, and let p and q be words over the joint alphabet $V = N \cup T$. We say that the word q can be derived in one step from the word p, if p can be written in a form $p = uxw$, q can be written in a form $q = uyw$, and $x \to y \in P$. (Denoted by $p \Rightarrow q$.)

Definition

The word q can be derived from the word p, if $q = p$ or there are words $r_1, r_2, \ldots, r_n$ such that $r_1 = p$, $r_n = q$ and the word r_i can be derived in one step from the word r_{i-1}, for each $2 \leq i \leq n$. (Denoted by $p \Rightarrow^* q$.)

Generative grammar – derivation example

Example

Let the generative grammar G *be* $G = (N, T, S, P)$, *where*
$N = \{S, A\}$,
$T = \{0, 1\}$, *and*
$P = \{$
$S \to 1$,
$S \to 1A$,
$A \to AA$,
$A \to 0$,
$A \to 1$
$\}$.
Let the words p, t *and* q *be* $p = A0S0$, $t = A01A0$, *and* $q = A0110$. *In this example, the word* t *can be derived from the word* p *in one step,* $(p \Rightarrow t)$, *because* p *can be written in a form* $p = uxw$, *where* $u = A0$, $x = S$, $w = 0$, *and the word* t *can be written in a form* $t = uyw$, *where* $y = 1A$, *and* $S \to 1A \in P$.
The word q *can be derived from the word* p, $(p \Rightarrow^* q)$, *because there exist words* r_1, r_2 *and* r_3 *such that* $r_1 = p$, $r_2 = t$, $r_3 = q$ *and* $r_1 \Rightarrow r_2$ *and* $r_2 \Rightarrow r_3$.

Generative grammar – generated language

Now we have all the definitions to demonstrate how we can use the generative grammar to generate a language.

Definition

Let $G = (N, T, S, P)$ be a generative grammar. The language generated by the grammar G is

$$L(G) = \{p \mid p \in T^*,\ S \Rightarrow^* p\}.$$

The above definition claims that the language generated by the grammar G contains each word over the terminal alphabet which can be derived from the start symbol S.

Generative grammar – generated language example

Example

Let the generative grammar G *be* $G = (N, T, S, P)$, *where*
$N = \{S, A\}$,
$T = \{a, b\}$, *and*
$P = \{$
$S \to bb$,
$S \to ASA$,
$A \to a$
$\}$.
In this example, the word bb can be derived from the start symbol S *in one step, because* $S \to bb \in P$, *so the word bb is in the generated language,* $bb \in L(G)$.
The word abba can be derived from the start symbol, because $S \Rightarrow ASA$, $ASA \Rightarrow aSA$, $aSA \Rightarrow abbA$, $abbA \Rightarrow abba$, *so the word abba is also in the generated language,* $abba \in L(G)$.
The word bab can not be derived from the start symbol, so it is not in the generated language, $bab \notin L(G)$.
In this case, it is easy to determine the generated language,
$L(G) = \{a^i bba^i \mid i \geq 0\}$.

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.