Basic concepts and notations - I

An **alphabet** V is a finite nonempty set of symbols or letters.

A finite string (sequence) of letters of V is also called a **word** over V.

The set of all words over V (including the **empty word**, λ) is denoted by V^*.

The set of all nonempty words over V is denoted by V^+.

Basic concepts and notations - II

For an alphabet V, and two words $P, Q \in V^*$ the string PQ is called the **catenation** (or concatenation) of P and Q.

The catenation is an associative operation, but usually not commutative.

V^* is closed with respect to catenation; and λ is the unit element for this operation, namely, $\lambda P = P\lambda = P$ for $P \in V^*$.

Basic concepts and notations - III

For a positive integer i and for a word P ($P \in V^*$), the i-times iterated catenation (the ith power) of P is denoted by P^i.

(It means that i copies of P are catenated).

By convention, $P^0 = \lambda$.

Basic concepts and notations - IV

For $P \in V^*$, the length of P is the number of letters of P and it is denoted by $|P|$.

Obviously, $|\lambda| = 0$.

Basic concepts and notations - V

Two words over V are equal if they are the same sequence of letters, letter by letter.

The word P is a part (a **subword**) of a word Q if $Q = P_1PP_2$ holds for some words P_1 and P_2. P is said to be a proper subword of Q if at least one of P_1 and P_2 is a nonempty word and $P \neq \lambda$.

If $P_1 = \lambda$, then P is said to be a head (a **prefix**) of Q and if $P_2 = \lambda$, then P is said to be a tail (a **suffix**) of Q.

Basic concepts and notations - VI

The mirror image (or the **reversal**) of a word P is denoted by P^{-1}, and it is obtained by writing the letters of P in the reverse order.

For $P = a_1 \ldots a_n$, $a_i \in V$, $1 \leq i \leq n$, we have $P^{-1} = a_n \ldots a_1$.

Basic concepts and notations - VII

For an alphabet V, a subset L of V^* is called a **language** over V.

The empty language is denoted by $\emptyset$.

A language is finite if it consists of a finite number of words (elements), otherwise it is infinite.

Generative Grammar - Definition

A **generative grammar** (a grammar, for short) G is a quadruple (V_N, V_T, S, F), where

- V_N and V_T are finite disjoint alphabets, the alphabet of nonterminal symbols and the alphabet of terminal symbols;

- $S \in V_N$, called the initial symbol or the start symbol.

- F is a finite set of ordered pairs (P, Q) such that $P, Q \in (V_N \cup V_T)^*$ and P contains at least one symbol from V_N. The elements of F are called rewriting rules or productions and also can be denoted as $P \to Q$.

Derivation step - Definition

Let $G = (V_N, V_T, S, F)$ be a generative grammar and let $X, Y \in (V_N \cup V_T)^*$.

We say that Y can be **derived** from X in G **in one step**, denoted by

$$X \Longrightarrow_G Y,$$

if $X = P_1 P P_2$, $Y = P_1 Q P_2$, $P_1, P_2 \in (V_N \cup V_T)^*$ and $P \to Q \in F$ holds.

Derivation - Definition

Let $G = (V_N, V_T, S, F)$ be a generative grammar and let $X, Y \in (V_N \cup V_T)^*$.

We say that Y can be **derived** from X in G, denoted by

$$X \Longrightarrow_G^* Y,$$

if either $X = Y$ or there is some word $Z \in (V_N \cup V_T)^*$ such that $X \Longrightarrow_G^* Z$ and $Z \Longrightarrow_G Y$ hold.

$\Longrightarrow^*$ denotes the reflexive transitive closure of $\Longrightarrow$.

The transitive closure of $\Longrightarrow$ is denoted by $\Longrightarrow^+$.

The generated language - Definition

Let $G = (V_N, V_T, S, F)$ be a generative grammar. The **language** $L(G)$ **generated by** G is defined by

$$L(G) = \{P | S \Longrightarrow_G^* P, P \in V_T^*\}.$$

That is, the language generated by G consists of those words that are in V_T^* and can be derived from the initial symbol S.

Equivalent Grammars and Languages

Two grammars are called **equivalent** if they generate the same language.

Two languages are called **weakly equivalent** if they differ in the empty word at most.

The Chomsky Hierarchy

A generative grammar $G = (V_N, V_T, S, F)$ is said to be of **type** i, for $i = 0, 1, 2, 3$ if it satisfies the corresponding conditions in the list:

- $i = 0$: No restriction.
- $i = 1$: For every rewriting rule $P \to Q \in F$ it holds that $P = P_1AP_2$, $Q = P_1WP_2$, where $A \in V_N$, $P_1, P_2, W \in (V_N \cup V_T)^*$, $W \neq \lambda$, except possibly $S \to \lambda$. In this latter case S does not occur at the right-hand side of any rule in F.
- $i = 2$: For every rule $P \to Q \in F$ it holds that $P \in V_N$.
- $i = 3$: Every rule in F is either of the form $A \to ZB$ or $A \to Z$, where $A, B \in V_N$ and $Z \in V_T^*$.

The Chomsky Hierarchy - continued

A language L is said to be of type i if it is generated by a type i grammar. The **class** (or family) of type i languages is denoted by $\mathcal{L}_i$, $i = 0, 1, 2, 3$.

Type 0 grammar are called **phrase-structure** grammars, type 1 grammars are called **context-sensitive** grammars, type 2 grammars are called **context-free** grammars. Type 3 grammars are called **regular** grammars.

The corresponding language classes are called the class of **recursively enumerable**, the class of **context-sensitive**, the class of **context-free**, and the class of **regular** languages.

The Chomsky Hierarchy - continued

It is obvious, that $\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_0$ and $\mathcal{L}_1 \subseteq \mathcal{L}_0$.

Later, we will see that the following hierarchy holds (the Chomsky hierarchy):

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0.$$

Remark: Observe that the relation between language classes $\mathcal{L}_2$ and $\mathcal{L}_1$ is not obvious.

Operations on Languages - I

Definition:

- $L_1 \cup L_2 = \{P \mid P \in L_1 \text{ or } P \in L_2\}$ - the **union** of languages L_1 and L_2;
- $L_1 \cap L_2 = \{P \mid P \in L_1 \text{ and } P \in L_2\}$ - the **intersection** of languages L_1 and L_2;
- $L_1 - L_2 = \{P \mid P \in L_1 \text{ and } P \notin L_2\}$ - the **difference** of languages L_1 and L_2;
  - Sometimes it is denoted by $P_1 \setminus P_2$;
- The **complement** of $L \subseteq V^*$ with respect to V is $\bar{L} = V^* - L$.

Operations on Languages - II

Definition

- $L_1L_2 = \{P_1P_2 \mid P_1 \in L_1, P_2 \in L_2\}$ - the **catenation** of languages L_1 and L_2;
- L^i the ith **iteration** of L with respect to catenation, $i \geq 1$, and $L^0 = \{\lambda\}$.
- We have the identities $\emptyset L = L\emptyset$ and $\{\lambda\}L = L\{\lambda\}$.
- The catenation **closure** of L is $L^* = \bigcup_{i \geq 0} L^i$.
- $L^+ = \bigcup_{i \geq 1} L^i$

Obviously, $L^+ = L^*$ if $\lambda \in L$ and $L^+ = L^* - \{\lambda\}$ if $\lambda \notin L$.

Operations on Languages - III

Definition:

$$L^{-1} = \{P^{-1} | P \in L\}$$

- the mirror image (**reversal**) of language L.

Property: $(L^{-1})^{-1} = L$ and $(L^{-1})^i = (L^i)^{-1}$, $i \geq 0$.

Definition:

$$HEAD(L) = \{P | P \in V^*, PQ \in L \, \text{for some} \, Q \in V^*\}$$

- the **head** of language L.

By definition, $L \subseteq HEAD(L)$ for any $L \in V^*$.

Operations on Languages - IV

Definition:

Let V_1 and V_2 alphabets. The mapping $h : V_1^* \to V_2^*$ is called a **homomorphism** if $h(PQ) = h(P)h(Q)$ for all $P, Q \in V_1^*$.

This implies that $h(\lambda) = \lambda$.

Every homomorphism is completely defined whenever it is defined for the letters of V_1.

For every $P = a_1a_2 \dots a_n$, where $a_i \in V_1, 1 \leq i \leq n$, it holds that $h(P) = h(a_1)h(a_2)\dots h(a_n)$. Thus, it is enough to give h for the elements of V_1 and then extend it to V_1^*.

Operations on Languages - V

Definition: A homomorphism is λ-**free** if $h(P) \neq \lambda$ for $P \in V_1^+$.

Definition: Let $h : V_1^* \to V_2^*$ be a homomorphism. The **homomorphic image** of a language $L \subseteq V_1^*$ is defined as

$$h(L) = \{W \in V_2^* \mid W = h(P), P \in L\}$$

.

Operations on Languages - VI

Definition: A homomorphism is called **isomorphism** if for all P and Q in V_1^* $h(P) = h(Q)$ implies $P = Q$.

An example for the isomorphism is the so-called binary coded decimal representation of integers:

$$V_1 = \{0, 1, 2, \ldots, 9\}, V_2 = \{0, 1\},$$
$$h(0) = 0000, \;\; h(1) = 0001, \;\; \ldots, h(9) = 1001$$

Normal Form Theorem

For every grammar $G = (V_N, V_T, S, F)$ we can give an equivalent grammar $G' = (V'_N, V_T, S, F')$ of the same type such that no terminal letter occurs on the left-hand side of any production in F'.

Proof idea

- For languages of type 3 and 2, there is nothing to prove.

- Let $G = (V_N, V_T, S, F)$ be of type 0 or 1.

  We define $V_N' = V_N \cup \bar{V}_N$, where $\bar{V}_N = \{\bar{a} \mid a \in V_T\}$. Let F' be obtained from F by replacing any occurrence of $a \in V_T$ by $\bar{a}$ on both sides of the rules and adding rules $\bar{a} \to a$ to F for any $a \in V_T$.

Proof idea - continued

(1) $L(G) \subseteq L(G')$

For $P = a_{i_1} \ldots a_{i_n} \in L(G)$, $a_{i_j} \in V_T$, we can derive $\bar{a}_{i_1} \ldots \bar{a}_{i_n}$ in G'. Then using rules $\bar{a}_{i_j} \to a_{i_j}$, we obtain $P \in L(G')$.

For $P = \lambda$ it is clear that if $\lambda \in L(G)$, then $\lambda \in L(G')$.

Proof idea - continued

(2) $L(G') \subseteq L(G)$

We define the homomorphism $h(\bar{a}) = a$ for $\bar{a} \in \bar{V}_N$ and $h(x) = x$ for $x \in (V_N \cup V_T)$.

If $P \Longrightarrow_{G'} Q$ holds, then $h(P) \Longrightarrow^*_G h(Q)$ holds as well. If Q is derived from P by a rule $\bar{a} \to a$, then $h(P) = h(Q)$. Otherwise, $P \Longrightarrow_{G'} Q$ involves the application of a rule from F, thus $h(P) \Longrightarrow_G h(Q)$ holds. Thus, $P \Longrightarrow^*_{G'} Q$ implies $h(P) \Longrightarrow^*_G h(Q)$. This means that for $S \Longrightarrow^*_{G'} P$, where $P \in V_T^*$ it holds that $S = h(S) \Longrightarrow^*_G h(P) = P$.

Closure Properties of Language Classes

The set of union, catenation, and the closure of iteration are called **regular operations**.

Theorem:

Any of the language classes $\mathcal{L}_i$, $i = 0, 1, 2, 3$ is closed under the regular operations.

Proof idea:

Let L and L' be two languages of type i, $i = 0, 1, 2, 3$. Suppose that they are generated by type i grammars $G = (V_N, V_T, S, F)$ and $G' = (V'_N, V'_T, S', F')$, respectively. Without the loss of generality we may assume that G and G' are in normal form (previous) and $V_N \cap V'_N = \emptyset$.

Proof idea - continued

Union:

(1) For $i = 0, 2, 3$ we consider $S_0 \notin (V_N \cup V'_N)$ and

$$G_u = (V_N \cup V'_N \cup \{S_0\}, V_T \cup V'_T, S_0, F \cup F' \cup \{S_0 \to S, S_0 \to S'\}).$$

(2) It is obvious that G_u is of the same type as G and G'.

(3) It is also obvious that $L(G) \cup L(G') \subseteq L(G_u)$.

(4) $L(G_u) \subseteq L(G) \cup L(G')$ also holds since V_N and V'_N are disjoint sets and the rules $S_0 \to S$, $S_0 \to S'$ guarantee that rules of F and F' cannot be mixed during any derivation of any terminal word of $L(G_u)$.

Proof idea - continued

For $i = 1$ and $\lambda \notin L \cup L'$ we construct G_u as above.

For $i = 1$ and $\lambda \in L \cup L'$ we first take $L_1 = L - \{\lambda\}$ and $L_2 = L' - \{\lambda\}$ and construct G_u as above. Then we introduce a new startsymbol S_1 and add productions $S_1 \to S_0$ and $S_1 \to \lambda$.

Proof idea - continued

Catenation: First consider $i = 0, 2$. Let $S_0 \notin (V_N \cup V'_N)$ and

$$G_c = (V_N \cup V'_N \cup \{S_0\}, V_T \cup V'_T, S_0, F \cup F' \cup \{S_0 \to SS'\}).$$

(2) It is obvious that G_c is of the same type as G and G'.

(3) It is also obvious that $L(G)L(G') \subseteq L(G_c)$.

(4) We show that $L(G_c) \subseteq L(G)L(G')$. Consider a derivation in G_c

$$S_0 \Longrightarrow P_1 \Longrightarrow P_2 \Longrightarrow \ldots \Longrightarrow P_m = P$$

where $P \in (V_T \cup V'_T)^*$. It can be shown by induction on j that $P_j = Q_j Q'_j$ such that $S \Longrightarrow^*_G Q_j$ and $S' \Longrightarrow^*_{G'} Q'_j$ holds. For $j = 1$ it is trivial. Suppose that it is true for j. Then the statement holds for P_{j+1} as well, since V_N and V'_N are disjoint sets and to obtain P_{j+1}, either Q_j or Q'_j is rewritten. This implies that each word of $L(G_c)$ is in $L(G)L(G')$.

Proof idea - continued

Catenation - continued:

(5) Let $i = 1$.

(a) If $\lambda \notin L(G)$ and $\lambda \notin L(G')$, then we consruct G_c as above.

(b) If $\lambda \in L(G) \cup L(G')$, then we first take $L_1 = L - \{\lambda\}$ and $L_2 = L' - \{\lambda\}$ and construct G_c as above. Language $L(G)L(G')$ will be identical to one of the following languages: $L(G_1)L(G_2) \cup L(G_2)$, $L(G_1)L(G_2) \cup L(G_1)$, $L(G_1)L(G_2) \cup L(G_2) \cup L(G_1) \cup \{\lambda\}$, depending on whether $\lambda \in L(G)$ or in $L(G')$, or conversely, or in both of them. The statement $L(G)L(G') \in \mathcal{L}_1$ follows from the fact that $L(G_1)L(G_2) \in \mathcal{L}_1$ and $\mathcal{L}_1$ is closed under the union.

Proof idea - continued

Catenation - continued:

Let $i = 3$.

From F we construct F'' by replacing each rule of the form $A \to P$, where $A \in V_N$ and $P \in V_T^*$ by $A \to PS'$. Then we consider

$$G_c = (V_N \cup V_N', V_T \cup V_T', S, F'' \cup F').$$

It is obvious that G_c is of type 3 and $L(G)L(G') \subseteq L(G_c)$. The reverse statement also holds. Any terminating derivation in G_c is of the form $S \Longrightarrow_{G_c}^* QS' \Longrightarrow_{G_c}^* QQ'$, where to obtain Q we use elements of F and to obtain Q' elements of F'. Thus $Q \in L(G)$ and $Q' \in L(G')$.

Proof idea - continued

Iteration Closure:

(1) Let $i = 2$ and $S_0 \notin V_N$. Then

$$G_* = (V_N \cup \{S_0\}, V_T, S_0, F \cup \{S_0 \to \lambda, S_0 \to SS_0\})$$

generates L^*.

(2) Let $i = 3$ and we define F_* such that $A \to PS$ is in F_* for any rule $A \to P \in F$, where $P \in V_T^*$. Then

$$G_* = (V_N \cup \{S_0\}, V_T, S_0, F_* \cup F \cup \{S_0 \to \lambda, S_0 \to S\})$$

generates L^*.

(3) Let $i = 0, 1$ and $\lambda \notin L$. Suppose that $S_0, S_1 \notin V_N$. Then the grammar

$G_* = (V_N \cup \{S_0, S_1\}, V_T, S_0, F \cup \{S_0 \to \lambda, S_0 \to S, S_0 \to S_1S\} \cup$
$\{S_1a \to S_1Sa \mid a \in V_T\} \cup \{S_1a \to Sa \mid a \in V_T\})$

generates L^*. Remember: we assumed that the initial grammars are in a normal form, where the left-hand side of a rule contains only nonterminal symbols.

Proof idea - continued

$G_* = (V_N \cup \{S_0, S_1\}, V_T, S_0, F \cup \{S_0 \to \lambda, S_0 \to S, S_0 \to S_1 S\} \cup \{S_1 a \to S_1 S a \mid a \in V_T\} \cup \{S_1 a \to S a \mid a \in V_T\})$

We show that $L(G_*) \subseteq L^*$.

Consider

$$S_0 \Longrightarrow_{G_*} P_1 \Longrightarrow_{G_*} P_2 \Longrightarrow_{G_*} \ldots \Longrightarrow_{G_*} P_m = P,$$

where $P \in V_T^*$. If in the first step $S_0 \to \lambda$ is used then $m = 1$ and $P_m = \lambda \in L^*$. If $P_1 = S$ then $P \in L^*$ holds, otherwise we have $P_1 = S_1 S$ and for every j, $1 \leq j \leq m$ it can be shown by induction that P_j is either of the two forms

(a) $S_1 Q_1 \ldots Q_k$, $k \geq 1$, where $S \Longrightarrow_G^* Q_l$, $l = 1, \ldots, k$ and each of the words $Q_2, \ldots, Q_l$ begins with a terminal symbol; or

(b) $Q_1 \ldots Q_k$, $k \geq 0$, where $S \Longrightarrow_G^* Q_l$, $l = 1, \ldots, k$ and each of the words $Q_2, \ldots, Q_l$ begins with a terminal symbol.

The form of $P_1 = S_1 S$ corresponds to form (a). Examining the rules in G_* we obtain that P_{j+1} is one of the forms (a) or (b) if P_j has this property. Thus, $L(G_*) \subseteq L^*$.

Proof idea - continued

If $i = 0, 1$ and $\lambda \in L$, then we first construct a grammar G_1 with $L(G_1) = L - \{\lambda\}$.

(a) For $i = 1$ this is easy, we omit $S \to \lambda$.

(b) For $i = 0$ we replace each rule $P \to \lambda$ with $PX \to X$ and $XP \to P$ for all $X \in (V_N \cup V_T)$. The type of G_1 is of the same as that of G and it also holds that $(L - \{\lambda\})^* = L^*$.

Corollary

If the language L is of type i, $i = 0, 1, 2, 3$, then L^+ is of type i.

Further Properties

(1) Each of the language families $i = 0, 1, 2, 3$ is closed under the mirror image.

(2) Each of the language families $i = 0, 2$ is closed under the homomorphism and $\mathcal{L}_1$ is closed under λ-free homomorphism.

3) Every finite language is in $\mathcal{L}_3$.

Normal Forms of Context-Free Grammars

Theorem

For every context-free grammar G we can construct an equivalent context-free grammar G' such that the right-hand sides of the rules of G' are different from λ except when $\lambda \in L(G)$ in which case $S' \to \lambda$ is the only rule with λ on the right-hand side but then S' does not occur on the right-hand sides of the rules.

Proof idea:

Let $G = (V_N, V_T, S, F)$ be a context-free grammar. We define the set of nonternimals $E_i \subseteq V_N$ as follows:

$$E_1 = \{X \mid X \to \lambda \in F\}$$

$$E_{i+1} = E_i \cup \{X \mid X \to P \in F \text{ for some } P \in E_i^*\},\ i \geq 1$$

.

It is obvious that E_i, $i = 1, 2, \ldots$, forms a nondecreasing sequence for inclusion and it has an upper limit (V_N), so there exists an index k such that $E_k = E_j$, for $j \geq k$. Let us denote E_k by E.

It can be seen immediately that $X \Longrightarrow_G^* \lambda$ if and only if $X \in E$ holds. This implies that $\lambda \in L(G)$ if and only if $S \in E$ holds.

Proof idea - continued:

We construct now F' as follows:

$X \to P' \in F'$ if $P' \neq \lambda$ and there is a word $P \in (V_N \cup V_T)^*$ such that $X \to P \in F$ and P' is obtained from P by erasing zero or more occurrences of one or more nonterminals in E occurring in P.

Then, it can be seen that $L(G') \subseteq L(G) - \{\lambda\}$ since each rule $X \to P'$ corresponds to a derivation consisting of the application of $X \to P$ combined with (zero or more) derivations $Z \Longrightarrow_G^* \lambda$ where $Z \in E$ and Z appears in P.

Conversely, if $S \Longrightarrow_G^* P$ and $P \neq \lambda$, then there is also a derivation $S \Longrightarrow_{G'}^* P$ since the need for the application of $X \to \lambda$-like rules can be avoided by using the appropriate rule from F'.

This implies that $L(G') = L(G) - \{\lambda\}$.

If $\lambda \in L(G)$ then we take $G'' = (V_N \cup \{S_1\}, V_T, S_1, F' \cup \{S_1 \to \lambda, S_1 \to S\})$ which generates $L(G)$.

λ-free Grammar

Definition

A grammar G is called λ-free if none of its rules has λ on the right-hand side.

Theorem

For every context-free (regular) grammar G we can construct a context-free (regular) grammar G' such that G' is λ-free and $L(G') = L(G) - \{\lambda\}$.

The statement is a direct consequence of the previous theorem.

An other normal form

Theorem
For every context-free grammar $G = (V_N, V_T, S, F)$ we can construct an equivalent context-free grammar $G' = (V'_N, V_T, S, F')$ such that each of its rules has either of the two forms

1. $X \to W$, where $X \in V'_N$, $W \in V'^*_N$,

2. $X \to a$, where $X \in V'_N$, $a \in V_T$.

Proof idea:

For every $a \in V_T$ we define a new non-terminal symbol $\overline{a} \in V_N'$. We construct F' as follows: $X \to P' \in F'$ if one of the following conditions holds:

1. $X \to P' \in F$ and $P' \in V_T \cup V_N^*$,
2. $X = \overline{a}, P' = a$, for some $a \in V_T$, or
3. $X \to P \in F$ and we obtain P' from P by replacing every terminal symbol a in P with the corresponding non-terminal symbol $\overline{a}$.

It can be easily seen that $L(G') = L(G)$.

Chomsky normal form of context-free grammars

Definition

A context-free grammar $G = (V_N, V_T, S, F)$ is said to be in Chomsky normal form if each of its rules has either of the two forms

1. $X \to a$, where $X \in V_N$, $a \in V_T$,

2. $X \to YZ$, where $X, Y, Z \in V_N$.

Chomsky normal form - continued:

Theorem

For every λ-free context-free grammar $G = (V_N, V_T, S, F)$ we can construct an equivalent grammar $G' = (V'_N, V_T, S, F')$ in Chomsky normal form.

Proof idea:

1) Without the loss of generality we may assume that terminal symbols appear in F only in rules of the form $X \to a$, where $a \in V_T$ (by a previous theorem).

2) All the other rules have the form $X \to P$, where $P \in V_N^*$.

3) We replace each rule of the form

$$X \to Y_1 Y_2 \dots Y_k, \ k \geq 3$$

with

$$\begin{aligned} X &\to Y_1 Z_1, \\ Z_1 &\to Y_2 Z_2, \\ &\dots, \\ Z_{k-2} &\to Y_{k-1} Y_k, \end{aligned}$$

where $Z_1, \dots, Z_{k-2}$ are new nonterminal symbols.

Proof idea - continued:

Thus we obtain a grammar $G_1 = (V'_N, V_T, S, F_1)$ where F_1 contains the following three kinds of rules:

1. $X \to a,\ X \in V'_N,\ a \in V_T,$
2. $X \to Y,\ X, Y \in V'_N,$
3. $X \to YZ,\ X, Y, Z \in V'_N.$

The nonterminal alphabet V'_N consists of elements of V_N and the new nonterminals separately introduced for each rule of F that has been eliminated.

Proof idea - continued:

Rules of the form $X \to Y$, where X and Y are nonterminals are called chain rules and they have to be eliminated.

We define for each $X \in V_N'$

$$Ch_1(X) = \{X\}$$

and

$$Ch_{i+1}(X) = Ch_i(X) \cup \{Y \mid Y \to Z \in F_1 \text{ for some } Z \in Ch_i(X)\},\ i = 1, 2, \ldots.$$

It is obvious that there is an integer k such that $Ch_k(X) = Ch_l(X)$ for any $l \geq k$. Let us denote $Ch_k(X) = Ch(X)$.

Then $Y \Longrightarrow^* X$ holds if and only if $Y \in Ch(X)$.

Proof idea - continued:

We define F' as follows:

1. $X \to a \in F'$ if and only if there is some $A \in V'_N$ such that $X \in Ch(A)$ and $A \to a \in F_1$,

2. $X \to YZ$ if and only if there is some $A \in V'_N$ such that $X \in Ch(A)$ and $A \to YZ \in F_1$.

It can be seen that $X \to a \in F'$ if and only if $X \Longrightarrow^*_{G_1} a$ and $X \to YZ \in F'$ if and only if $X \Longrightarrow^*_{G_1} A \Longrightarrow^*_{G_1} YZ$ for some A.

Thus if a word can be generated in G_1 it can be generated in G' as well and vice versa.

Consequences:

Theorem

For every context-free grammar G it is decidable whether or not an arbitrary word P belongs to the language $L(G)$.

Proof idea:

It is enough to consider $P \neq \lambda$, furthermore, we may suppose that G is in Chomsky normal form. If P can be derived in G in k steps then $|P| \leq k + 1$. More precisely, if P can be derived in G then it can be derived in $k = 2|P| - 1$ steps. Since the number of words that can be derived in $1, 2, \ldots k$ is finite, we can decide whether P appears in this set.

Corollary

For every context-free grammar G and every finite language L the problems whether $L \subseteq L(G)$ and $L \cap L(G) = \emptyset$ are decidable.

Tree

Let N be a finite nonempty set whose elements are called nodes.

The set of edges E is a set of ordered pairs of nodes, $E \subseteq N \times N$, and for an edge $e = (n_1, n_2) \in E$, $s(e) = n_1$ and $t(e) = n_2$ are called its starting node and terminating node, respectively.

A sequence of edges $e_1, e_2, \ldots, e_k$ is called a directed path of length k from $s(e_1)$ to $t(e_k)$ if $s(e_{i+1}) = t(e_i)$ for $i = 1, 2, \ldots, k-1$.

Tree - continued

The ordered pair (N, E) is called a directed tree if there is a node $r \in N$ such that

1. None of the terminating nodes of the edges in E is identical with r.
2. There is a unique directed path from r to every other node in N.

The node r is called the root of the tree.

Each node of a tree is the root of a subtree.

Extremal nodes having no outgoing edges are called leaves.

Derivation Tree

We may associate trees to derivations in context-free grammars.

The root of the derivation tree is labelled by S.

Every other node is labelled by an element of $V_N \cup V_T$.

A node labelled by a terminal must be a leaf.

If a node is labelled by X and its child nodes from left to right are labelled by $Y_1, \ldots, Y_k$, then $X \to Y_1 \ldots Y_k \in F$.

A derivation tree does not always specify the order of the application of the rules. Two derivations are essentially the same if they differ only in the order of the application of the rules, i.e., if they have the same derivation tree.

There is a unique leftmost derivation for every derivation tree of a context-free grammar.

Emptyness of a $\overset{CF}{\underline{\text{context-free}}}$ language

Theorem

For any context-free grammar it is decidable whether it generates the empty language $\emptyset$.

Proof idea

Let $G = (V_N, V_T, S, F)$ be a context-free grammar. Without the loss of generality we may assume that G is λ-free.

Let n be the number of elements in V_N.

Suppose that there exists a derivation $S \Longrightarrow_G^* P$ in G where $P \in V_T^*$.

Proof idea - continued

Consider the derivation tree associated with this derivation.

If the length of the longest path in this derivation is longer than n, then we have a $P' \in L(G)$ having a derivation tree whose longest path is less than n.

This is obvious, since if the length of the path is longer than n, then at least one nonterminal in the path must occur at least twice on the path.

Consider two nodes with the same label and replace the subtree belonging to the former one by the latter one. Then we still obtain a terminal word.

By repeating this procedure as many times as necessary, we can cut back the length of any path which is longer than n.

Thus, if the language $L(G)$ is not empty then it must contain a word with a derivation tree whose longest path is not longer than n.

Algorithm

1) Take the root S as the only element of the set (forest) of derivation trees T.

2) If there is a derivation tree in the set T which has only terminal symbols as labels of its leaves, then $L(G)$ is not empty and the procedure is finished.

3) Select each tree from T and extend it in every possible way by the application of a rule to one of its leaves labelled by a nonterminal. Include each extended tree in the set T unless its longest path is longer than n or it has already been included in T.

4) If during the last execution step 3 no new tree has been added to T then $L(G)$ is empty and the procedure is finished, otherwise it will be repeated from step 2.

Useless nonterminals

Definition

1) A nonterminal of a context-free grammar is called inactive or dead if no terminal word can be derived from it; otherwise it is called active.

2) A nonterminal of a context-free grammar is called unreachable if it does not occur in any word derivable from the initial symbol of the grammar; otherwise it is called reachable.

A nonterminal is useless if it is inactive or unreachable or both. A nonterminal is useful if it is reachable and active.

Useless nonterminals

It is decidable whether or not a nonterminal A is inactive. For the context-free grammar $G = (V_N, V_T, S, F)$ we consider the context-free grammar $G_A = (V_N, V_T, A, F)$. If $L(G_A)$ is empty, then A is inactive.

It is decidable whether or not a nonterminal A is unreachable. Let us consider $G = (V_N, V_T, S, F)$ and remove from F all rules which have A on their left-hand side and let us denote the remaining rule set by F_1. Then construct

$$G_A^\lambda = ((V_N - \{A\} \cup V_T, \{A\}, S, F_1 \cup \{X \to \lambda \mid X \in (V_N - \{A\} \cup V_T)\}).$$

If A is unreachable, then $L(G_A^\lambda) = \{\lambda\}$, otherwise is must contain a word with at least one occurrence of A. Then we construct a context-free grammar G' such that $L(G') = L(G_A^\lambda) - \{\lambda\}$. Since we can decide whether $L(G')$ is empty, we can decide the unreachability of A as well.

Redundancy

Definition

A context-free grammar is nonredundant or reduced if each of its nonterminals is active and reachable.

Redundancy

Theorem

For every context-free grammar we can find an equivalent nonredundant context-free grammar.

Inactive and unreachable nonterminals, together by the rules in which they occur can be eliminated from the grammar without changing the generated language, as we have seen before. So we can construct an equivalent nonredundant grammar.

Idea of a direct proof

Let $G = (V_N, V_T, S, F)$ be a context-free grammar. We construct

$$A_1 = \{X \mid X \to P \in F, P \in V_T^*\},$$
$$A_{i+1} = A_i \cup \{X \mid X \to W \in F, W \in (V_T \cup A_i)^*\},\ i = 1, 2, \ldots$$

It is clear that A_i, $i = 1, 2, \ldots$, forms a nondecreasing sequence for inclusion, and it has a trivial upper limit, V_N. Thus, there exists an integer k such that $A_k = A_l$ for any $l \geq k$. Then, A_k is the set of active nonterminals of G.

Idea of a direct proof - continued

Similarly, we define

$$R_1 = \{S\},$$
$$R_{i+1} = R_i \cup \{Y \mid X \to UYW \in F, X \in R_i, U, W \in (V_N \cup V_T)^*\},\ i = 1, 2, \ldots$$

It is clear that R_i, $i = 1, 2, \ldots$, forms a nondecreasing sequence for inclusion, with the trivial upper limit V_N. Thus, there exists an integer m such that $R_m = R_l$ for any $l \geq m$. Then, R_m is the set of reachable nonterminals of G.

After computing A_k and R_m, we eliminate all nonterminals which are not in $A_k \cap R_m$ together with all rules in which they occur. The same process should be repeated with the resulting grammar until we get a nonredundant grammar.

Bar-Hillel or pumping lemma

Theorem

For every context-free language L we can give two natural numbers, p and q, such that each word in L that is longer than p has the form $UXWYZ$, where $|XWY| \leq q$, $XY \neq \lambda$, and UX^iWY^iZ is in L for all $i \geq 0$ $(U, X, W, Y, Z \in V_T^*)$.

Proof idea

We consider only λ-free languages L and Chomsky normal form grammars $G = (V_N, V_T, S, F)$ to generate L.

Let n be the number of nonterminals in G and let $p = 2^n$ and $q = 2^{n+1}$.

If $|P| > p$ for some $P \in L$, then the longest paths in the derivation tree of P should be longer than n. (The tree is binary, i.e., from any node at most two nodes can be derived and the number of leaves in a binary tree with longest path k is at most 2^k.)

Now we consider the final $n + 1$ edges of the longest path.

Then there must be a nonterminal A in this subpath which occurs at least twice.

Proof idea - continued

The subtree belonging to the first occurrence of A on this subpath represents a derivation $A \Longrightarrow^* Q$ for some $Q \in V_T^*$. Similarly, we have $A \Longrightarrow^* W$, $W \in V_T^*$, from the second occurrence of A.

Furthermore, there are two words $X, Y \in V_T^*$ such that $Q = XWY$, and there is a derivation $A \Longrightarrow^* XAY$ involving the two As.

At the same time we have the derivation

$$S \Longrightarrow^* UAZ \Longrightarrow^* UXAYZ \Longrightarrow^* UXWYZ = UQZ.$$

The positions of the given occurrences of A imply that $|Q| \leq 2^{n+1}$. Furthermore, the derivation $A \Longrightarrow^* XAY$ implies that at least one rule of the form $B \to CD$ should be applied, thus $|XY| \neq \lambda$ holds.

Thus we can see that $S \Longrightarrow^* UWZ$ implies $S \Longrightarrow^* UX^iW^iYZ$ for $i \geq 1$.

Consequence

There are non-context-free phrase structure languages.

See, $L = \{a^n b^n c^n | n \geq 1\}$.

Consequence

Theorem

It is decidable whether a context-free grammar generates an infinite language.

Proof idea:

We consider only λ-free grammars $G = (V_N, V_T, S, F)$. We show that $L(G)$ is infinite if and only if it contains a word P such that $p < |P| \leq p + q$ holds.

1) If G satisfies this property, then by the pumping lemma we have the statement.

2) We show that the converse statement holds. If $L(G)$ is infinite then it must contain a word P with $p < |P|$, so we show that it contains a one with $p < |P| \leq p+q$.

Suppose the contrary, that is, for all P with $p < |P|$ it holds that $p + q < |P|$.

But, if $p < |P|$, then P has the form $UXWYZ$ with $UWZ \in L(G)$ and $|UWZ| < |P|$ since $XY \neq \lambda$.

If $p < |UWZ|$ then we repeat the argument until we obtain $P' = U'X'W'Y'Z'$ with $p < |P'|$ and $|U'W'Z'| \leq p$.

Using condition $|X'W'Y'| \leq q$, we obtain $p < |U'X'W'Y'Z'| \leq p + q$ which contradicts to the assumption.

(The upper bound $p + q$ makes possible to search appropriate derivation trees.)

Formal Languages - 4. Lecture

Erzsébet Csuhaj-Varjú

Gábor Kolonits

Department of Algorithms and Their Applications
Faculty of Informatics
Eötvös Loránd University
H-1117 Budapest
Pázmány Péter sétány 1/c
Hungary
E-mail: {csuhaj,kolomax}@inf.elte.hu

Linear Grammars and Regular Languages

Definition

A context-free grammar $G = (V_N, V_T, S, F)$ is called linear if each of its rules has either of the two forms

1. $A \to P$, where $A \in V_N$ and $P \in V_T^*$,

2. $A \to Q_1 B Q_2$, where $A, B \in V_N$ and $Q_1, Q_2 \in V_T^*$.

G is called left-linear (right-linear) if $Q_1 (Q_2)$ is the empty word, λ.

Linear Grammars and Languages

Remark: Right-linear grammars are regular (type 3) grammars.

Definition

A language is called linear, or left-linear, or right-linear, if it is generated by a linear, or left-linear, or right-linear grammar, respectively.

Left-Linear versus Regular Languages

Theorem Every left-linear grammar generates a regular (type 3) language.

Proof idea

Let $G = (V_N, V_T, S, F)$ be a left-linear grammar and let $V_N = \{S, A_1, \ldots, A_n\}$, $n \geq 1$. Without the loss of generality we may assume that S does not appear at the right-hand side of any rule.

We construct a right-linear grammar $G' = (V_N, V_T, S, F')$ such that $L(G) = L(G')$ holds.

Let us define

1. $S \rightarrow P \in F'$ if and only if $S \rightarrow P \in F$ and $P \in V_T^*$,
2. $S \rightarrow PA_k \in F'$ if and only if $A_k \rightarrow P \in F$, $k \in \{1, \ldots, n\}$, $P \in V_T^*$,
3. $A_j \rightarrow PA_k \in F'$ if and only if $A_k \rightarrow A_jP \in F$, $j, k \in \{1, \ldots, n\}$, $P \in V_T^*$,
4. $A_k \rightarrow P \in F'$ if and only if $S \rightarrow A_kP \in F$, $k \in \{1, \ldots, n\}$, $P \in V_T^*$.

Proof idea - continued

We show that $L(G) = L(G')$.

Let $P \in L(G)$. If $S \to P \in F$, then $S \to P \in F'$, thus $P \in L(G')$. Otherwise, there is a derivation in G of the form

$$S \Longrightarrow_G A_{i_1} P_1 \Longrightarrow_G \ldots \Longrightarrow_G A_{i_{m-1}} P_{m-1} \ldots P_1 \Longrightarrow_G P_m \ldots P_1 = P,$$

to which there is a derivation

$$S \Longrightarrow_{G'} P_m A_{i_{m-1}} \Longrightarrow_{G'} P_m P_{m-1} A_{i_{m-2}} \Longrightarrow_{G'} \ldots \Longrightarrow_{G'} P_m \ldots P_2 A_{i_1} \Longrightarrow_{G'} P_m \ldots P_1.$$

This implies that $P \in L(G')$. Thus, $L(G) \subseteq L(G')$. The reverse inclusion comes from the symmetry.

Consequences

Corollary

$\mathcal{L}_3$ is closed under mirror image.

Corollary Every regular language can be generated by a left-linear grammar.

The proofs are immediate. To every right-linear grammar G we can construct a left linear grammar G' which generates the mirror image of $L(G)$. If $L \in \mathcal{L}_3$, then L^{-1} can be generated by a left-linear grammar G and by the previous theorem L^{-1} is regular. But then $(L^{-1})^{-1}$ is regular as well.

Example

Let $L_1 = \{a^n b^n c^k \mid n \geq 1, k \geq 1\}$ and $L_2 = \{a^k b^n c^n \mid n \geq 1, k \geq 1\}$.

Both L_1 and L_2 are linear languages and

$$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\}$$

is context-sensitive but not context-free language.

L_1 can be generated by $G_1 = (\{S, A\}, \{a, b, c\}, S, F_1)$, where
$F_1 = \{S \to Sc, S \to Ac, A \to ab, A \to aAb\}$ and
L_2 can be generated by $G_2 = (\{S, B\}, \{a, b, c\}, S, F_2)$, where
$F_2 = \{S \to aS, S \to aB, B \to bc, B \to bBc\}$.

Normal form for type 3 grammars

Theorem

Every type 3 (regular) language can be generated by a grammar having the following two types of rules:

1. $X \to aY$, where $X, Y \in V_N$, and $a \in V_T$,

2. $X \to \lambda$, where $X \in V_N$.

Proof idea

Let $G = (V_N, V_T, S, F)$ be a type 3 grammar. It is known that G has the following two types of rules:

1. $A \to PB$, where $A, B \in V_N$, and $P \in V_T^*$,

2. $A \to P$, where $A \in V_N$ and $P \in V_T^*$.

If $|P| = 1$, we are ready.

Proof idea - continued

Suppose that $|P| > 1$.

Let $P = a_1 \dots a_n$, $n \geq 2$, then we replace the rule $A \rightarrow a_1 \dots a_n B$ with the set of rules $\{A \rightarrow a_1 Z_1, Z_1 \rightarrow a_2 Z_2, \dots, Z_{n-1} \rightarrow a_n B\}$, where $Z_1, \dots, Z_{n-1}$ are new nonterminals.

Analogously, rules of the form $A \rightarrow a_1 \dots a_m$, $m \geq 1$ can be replaced by the set of rules $\{A \rightarrow a_1 Y_1, Y_1 \rightarrow a_2 Y_2, \dots, Y_{m-1} \rightarrow a_m Y_m, Y_m \rightarrow \lambda\}$, where $Y_1, \dots, Y_m$ are new nonterminals.

We obtain then a rule set F_1 with rules of the form $X \rightarrow aY$, $X \rightarrow Y$, $X \rightarrow \lambda$, where $X, Y \in V_N$ and $a \in V_T$. Next, we eliminate the chain rules.

Proof idea - continued

Let V'_N be the set of all nonterminals occurring in F_1 (including the new ones). For any $X \in V'_N$, let $Ch(X) = \{Y \mid Y \Longrightarrow^*_G X\}$.

We then define F' as follows:

1. $X \to aY$ if any only if there is some $Z \in V'_N$ such that $X \in Ch(Z)$ and $Z \to aY \in F_1$,

2. $X \to \lambda$ if any only if there is some $Z \in V'_N$ such that $X \in Ch(Z)$ and $Z \to \lambda \in F_1$.

It is easy to see that $L(G) = L(G')$, where $G' = (V'_N, V_T, S, F')$.

Corollary

Every type 3 (regular) language can be generated by a grammar having the following two types of rules:

1. $X \to aY$, where $X, Y \in V_N$, and $a \in V_T$,
2. $X \to a$, where $X \in V_N$ and $a \in V_T$.

The proof comes by eliminating the λ-rules from the previous normal form grammar.

Further Consequences

Corollary 1

The class of regular languages ($\mathcal{L}_3$) is properly included in the class of context-free languages ($\mathcal{L}_2$).

Corollary 2

The class of regular languages ($\mathcal{L}_3$) is properly included in the class of linear languages.

The proofs directly follow from the previous theorems and the previous example.

Closure Property of Context-Free Languages

Theorem

The class $\mathcal{L}_2$ (the class of context-free languages) is closed under intersection with type 3 (regular) languages.

Proof idea

Let L be an arbitrary context-free language ($L \in \mathcal{L}_2$) and let L' ($L' \in \mathcal{L}_3$) be a regular language. We show that $L \cap L'$ is context-free.

We first suppose that $\lambda \notin L$, L is generated by a Chomsky normal form grammar $G = (V_N, V_T, S, F)$ and L' by a grammar $G' = (V'_N, V'_T, S', F')$ in the previous normal form.

Let $\{X_1, \ldots, X_k\}$ be the set of nonterminals of G' for which $X_i \to \lambda$, $1 \le i \le k$ holds.

Proof idea - continued

We define the following context-free grammars

$$G_i = (V'_N \times (V_N \cup V_T) \times V'_N, V_T \cup V'_T, [S', S, X_i], F''),$$

for $i \in \{1, \ldots, k\}$ where

1. $[X, A, Y] \to [X, B, Z][Z, C, Y] \in F''$ for all $X, Y, Z \in V'_N$, if and only if $A \to BC \in F$,
2. $[X, A, Y] \to [X, a, Y] \in F''$ for all $X, Y \in V'_N$, if and only if $A \to a \in F$
3. $[X, a, Y] \to a \in F''$ if and only if $X \to aY \in F'$

We prove that $L \cap L' = \bigcup_{i=1}^{k} L(G_i)$.

Proof idea - continued

Let $P = a_1 \dots a_n \; a_i \in V_T,\ 1 \leq i \leq n$.

1. P can be derived in G if and only if for every i, $1 \leq i \leq n$, and for every sequence of nonterminals $Z_1, \dots, Z_n$ in V'_N there is a derivation of the form

$$[S', S, X_i] \Longrightarrow_{G_i}{}^{*} [S', a_1, Z_1][Z_1, a_2, Z_2] \dots [Z_{n-1}, a_n, X_i].$$

2. P can be derived in G' if and only if there exists a sequence of nonterminals $Z_1, \dots, Z_n$ and some $X_i \in V'_N$ with $X_i \to \lambda \in F'$ such that

$$S' \Longrightarrow_{G'} a_1 Z_1 \Longrightarrow_{G'} a_1 a_2 Z_2 \Longrightarrow_{G'} \dots \Longrightarrow_{G'} a_1 \dots a_n X_i \Longrightarrow_{G'} a_1 a_2 \dots a_n.$$

This means that P is derivable in G' if and only if there are $Z_i \in V'_N$, $1 \leq i \leq n$, and there is a derivation in G_i where

$[S', a_1, Z_1][Z_1, a_2, Z_2] \dots [Z_{n-1}, a_n, X_i] \Longrightarrow_{G_i}{}^{*} a_1 a_2 \dots a_n$ holds.

This implies that $P \in L \cap L'$ if and only if $P \in L(G_i)$. If $\lambda \in L$, we apply the usual techniques.

Self-embedding context-free grammar

Definition

A context-free grammar $G = (V_N, V_T, S, F)$ is called self-embedding if it has a nonterminal A with $A \Longrightarrow_G^* XAY$ for some $X, Y \in (V_N \cup V_T)^+$.

Theorem

If a context-free grammar G is not self-embedding, then it is left- or right-linear.

Proof idea

Without the loss of generality we may assume that $G = (V_N, V_T, S, F)$ is reduced. We distinguish two cases.

Case 1.

For each nonterminal A there is a derivation $A \Longrightarrow_G USZ$, where $U, Z \in (V_N \cup V_T)^*$.

Each rule $A \to W \in F$, where W has at least one occurrence of a nonterminal has either of the forms:

1. $A \to XBY$,
2. $A \to XB$,
3. $A \to BY$,
4. $A \to B$,

where $A, B \in V_N$ and $X, Y \in (V_N \cup V_T)^+$.

Proof idea - continued

Case 1, rule of the form 1.

If there is a rule of the form 1 ($A \to XBY$) in F then there is a derivation $A \Longrightarrow_G XBY \Longrightarrow_G^* XUSZY \Longrightarrow_G^* XUPAQZY$, and this contradicts to the non-selfembedding property.

Case 1, rules of the form 2, 3.

For rules of the form $A \to XB$ or $C \to DY$, then we obtain again contradiction since $A \Longrightarrow_G XB \Longrightarrow_G^* XU_1SZ_1 \Longrightarrow_G^* XU_1P_1CQ_1Z_1 \Longrightarrow_G XU_1P_1DYQ_1Z_1 \Longrightarrow_G^* XU_1P_1U_2SZ_2YQ_1Z_1 \Longrightarrow_G^* XU_1P_1U_2P_2AQ_2Z_2YQ_1Z_1$.

For the rule of the form 2 where $X \in V_T^*$ or that of the form 3 with $Y \in V_T^*$ the proof can be done similarly. This implies that F must have only rules either of the form $A \to XB$ and $A \to X$, where A, B are nonterminals and X is a terminal word, or else $A \to BY$ and $A \to Y$, where A, Y are nonterminals and Y is a terminal word. Thus, G is left- or right-linear.

Proof idea - continued

Case 2.

There is at least one nonterminal A_1 such that $A_1 \Longrightarrow_G^* USZ$ does not hold for any $U, Z \in (V_N \cup V_T)^*$

We prove the statement by induction. If $V_N = \{S\}$, the case must not hold, since $S \Longrightarrow_G^* S$. Suppose that the statement holds for any grammar with n nonterminals. We show that it holds for G with $n+1$ nonterminals as well.

Consider $G_1 = (V_N - \{S\}, V_T, A_1, F_1)$, where F_1 is obtained with cancelling all rules from F which contain S. Analogusly, consider $G_2 = (V_N - \{A_1\}, V_T \cup \{A_1\}, S, F_2)$, where F_2 is obtained from F with cancelling all rules with A_1 on the left-hand side. Both G_1 and G_2 are non-self-embedding since $F_1 \subseteq F$ and $F_2 \subseteq F$, thus $L(G_1)$ and $L(G_2)$ are non-self-embedding. It can be seen that the language that can be obtained by substituting $L(G_1)$ for A_1 and for all $\{a\}$ in $L(G_2)$, where $a \in V_T$ is equal to $L(G)$ and since $\mathcal{L}_3$ is closed under substitution under regular languages the statement holds.

Formal Languages - 5. Lecture

Erzsébet Csuhaj-Varjú

Gábor Kolonits

Department of Algorithms and Their Applications
Faculty of Informatics
Eötvös Loránd University
H-1117 Budapest
Pázmány Péter sétány 1/c
Hungary
E-mail: {csuhaj,kolomax}@inf.elte.hu

Regular Expressions

Motivation:

It is known that every finite language is a regular language. Furthermore, we know that the class of regular languages ($\mathcal{L}_3$) is closed under set union, catenation and iteration closure. This implies that starting from a finite number of finite languages and using the previous, regular operations, we obtain regular languages.

It is an interesting question whether or not this method is sufficient to describe $\mathcal{L}_3$, i.e., the class of regular languages?

Regular Expressions

Definition

A regular expression over a finite alphabet V is defined inductively as follows:

1. λ is a regular expression.
2. a is a regular expression for every $a \in V$.
3. If R is a regular expression over V, then $(R)^*$ is a regular expression over V as well.
4. If Q and R are regular expressions over V, then $(Q);(R)$ and $(Q)|(R)$ are regular expressions over V as well.

Symbol $*$, ; and $|$ denotes operation iteration, catenation and set union, respectively.

Regular Expressions

Every regular expression defines (denotes) some regular language.

- $L(\lambda) = \{\lambda\}$
- $L(a) = \{a\}$ for every $a \in V$
- $L((R)^*) = (L(R))^*$
- $L((Q);(R)) = L(Q)L(R)$
- $L((Q)|(R)) = L(Q) \cup L(R)$

Examples

Remark: Parentheses can be omitted if priority of operations is defined.

- a^* is the same as $(a)^*$ and denotes $\{a\}^*$
- $(a|b)^*$ is the same as $((a)|(b))^*$ and denotes $\{a, b\}^*$
- $a^*; b$ is the same as $((a)^*); b$ and denotes $\{a\}^*\{b\}$
- $b|a; b^*$ is the same as $(b)|((a); (b)^*)$ and denotes $\{b\} \cup \{a\}\{b\}^*$
- $(a|b); a^*$ is the same as $((a)|(b)); ((a)^*)$ and denotes $\{a, b\}\{a\}^*$

It is easy to see that $\{a, b\}\{a\}^*$ is the same as $\{a\}\{a\}^* \cup \{b\}\{a\}^*$. Thus, $(a|b)a^* = a; a^*|b; a^*$.

Axioms for Regular Expressions

Let P, Q, R be regular expression. Then

$$P|(Q|R) = (P|Q)|R \qquad P;(Q;R) = (P;Q);R$$

$$P|Q = Q|P \qquad P;(Q|R) = (P;Q)|P;R$$

$$(P|Q);R = P;R|Q;R \qquad P^* = \lambda|P;P^*$$

$$\lambda;P = P;\lambda = P \qquad P^* = (\lambda|P)^*$$

Axioms for Regular Expressions - continued

We need one more inference rule, namely if

$P = R|P;Q$ and $\lambda \notin Q$ then $P = R;Q^*$.

We add $\emptyset$ that denotes the empty language. In this case we do not need λ, because $\emptyset^* = \{\lambda\}$.

Thus, in the definition we may replace λ with $\emptyset$.

Then we add one more axiom

$$\emptyset;P = P;\emptyset = \emptyset.$$

The axioms and the two inference rules are sufficient to deduce any valid equation between regular expressions.

Regular Expressions versus Regular Languages

Theorem

Every regular expression denotes a regular (type 3) language, and conversely, every regular language can be denoted by a regular expr-ession.

Proof idea

1) The first part of the statement is obvious by the previous discussion.

2) We show that for any regular language L, generated by a regular grammar $G = (V_N, V_T, S, F)$ in normal form we can construct a regular expression which denotes L.

Let $V_N = \{A_1, \ldots, A_n\}$, $n \geq 1$, $S = A_1$. (Notice that each rule is either of the form $A_i \to aA_j$ or $A_i \to \lambda$, $a \in V_T$, $1 \leq i, j \leq n$.)

We say that derivation $A_i \Longrightarrow_G^* PA_j$ involves A_m iff A_m occurs in an intermediate sentential form between A_i and PA_j in the derivation.

The derivation $A_i \Longrightarrow_G^* PA_j$ is k-restricted iff $0 \leq m \leq k$ for every A_m which is involved in the derivation.

Proof idea - continued

We define

$E^k_{i,j} = \{P \in V_T^* \mid$ there is a k$-$restricted derivation $A_i \Longrightarrow^* PA_j\}$.

We prove by induction on k that $E^k_{i,j}$ is denoted by a regular expression for i, j, k where $0 \leq i, j, k \leq n$.

Basis: for $i \neq j$ $E^0_{i,j}$ is either empty or consists of some letters of V_T. ($a \in E^0_{i,j}$ if $A_i \to aA_j \in F$.) For $i = j$, $E^0_{i,i}$ contains λ and zero or more letters from V_T. Thus $E^0_{i,j}$ is denoted by a regular expression.

Induction step: Suppose that for a fixed k, $0 < k \leq n$, each of the sets $E^{k-1}_{i,j}$ is denoted by a regular expression. Then for every i, j, k it holds

$$E^k_{i,j} = E^{k-1}_{i,j} | E^{k-1}_{i,k}; (E^{k-1}_{k,k})^*; E^{k-1}_{k,j}.$$

Thus, $E^k_{i,j}$ is also denoted by a regular expression.

Let I_λ be the set if indices such that $A_i \to \lambda$ for $i \in I_\lambda$. Then $L(G) = \bigcup_{i \in I_\lambda} E^n_{I_\lambda}$, thus L is denoted by a regular expression.

Formal Languages - 6. Lecture

Erzsébet Csuhaj-Varjú
Gábor Kolonits

Department of Algorithms and Their Applications
Faculty of Informatics
Eötvös Loránd University
H-1117 Budapest
Pázmány Péter sétány 1/c
Hungary
E-mail: {csuhaj,kolomax}@inf.elte.hu

Automata

It is also reasonable to define a formal language with the aid of a device which is capable of processings strings of symbols.

Such a device, called automaton, has two possible reactions for an input string: accepts or rejects (yes or no). We note that it is realistic to consider the possibility of getting no answer at all.

Automata are analytical devices, while grammars use synthetic app-roach.

Finite Automata

Definition

A finite automaton is an ordered quintuple

$$A = (K, T, M, q_0, H)$$

1. where K is a finite nonempty set of states,
2. T is a finite alphabet of input symbols,
3. M is a mapping called the transition function from set $K \times T$ to K,
4. $q_0 \in K$ is the initial state,
5. $H \subseteq K$ is a set of accepting states.

Finite Automata - Working Mode

The way of working of a finite automaton can be considered as brisk moves at discrete time intervals.

Each move consists of reading the next input symbol and entering a state that is defined by the transition function, applied to the argument defined by the current state and the current input symbol.

The input is presented (we may assume) on an input tape that is divided into squares, where each square contains a single symbol.

In each move, the input tape is advanced by one square in front of a reading head connected to the finite state control device.

Finite Automata - Working Mode

At the beginning the finite automaton A is in the initial state q_0 and the reading head scans the first symbol of a word $P \in T^*$ put on the input tape.

Then, the finite automaton makes a sequence of moves and thus reads through the input word P.

If after reading (scanning) the last symbol of P the finite automaton A enters a state $q \in H$, then P is accepted by A, otherwise it is rejected.

Notice that by the above definition, the finite automaton never enters an infinite loop, since it has to read a new symbol for each move. This implies that it gives a definite answer on the acceptance/rejection of P in $|P|$ moves.

Example 1

Let

$$A = (K, T, M, q_0, H)$$

be a finite automaton with

$$K = \{q_0, q_1, q_2, q_3\} \;\; T = \{a, b\}, \;\; H = \{q_0\}$$

and let

$$\begin{array}{ll} M(q_0, a) = q_2, & M(q_0, b) = q_1, \\ M(q_1, a) = q_3, & M(q_1, b) = q_0, \\ M(q_2, a) = q_0, & M(q_2, b) = q_3, \\ M(q_3, a) = q_1, & M(q_3, b) = q_2. \end{array}$$

Then A accepts exactly those words which have an even number of as and also an even number of bs.

Transition table for A

M	a	b
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Example 1 - continued

$$\begin{array}{ll} M(q_0, a) = q_2, & M(q_0, b) = q_1, \\ M(q_1, a) = q_3, & M(q_1, b) = q_0, \\ M(q_2, a) = q_0, & M(q_2, b) = q_3, \\ M(q_0, a) = q_1, & M(q_3, b) = q_2. \end{array}$$

Let us start with $bbabab$.

The state sequence is q_0, q_1, q_0, q_2, q_3, q_1, q_0.

Deterministic - Nondeterministic Finite Automaton

The mapping M is a one-valued function, thus for each pair (q, a) in $K \times T$ we have a unique state s such that $M(q, a) = s$ holds. Therefore the finite automaton is called **deterministic**.

If we allow multivalued transition function, i.e., M is a mapping from $K \times T$ into 2^K, then we speak on a **nondeterministic** finite automaton. Here, the current state is to be considered as any of the states of a subset of K, rather than a unique state. This means that the initial state may be replaced with a set $Q_0 \subseteq K$ of initial states. It also may happen that for an input symbol a the set $M(q, a)$ is empty. In this case the automaton gets stuck.

Notation

The mapping M can also be given as

$$qa \to p \in M$$

iff $p \in M(q, a)$.

If M contains exactly one rule with the left-hand side qa for each pair (q, a) in $K \times T$, then the finite automaton is deterministic, otherwise it is nondeterministic.

Definition

Given a finite automaton $A = (K, T, M, Q_0, H)$ and two words $X, Y \in KT^*$ (where KT^* denotes the catenation of K and T^*) we say that A reduces X to Y in one move (in symbols $X \Longrightarrow_A Y$) iff there is a move $qa \to p \in M$ and a word $P \in T^*$ such that $X = qaP$ and $Y = pP$.

Definition

A finite automaton A reduces a word $X \in KT^*$ to a word $Y \in KT^*$ (in symbols $X \Longrightarrow_A^* Y$) iff $X = Y$ or there is some $Z \in KT^*$ such that $X \Longrightarrow_A^* Z$ and $Z \Longrightarrow_A Y$.

The relation $\Longrightarrow_A$ and its transitive and reflexive closure $\Longrightarrow_A^*$ are as essentially the same as the derivation in a grammar.

Language

The language accepted by a finite automaton A is defined as

$$L(A) = \{P \in T^* |\, q_0 P \Longrightarrow_A^* p \text{ for some } q_0 \in Q_0 \text{ and } p \in H\}.$$

The empty word λ is in $L(A)$ if and only if $Q_0 \cap H \neq \emptyset$. Each move $qa \to p \in M$ is length-decreasing rewriting rule but it can easily be seen that reductions correspond to reverse derivations in left-linear grammars.

Theorem

For every nondeterministic finite automaton A, we can give a type 3 grammar G such that $L(G) = L(A)$.

Proof idea

We first show that $L(A) \subseteq L(G)$.

Let $A = (K, T, M, Q_0, H)$ be a nondeterministic finite automaton. We define a grammar $G = (V_N, V_T, S, F)$ such that $V_N = K \cup \{S\}$, $V_T = T$ and

1. $p \to a \in F$ iff $q_0 a \to p \in M$ for some $q_0 \in Q_0$,
2. $p \to qa \in F$ iff $qa \to p \in M$,
3. $S \to p \in F$ iff $p \in H$,
4. $S \to \lambda \in F$ iff $Q_0 \cap H \neq \emptyset$.

Proof idea - continued

1. It is obvious that $\lambda \in L(G)$ iff $\lambda \in L(A)$.

2. Suppose that $P \neq \lambda \in L(A)$. Then there is a $q_0 \in Q_0$ and $p \in H$ such that $q_0 P \Longrightarrow_A^* p$ holds. Following this reduction in the reverse sense, we can construct the derivation $p \Longrightarrow_G^* q_0 P$ using the rules from group 2.

3. The last step of that derivation in G is the application of a rule $p_1 \to q_0 a$ where $q_0 \in Q_0$. Therefore we have a related rule $p_1 \to a$ in group 1. This implies that $p \Longrightarrow_G^* P$.

4. Finally, we take $S \to p$ from group 3 and thus, $S \Longrightarrow_G p \Longrightarrow_G^* P$.

Thus, $L(A) \subseteq L(G)$.

Proof idea - continued

We show now that $L(G) \subseteq L(A)$.

1. Let $P \in L(G)$ and $P \neq \lambda$, i.e., $S \Longrightarrow_G^* P$ and $P \in T^+$.

2. By the construction of G, then there should be a derivation of the form

$$S \Longrightarrow_G p \Longrightarrow_G^* p_1 Q \Longrightarrow_G aQ = P$$

where $p \in H$ and $p_1 \to a \in F$.

3. This implies that $q_0 P \Longrightarrow_A^* p$, thus $P \in L(A)$.

The rules in G are left-linear, but we know that we can find an equivalent right-linear grammar, which completes the proof.

Theorem

For every type 3 grammar G we can give a finite automaton A such that $L(A) = L(G)$ holds.

Proof idea

1. Without the loss of any generality we can assume that $G = (V_N, V_T, S, F)$ is in normal form (every production is one of the forms $X \to aY$, $X \to \lambda$, where $X, Y \in V_N$ and $a \in V_T$).

2. We construct $A = (K, T, M, Q_0, H)$ such that $K = V_N$, $T = V_T$, $Q_0 = \{S\}$ and $H = \{Z \in V_N \mid Z \to \lambda \in F\}$, and M is defined by

$$Xa \to Y \in M \text{ iff } X \to aY \in F$$

.

3. It is easy to see that to every derivation $S \Longrightarrow_G^* P$ in G we have a sequence of moves in A of the form $SP \Longrightarrow_A^* Z$ with $Z \in H$.

4. Also, conversely, to every reduction of the latter form in A we can find a corresponding derivation in G. Thus, the proof is completed.

Formal Languages - 7. Lecture

Erzsébet Csuhaj-Varjú

Gábor Kolonits

Department of Algorithms and Their Applications
Faculty of Informatics
Eötvös Loránd University
H-1117 Budapest
Pázmány Péter sétány 1/c
Hungary
E-mail: {csuhaj,kolomax}@inf.elte.hu

Finite Automata - continued

It is known that for any nondeterministic finite automaton A we can construct a regular grammar G such that $L(G) = L(A)$. It is also known that for any regular grammar G we can construct a finite automaton A such that $L(A) = L(G)$ holds.

Question: Is there a regular language that is accepted by a nondeterministic finite automaton but cannot be accepted by a deterministic one?

Answer: No.

Example

Let $A = (K, T, M, K_0, H)$ be a nondeterministic finite automaton with

$$\begin{aligned} K = \{0,1,2\}, T = \{a,b\}, K_0 = \{0\}, H = \{2\},& \\ M(0,a) = \{0,1\}, &\quad M = (0,b) = \{1\}, \\ M(1,a) = \emptyset, &\quad M = (1,b) = \{2\}, \\ M(2,a) = \{0,1,2\}, &\quad M = (2,b) = \{1\}. \end{aligned}$$

Example - continued

The corresponding deterministic finite automaton will have states

$$\emptyset, \{0\}, \{1\}, \{2\}, \{0,1\}, \{0,2\}, \{1,2\}\{0,1,2\}$$

and is given with $H = \{\{2\},\{0,2\},\{1,2\},\{0,1,2\}\}$ and M' where

$$\begin{aligned}
M'(\emptyset,a) &= \emptyset, & M'(\emptyset,b) &= \emptyset,\\
M'(\{0\},a) &= \{0,1\}, & M'(\{0\},b) &= \{1\},\\
M'(\{1\},a) &= \emptyset, & M'(\{1\},b) &= \{2\},\\
M'(\{2\},a) &= \{0,1,2\}, & M'(\{2\},b) &= \{1\},\\
M'(\{0,1\},a) &= \{0,1\}, & M'(\{0,1\},b) &= \{1,2\},\\
M'(\{0,2\},a) &= \{0,1,2\}, & M'(\{0,2\},b) &= \{1\},\\
M'(\{1,2\},a) &= \{0,1,2\}, & M'(\{1,2\},b) &= \{1,2\},\\
M'(\{0,1,2\},a) &= \{0,1,2\}, & M'(\{0,1,2\},b) &= \{1,2\}.
\end{aligned}$$

Deterministic versus Nondeterministic Finite Automata

Theorem 1

For every nondeterministic finite automaton $A = (K, T, M, K_0, H)$ we can construct a deterministic finite automaton $A' = (K', T, M', q_0', H')$ such that $L(A) = L(A')$ holds.

Proof idea of Theorem 1

Let K' be the set of all subsets of K.

We define $M' : K' \times T \to K'$ as $M'(q', a) = \bigcup_{q \in q'} M(q, a)$. Further, let $q'_o = K_0$ and $H' = \{q' \in K' \mid q' \cap H \neq \emptyset\}$

We first show that $L(A) \subseteq L(A')$. To do this, we prove the next lemma.

Proof idea of Theorem 1 - continued

Lemma 1

For every $p, q \in K$, $q' \in K'$ and $P, Q \in T^*$ if

$$qP \Longrightarrow_A^* pQ \text{ and } q \in q'$$

then there is a $p' \in K'$ such that

$$q'P \Longrightarrow_{A'}^* p'Q \text{ and } p \in p'.$$

Proof idea of Lemma 1:

The statement can be proved by induction on the number of moves involved in the reduction $qP \Longrightarrow_A^* pQ$.

For zero moves the statement is trivial. Suppose that it holds for n moves $n \geq 0$.

Proof idea of Lemma 1 - continued

(**Lemma 1** : For every $p, q \in K$, $q' \in K'$ and $P, Q \in T^*$ if

$$qP \Longrightarrow_A^* pQ \text{ and } q \in q'$$

then there is a $p' \in K'$ such that

$$q'P \Longrightarrow_{A'}^* p'Q \text{ and } p \in p'.)$$

Let $qP \Longrightarrow_A^* pQ$ consist of $n+1$ moves. Then for some $q_1 \in K$ and $P_1 \in T^*$ $qP \Longrightarrow_A q_1P_1 \Longrightarrow_A^* pQ$ holds. Hence, there is an $a \in T$ with $P = aP_1$ and $q_1 \in M(q, a)$. But $M(q, a) \subseteq M'(q', a)$ for $q \in q'$, and thus we can choose $q'_1 = M'(q', a)$ which implies that $q'P \Longrightarrow_{A'}^* q'_1P_1$ with $q_1 \in q'_1$. By the induction hypothesis, for some $p' \in K'$

$$q'_1P_1 \Longrightarrow_{A'}^* p'Q \text{ and } p \in p'.$$

Thus Lemma 1 holds.

Proof idea of Theorem 1 - continued

Let $P \in L(A)$, that is $q_0 P \Longrightarrow_A^* p$ for some $q_0 \in K_0$ and let $p \in H$. Then, by Lemma 1, for some p' we have $q_0' P \Longrightarrow_{A'}^* p'$ and $p \in p'$.

By the definition of H', $p \in p'$ and $p \in H$ implies that $p' \in H'$ which gives $L(A) \subseteq L(A')$.

Next we show that $L(A') \subseteq L(A)$.

This will be done by Lemma 2.

Proof idea of Theorem 1 - continued

Lemma 2

For every $p', q' \in K'$, $p \in K'$ and $P, Q \in T^*$ if

$$q'P \Longrightarrow^*_{A'} p'Q \text{ and } p \in p'$$

then there is a $q \in K$ such that

$$qP \Longrightarrow^*_{A} pQ \text{ and } q \in q'.$$

Proof idea of Lemma 2: It is done by induction of moves. For zero moves the assertion is trivial. Suppose that it holds for n moves, $n \geq 0$, and let $q'P \Longrightarrow^*_{A'} p'Q$ consist of $n+1$ moves. Then $q'P \Longrightarrow^*_{A'} q'_1 Q_1 \Longrightarrow_{A'} p'Q$ holds where $Q_1 = aQ$ for some $a \in T$ and $p' \in K'$. Then,

$$p \in p' = M'(q'_1, a) = \bigcup_{q_1 \in q'_1} M(q_1, a),$$

thus there should be $q_1 \in q'_1$ for which $p \in M(q_1, a)$. For such q_1 we have $q_1 Q_1 \Longrightarrow_{A} pQ$ and by the induction hypothesis this gives $q \in q'$ where $qP \Longrightarrow^*_{A} q_1 Q_1$. Thus Lemma 2 holds.

Proof idea of Theorem 1 - continued

Now let $q_0'P \Longrightarrow^*_{A'} p'$ and $p' \in H'$. By definition of H', we have some $p \in p'$ with $p \in H$ and thus, by Lemma 2, there is some $q_0 \in q_0'$ such that $q_0 P \Longrightarrow^*_{A} p$ holds. Thus, $L(A') \subseteq L(A)$. This implies that the theorem holds.

Consequences

Corollary

The class of regular languages (the class $\mathcal{L}_3$) is closed under complementation.

Proof idea:

Let L be a regular language accepted by some deterministic automaton $A = (K, T, M, q_0, H)$. Then $\bar{L} = T^* - L$ is accepted by $\bar{A} = (K, T, M, q_0, K - H)$.

Consequences

Corollary

The class of regular languages is closed under intersection.

Proof idea:

It is known that the class of regular languages is closed under union. Then,

$$L_1 \cap L_2 = \overline{\bar{L}_1 \cup \bar{L}_2}$$

which implies the result.

Consequences

Corollary

It is decidable whether two arbitrary regular grammars generate the same language.

Proof idea:

Let G_1 and G_2 be regular grammars generating languages L_1 and L_2, respectively. The language $L_3 = (L_1 \cap \bar{L}_2) \cup (\bar{L}_1 \cap L_2)$ is also regular, so there is a regular grammar G_3 that generates L_3, and G_3 can be constructed from G_1 and G_2. But $L_1 = L_2$ if and only if $L_3 = \emptyset$ which is decidable for every context-free grammar G_3.

Auxiliary notions

Let L be a language over an alphabet T. The equivalence relation E_L induced by L is the binary relation on the set of words over T if the following condition holds: PE_LQ holds if and only if there is no word R over T such that exactly one of the words PR and QR belongs to L.

E_L is right invariant: Whenever PE_LQ holds then PRE_LQR holds for any $R \in T^*$ as well.

By the index of E_L we mean the number of its equivalence classes.

Myhill-Nerode Theorem

The following three statements are equivalent:

1. $L \subseteq T^*$ is accepted by a deterministic finite automaton.

2. L is the union of some equivalence classes of a right invariant equivalence relation of finite index on T^*.

3. Let equivalence relation E_L be defined over T^* as follows: PE_LQ if and only if for any $R \in T^*$ it holds that $PR \in L$ if and only if $QR \in L$. Then E_L is of finite index.

Minimum state automaton

A deterministic finite automaton A is said to be minimum state automaton if there is no deterministic finite state automaton A' which has less number of states than A and accepts the same language.

Theorem

The minimum state automaton accepting a regular language L is unique up to an isomorphism.

Minimization of a deterministic finite state automaton

Let $A = (K, T, M, q_0, H)$ be a deterministic finite state automaton. Let us define an equivalence relation R on K such that pRq if for each input string $x \in T^*$ $px \Longrightarrow_A^* r$ if and only if $qx \Longrightarrow_A^* r'$ holds for some $r, r' \in H$. (Note that $r = r'$ is possible, but not necessary).

We say that p and q are distinguishable, if there exists $z \in T^*$ such that either $px \Longrightarrow_A^* r$ with $r \in H$ or $qx \Longrightarrow_A^* r'$ with $r' \in H$ but not both holds.

If p and q are equivalent, then $M(p, a)$ and $M(q, a)$ are equivalent for any $a \in T$ as well.

(If this is not the case, $M(p, a) = r$ and $M(q, a) = s$ are distinguishable by $x \in T^*$, then p and q are distinguishable by ax.)

DFA

Algorithm for finding a minimum state deterministic finite state automaton

We get a partition of the set of states K as follows:

Step 1: Consider the set of states in K. Divide them into two blocks, H and $K - H$. (Any state in H is distinguishable from a state in $K - H$ by λ.) Repeat the following splits until no more split is possible.

Algorithm for finding a minimum state deterministic finite state automaton - continued

Step 2: Consider the set of states in a block. Consider their successor state for a (i.e., if $p \in K$ is in the block and $M(p, a) = s$, then s is called the a-successor of p.) If they belong to different blocks then split this block into as many blocks as the a-successors determine.)

For example, if the block consists of states p, q, r and $M(p, a) = s_1$, $M(q, a) = s_2$, and $M(r, a) = s_3$ and s_1, s_2 belong to the same block but s_3 not, then the original block is split into two blocks, one consisting of p and q and one consisting of r.

Algorithm for finding a minimum state deterministic finite state automaton - continued

Step 3: For each block B_i consider a state b_i. Construct $A = (K', T, M', q'_0, H')$ where

$$K' = \{b_i \mid B_i \text{ is a block of the partition obtained in Step 2}\}.$$

q'_0 corresponds to the block containing q_0. $M(b_i, a) = b_j$ if there exists $q_i \in B_i$ and $q_j \in B_j$ such that $M(q_i, a) = q_j$ holds. H' consists of states corresponding to the blocks containing the states of H.

Formal Languages - 8. Lecture

Erzsébet Csuhaj-Varjú
Gábor Kolonits

Department of Algorithms and Their Applications
Faculty of Informatics
Eötvös Loránd University
H-1117 Budapest
Pázmány Péter sétány 1/c
Hungary
E-mail: {csuhaj,kolomax}@inf.elte.hu

Pushdown Automata

Generalization of finite automata - with a potentially infinite pushdown store and a finite state control device

The pushdown store is similar to a stack where new data are added always on the top of the previous recordings - the latter being pushed down the store - but reading occurs in the reverse order and clears the information that has been read.

The storage technique is called first-in-last-out (FILO).

Definition

A pushdown automaton is an ordered seventuple

$$A = (Z, K, T, M, z_0, q_0, H),$$

where

- Z is a finite set of pushdown symbols,
- K is a finite set of internal states,
- T is a finite set of input symbols,
- M is a mapping from $Z \times K \times (T \cup \{\lambda\})$ into the finite subsets of $Z^* \times K$, the transition function,
- $z_0 \in Z$ is the initial symbol,
- $q_0 \in K$ is the initial state,
- $H \subseteq K$ is the set of accepting states.

Definition

The configuration of a pushdown automaton is a word of the form Wq where $W \in Z^*$ is the current contents of the pushdown store and $q \in K$ is the current state of the finite control.

The initial configuration is z_0q_0.

Note: in other sources the configuration of a pushdown automaton is a triplet (W, q, v), where $W \in Z^*$ and $q \in K$ are the same as above, and $v \in T^*$ is what is left from the input during the computation. In this case the initial configuration for an input word $u \in T^*$ is (z_0, q_0, u).

A direct move of a pushdown automaton

Suppose that the reading head scans input symbol a, the finite state control is in state q, and the top symbol of the pushdown store is z. Let $M(z,q,a) = \{(u_1,r_1),\ldots,(u_n,r_n)\}$, where $u_i \in Z^*$ and $r_i \in K$, $1 \leq i \leq n$. Then A enters state r_i and replaces z with u_i for some i and its reading head moves one cell to the right on the input tape.

If $M(z,q,\lambda)$ is not empty, then a so-called λ-move can be performed.

If the input tape contains a word $P \in T^*$ and starting from the initial configuration z_0q_0 and performing a sequence of moves the pushdown automaton A enters a configuration Wp, where p is an accepting state, then we say that A accepts P.

Definition

A pushdown automaton A reduces some word $X \in Z^+KT^*$ to a word $Y \in Z^*KT^*$ in one move, denoted by $X \Longrightarrow_A Y$ iff there are $z \in Z$, $q, p \in K$, $a \in T \cup \{\lambda\}$, $W, u \in Z^*$ and $P \in T^*$ such that $(u, p) \in M(z, q, a)$ with $X = WzqaP$ and $Y = WupP$.

A pushdown automaton A reduces some word $X \in Z^+KT^*$ to a word $Y \in Z^*KT^*$, denoted by $X \Longrightarrow_A^* Y$ iff $X = Y$ or else there exists a finite sequence of words $X_1, \ldots, X_n \in Z^*KT^*$ such that $X = X_1$, $Y = X_n$, and $X_i \Longrightarrow_A X_{i+1}$ holds for $1 \leq i \leq n-1$.

Definition

The language accepted by a pushdown automaton A is defined as

$$L(A) = \{P \in T^* \mid z_0 q_0 P \Longrightarrow_A^* Wp \text{ for some } W \in Z^*, p \in H\}.$$

Remark: The mapping M can be given in terms of rewriting rules this way:

1. $zqa \to up \in M$ iff $(u, p) \in M(z, q, a)$,

2. $zq \to up \in M$ iff $(u, p) \in M(z, q, \lambda)$

Definition

A pushdown automaton $A = (Z, K, T, M, z_0, q_0, H)$ is said to be deterministic iff for every pair $(z, q) \in Z \times K$ either (1) $M(z, q, a)$ contains exactly one element for each $a \in T$, while $M(z, q, \lambda) = \emptyset$, or (2) $M(z, q, \lambda)$ contains exactly one element while $M(z, q, a) = \emptyset$ for each $a \in T$.

Remark: Deterministic pushdown automata are less powerful than nondeterministic pushdown automata.

Example languages:

$$L_1 = \{PcP^{-1} \mid P \in \{a, b\}^*\}$$
$$L_2 = \{PP^{-1} \mid P \in \{a, b\}^*\}$$

The first one can be accepted by a deterministic pushdown automaton, the second one not, both can be accepted by a nondeterministic pushdown automaton.

Definition

The language $N(A)$ accepted by a pushdown automaton A with empty store is defined as

$$N(A) = \{P \in T^* \mid z_0 q_0 P \Longrightarrow_A^* p \text{ for some } p \in K\}.$$

Note that if the pushdown store is empty, then the automaton is blocked since no move is defined for the empty store. (This is why we need z_0.)

Lemma 1

For every pushdown automaton A we can give a pushdown automaton A' such that $L(A) = N(A')$ holds.

equivalent PA

Proof idea of Lemma 1

Let

$$A = (Z, K, T, M, z_0, q_0, H)$$

be a pushdown automaton. We define

$$A' = (Z \cup \{z'_0\}, K \cup \{q', q'_h\}, M', z'_0, q'_0, \emptyset),$$

where $z'_0 \notin Z$, $q'_0, q'_h \notin K$ and M' is defined as follows.

$$\begin{aligned} M'(z'_0, q'_0, \lambda) = \{(z'_0 z_0, q_0)\}, \\ M'(z, q, a) = M(z, q, a), z \in Z, q \in K, a \in T, \\ M(z, q, \lambda) \subseteq M'(z, q, \lambda), z \in Z, q \in K, \\ (\lambda, q'_h) \in M'(z, q, \lambda), z \in Z \cup \{z'_0\}, q \in H \cup \{q'_h\}. \end{aligned}$$

A' simulates the work of A; symbol $z'_0 \neq z_0$ is needed to prevent A' from accepting some $P \notin L(A)$ which empties the pushdown in A.

Proof idea continued

Let $P \in L(A)$. Then

$$z_0 q_0 P \Longrightarrow_A^* Wq,$$

where $W \in Z^*$ and $q \in H$. But then

$$z_0' q_0' P \Longrightarrow_{A'} z_0' z_0 q_0 P \Longrightarrow_{A'}^* z_0' W q \Longrightarrow_{A'}^* q_h',$$

which implies that $P \in N(A')$.

Proof idea continued

Let $P \in N(A')$. Then

$$z_0' q_0' P \Longrightarrow_{A'}^* q',$$

for some $q' \in K \cup \{q_0', q_h'\}$.

But then the first step should be $z_0' q_0' P \Longrightarrow_{A'} z_0' z_0 q_0 P$.

Further, z_0' can only be cancelled from the pushdown with λ-move of type $(\lambda, q_h') \in M'(z, q, \lambda), z \in Z \cup \{z_0'\}, q \in H \cup \{q_h'\}$.

Thus, there is some $q \in H$ and $W \in Z^*$ such that

$$z_0' z_0 q_0 P \Longrightarrow_{A'}^* z_0' W q \Longrightarrow_{A'}^* q_h',$$

where $z_0 q_0 P \Longrightarrow_A^* W q$, which implies that $P \in L(A)$.

by CF grammar , PA accepting

Theorem 1

For every context-free grammar G, we can give a pushdown automaton A such that $L(A) = L(G)$.

Proof idea of Theorem 1

Let $G = (V_N, V_T, S, F)$ be a Chomsky-normal form grammar except possibly the rule $S \to \lambda$, but then S does not occur on the right-hand side of any rule. We construct a pushdown automaton A such that $L(G) = L(A)$ holds.

Proof idea continued

Let $A = (V_N \cup \{z_0\}, K, V_T, M, z_0, q_0, \{q_h\})$ such that

$$K = (\bigcup_{X \in V_N} q_X) \cup \{q_0, q_h\}$$

and M is defined with

$z_0 q_0 \to z_0 q_S \in M$ iff $S \to \lambda \in F$,

$z_0 q_0 a \to z_0 q_X \in M$ iff $X \to a \in F$,

$Z q_Y a \to ZY q_X \in M$ for every $Z \in V_N \cup \{z_0\}, Y \in V_N, X \to a \in F$

$Z q_Y \to q_X \in M$ iff $X \to ZY \in F$,

$z_0 q_S \to q_h \in M$

Proof idea continued

Let $P \in L(A)$. Then $z_0 q_0 P \Longrightarrow_A^* W q_h$ for some $W \in Z^*$. Then we should have

$$z_0 q_0 P \Longrightarrow_A^* z_0 q_S \Longrightarrow_A q_h.$$

If $P = \lambda$ then $z_0 q_0 \Longrightarrow_A z_0 q_S$ and thus $S \to \lambda \in F$. Otherwise we have

$$z_0 q_0 P \Longrightarrow_A z_0 q_X Q \Longrightarrow_A^* z_0 q_S \Longrightarrow_A q_h,$$

where $P = aQ$ for some $a \in V_T$ and $Q \in V_T^*$ and $X \to a \in F$. Then it is easy to show that

$$z_0 q_0 P \Longrightarrow_A^* z_0 q_S \text{ implies } S \Longrightarrow_G^* P.$$

Thus, $L(A) \subseteq L(G)$. The reverse inclusion, $L(G) \subseteq L(A)$ can also be easily shown.

PA accepting by CF grammar

Theorem 2

For every context-free grammar, G we can give a pushdown automaton A such that $N(A) = L(G)$.

Proof idea of Theorem 2

Let $G = (V_N, V_T, S, F)$ be a context-free grammar. We will construct a pushdown automaton that will simulate the derivations of G in the pushdown store.

Proof idea continued

Let $A = (V_N \cup V_T, \{q_0\}, V_T, M, S, q_0, \emptyset)$ be such that

$$M(a, q_0, a) = \{(\lambda, q_0)\}$$

for every $a \in V_T$, and

$$M(A, q_0, \lambda) = \{(u^{-1}, q_0) | \ A \to u \in F\}$$

for every $A \in V_N$.
It is easy to show that A accepts a word iff it can be generated by G.

CF grammar generating language accepted by PA.

Theorem 3

For every pushdown automaton A there exists a context-free grammar G such that $N(A) = L(G)$.

Proof idea of Theorem 3

Let $A = (Z, K, T, M, z_0, q_0, \emptyset)$ be a pushdown automaton. We define the context-free grammar $G = (V_N, V_T, S, F)$ as follows: $V_T = T$ and the elements of V_N are triplets of the form

$$[q, x, p]$$

where $q, p \in K$ and $x \in Z$. Further, we have a new symbol S and $V_N = (K \times Z \times K) \cup \{S\}$. The rules in F are defined as follows:

1. $S \to [q_0, z_0, p]$ for all $p \in K$,
2. If $xqa \to y_1 \ldots y_m p_m \in M$, for $a \in V_T \cup \{\lambda\}$, then for every $p_0, \ldots, p_{m-1} \in K$ we have $[q, x, p_0] \to a[p_m, y_m, p_{m-1}] \ldots [p_1, y_1, p_0] \in F$.
3. For $xqa \to p \in M$ let $[q, x, p] \to a \in F$ (the case $m = 0$.)
4. No other rules are in F.

Reference:

György E. Révész, Introduction to Formal Languages, Dover, 2012, Chapter 6.2

Formal Languages - 9. Lecture

Erzsébet Csuhaj-Varjú

Gábor Kolonits

Department of Algorithms and Their Applications
Faculty of Informatics
Eötvös Loránd University
H-1117 Budapest
Pázmány Péter sétány 1/c
Hungary
E-mail: {csuhaj,kolomax}@inf.elte.hu

Theorem 1

For every context-free grammar G, we can give a pushdown automaton A such that $L(A) = L(G)$.

Proof idea of Theorem 1

Let $G = (V_N, V_T, S, F)$ be a Chomsky-normal form grammar except possibly the rule $S \to \lambda$, but then S does not occur on the right-hand side of any rule. We construct a pushdown automaton A such that $L(G) = L(A)$ holds.

Proof idea continued

Let $A = (V_N \cup \{z_0\}, K, V_T, M, z_0, q_0, \{q_h\})$ such that

$$K = (\bigcup_{X \in V_N} q_X) \cup \{q_0, q_h\}$$

and M is defined with

$z_0 q_0 \to z_0 q_S \in M$ iff $S \to \lambda \in F$,

$z_0 q_0 a \to z_0 q_X \in M$ iff $X \to a \in F$,

$Z_0 q_Y a \to Z Y q_X \in M$ for every $Z \in V_N \cup \{z_0\}, Y \in V_N, X \to a \in F$

$Z q_Y \to q_X \in M$ iff $X \to ZY \in F$,

$z_0 q_S \to q_h \in M$

Proof idea continued

Let $P \in L(A)$. Then $z_0 q_0 P \Longrightarrow_A^* W q_h$ for some $W \in Z^*$. Then we should have

$$z_0 q_0 P \Longrightarrow_A^* z_0 q_S \Longrightarrow_A q_h.$$

If $P = \lambda$ then $z_0 q_0 \Longrightarrow_A z_0 q_S$ and thus $S \to \lambda \in F$. Otherwise we have

$$z_0 q_0 P \Longrightarrow_A z_0 q_X Q \Longrightarrow_A^* z_0 q_S \Longrightarrow_A q_h,$$

where $P = aQ$ for some $a \in V_T$ and $Q \in V_T^*$ and $X \to a \in F$. Then it is easy to show that

$$z_0 q_0 P \Longrightarrow_A^* z_0 q_S \text{ implies } S \Longrightarrow_G^* P.$$

Thus, $L(A) \subseteq L(G)$. The reverse inclusion, $L(G) \subseteq L(A)$ can also be easily shown.

Theorem 2

For every pushdown automaton A we can give context-free grammar G such that $L(G) = N(A)$.

Proof idea:

Let $A = (Z, K, T, M, z_0, q_0, \emptyset)$ be a pushdown automaton.

We define $G = (V_N, V_T, S, F)$ such that $V_T = T$, and the elements of V_N are the ordered triplets of the form $[q, x, p]$, where $q, p \in K$ and $x \in Z$. We introduce a new symbol S and thus $V_N = K \times Z \times K \cup \{S\}$.

Proof idea - continued:

We define F as follows:

1. Let $S \to [q_0, z_0, p] \in F$ for all $p \in K$.

2. If $xqa \to y_1 \dots y_m p_m \in M$ where $a \in (T \cup \{\lambda\})$, then for every $p_0, \dots, p_{m-1} \in K$ let $[q, x, p_0] \to a[p_m, y_m, p_{m-1}] \dots [p_1, y_1, p_0] \in F$.

   (For $m = 0$ let $[q, x, p_0] \to a \in F$).

3. No other rules are in F.

Proof idea - continued:

We first show that $L(G) \subseteq N(A)$. To this, we prove that for every $x \in Z$, $q, p \in K$, $P \in T^*$

$$[q, x, p] \Longrightarrow_G^* P \text{ implies } xqP \Longrightarrow_A^* p.$$

The proof is done by induction on the number of steps in $[q, x, p] \Longrightarrow_G^* P$.

For one step we obviously have $P \in \{a\} \cup \{\lambda\}$ for some $a \in T$, thus $xqa \to p \in M$.

Proof idea - continued:

Assume that the statement is valid for every derivaton of steps at most n and let the derivation $[q, x, p] \Longrightarrow_G^* P$ have $n + 1$ steps. Then it must be of the form

$$[q, x, p_0] \to a[p_m, y_m, p_{m-1}] \dots [p_1, y_1, p_0] \Longrightarrow_G^* P,$$

which implies that there are words $P_m, P_{m-1}, \dots, P_1 \in T^*$ such that $P = aP_mP_{m-1} \dots P_1$ and

$$[p_i, y_i, p_{i-1}] \Longrightarrow_G^* P_i \text{ holds for } 1 \le i \le m.$$

By the induction hypothesis $y_ip_iP_i \Longrightarrow_A^* p_{i-1}$ and by the definition of G

$$xqa \Longrightarrow_A y_1 \dots y_mP_m.$$

Hence we obtain

$xqP = xqaP_m \dots P_1 \Longrightarrow_A y_1 \dots y_mp_mP_m \dots P_1 \Longrightarrow_A^* y_1 \dots y_{m-1}p_{m-1}P_{m-1} \dots P_1 \Longrightarrow_A^* p_0 = p$.

Proof idea - continued:

Thus, if $P \in L(G)$, then there is a $p \in K$ such that

$$S \Longrightarrow_G [q_0, z_0, p] \Longrightarrow_G^* P$$

which implies that

$$z_0 q_0 P \Longrightarrow_A^* p,$$

thus $P \in N(A)$.

Proof idea - continued:

To prove the reverse inclusion we show that

$$xqP \Longrightarrow_A^* p \text{ implies } [q,x,p] \Longrightarrow_G^* P.$$

We use induction on the number of moves. For the single move we have $xqa \to p \in M$ which implies that $[q,x,p] \to a \in F$.

For more than one move $xqP \Longrightarrow_A^* p$ must have the form

$$xqP = xqaQ \Longrightarrow_A y_1 \ldots y_m p_m Q \Longrightarrow_A^* p.$$

Let us consider this reduction until symbol y_{m-1} becomes the top of the stack.

Proof idea - continued:

For more than one move $xqP \Longrightarrow_A^* p$ must have the form

$$xqP = xqaQ \Longrightarrow_A y_1 \dots y_m p_m Q \Longrightarrow_A^* p.$$

Let us consider this reduction until symbol y_{m-1} becomes the top of the stack. This means that there is some $P_m \in T^*$ such that $Q = P_m Q_1$ and $y_m p_m P_m \Longrightarrow_A^* p_{m-1}$ for some $p_{m-1} \in K$ and thus

$$xqP \Longrightarrow_A y_1 \dots y_m p_m P_m Q_1 \Longrightarrow_A^* y_1 \dots y_{m-1} p_{m-1} Q_1$$

.

Proof idea - continued:

Continuing, we obtain $P_{m-1}, \ldots, P_1$ such that

$$P = aP_mP_{m-1} \ldots P_1$$

and

$$y_ip_iP_i \Longrightarrow_A^* p_{i-1}, \text{ for } i = 1, \ldots m.$$

By the induction hypothesis this implies

$$[p_i, y_i, p_{i-1}] \Longrightarrow_G^* P_i \text{ for } i = 1, \ldots m$$

and by the definition of G

$$[q, x, p] \Longrightarrow_G a[p_m, y_m, p_{m-1}] \ldots [p_1, y_1, p]$$

which implies that $[q, x, p] \Longrightarrow_G^* P$.

Proof idea - continued:

Thus, if $P \in N(A)$ then $z_0 q_0 P \Longrightarrow_A^* p$ for some $p \in K$ and thus

$$S \Longrightarrow_G [q_0, z_0, p] \Longrightarrow_G^* P,$$

i.e., $N(a) \subseteq L(G)$.

Corollary For every pushdown automaton A, we can give a pushdown automaton A' such that $L(A') = N(A)$.

Corollary For every pushdown automaton A, we can give a context-free grammar G such that $L(A) = L(G)$.

Reference:

György E. Révész, Introduction to Formal Languages, Dover, 2012, Chapter 6.2