Introduction to Computer Science

Géza Horváth

Second Lecture

Chomsky hierarchy

The Chomsky hierarchy was described first by Noam Chomsky in 1956. It classifies the generative grammars based on the forms of their production rules. The Chomsky hierarchy also classifies languages, based on the classes of generative grammars generating them.

Chomsky hierarchy – definition

Definition (Chomsky hierarchy)

Let $G = (N, T, S, P)$ be a generative grammar.

- *Type 0 or unrestricted grammars. Each generative grammar is unrestricted.*
- *Type 1 or context-sensitive grammars. G is called context-sensitive, if all of its production rules have a form*

$$p_1 A p_2 \to p_1 q p_2,$$

or

$$S \to \lambda,$$

where $p_1, p_2 \in V^*$, $A \in N$ and $q \in V^+$. If $S \to \lambda \in P$ then S does not appear in the right hand side word of any other rule.

Chomsky hierarchy – definition

Definition (Chomsky hierarchy)

- *Type 2 or context-free grammars. The grammar G is called context-free, if all of its productions have a form*

$$A \to p,$$

where $A \in N$ and $p \in V^*$.

- *Type 3 or regular grammars. G is called regular, if all of its productions have a form*

$$A \to r,$$

or

$$A \to rB,$$

where $A, B \in N$ and $r \in T^*$.

Chomsky hierarchy – unrestricted example

Example

Let the generative grammar G_0 be

$G_0 = (\{S, X\}, \{x, y\}, S, P)$
$P = \{$
$S \to SXSy,$
$XS \to y,$
$X \to SXS,$
$S \to yxx$
$\}.$

This grammar is unrestricted, because the second rule is not a context-sensitive rule.

Chomsky hierarchy – context-sensitive example

Example

Let the generative grammar G_1 *be*

$G_1 = (\{S, X\}, \{x, y\}, S, P)$
$P = \{$
$S \to SXSy,$
$XSy \to XyxXy,$
$S \to yXy,$
$X \to y$
$\}.$

This grammar is context-sensitive, because each production rule has an appropriate form.

Chomsky hierarchy – context-free example

Example

Let the generative grammar G_2 be

$G_2 = (\{S, X\}, \{x, y\}, S, P)$
$P = \{$
$S \to SyS,$
$S \to XX,$
$S \to yxy,$
$X \to ySy,$
$X \to \lambda$
$\}.$

This grammar is context-free, because the left hand side of each production rule is a nonterminal symbol.

Chomsky hierarchy – regular example

Example

Let the generative grammar G_3 be

$G_3 = (\{S, X\}, \{x, y\}, S, P)$
$P = \{$
$S \to xyS,$
$S \to X,$
$X \to yxS,$
$S \to x,$
$X \to \lambda$
$\}.$

This grammar is regular, because the left hand side of each production rule is a nonterminal, and the right hand side contains at most one nonterminal symbol, in last position.

Chomsky hierarchy – example

What is the type of the following grammar?

$G = (\{S, A\}, \{0, 1\}, S, P)$
$P = \{$
$S \to 0101A,$
$S \to \lambda,$
$A \to 1S,$
$A \to 000$
$\}.$

Chomsky hierarchy – example

What is the type of the following grammar?

$G = (\{S, A, B\}, \{0, 1\}, S, P)$
$P = \{$
$S \to 0A01B,$
$S \to \lambda,$
$0A01 \to 01A101,$
$A \to 0BB1,$
$B \to 1A1B,$
$B \to 0011,$
$A \to 1$
$\}.$

Chomsky hierarchy – example

What is the type of the following grammar?

$G = (\{S, A, B\}, \{0, 1\}, S, P)$
$P = \{$
$S \to 0ABS1,$
$S \to 10,$
$0AB \to 01SAB,$
$1SA \to 111,$
$A \to 0,$
$B \to 1$
$\}.$

Chomsky hierarchy – example

What is the type of the following grammar?

$G = (\{S, A\}, \{0, 1\}, S, P)$
$P = \{$
$S \to 00A,$
$S \to \lambda,$
$A \to \lambda,$
$A \to S1S$
$\}.$

Chomsky hierarchy – languages

Definition

The language L is called recursively enumerable, if there exists an unrestricted grammar G such that $L = L(G)$.

Definition

The language L is context-sensitive, if there exists a context-sensitive grammar G such that $L = L(G)$.

Definition

The language L is context-free, if there exists a context-free grammar G such that $L = L(G)$.

Definition

The language L is regular, if there exists a regular grammar G such that $L = L(G)$.

Empty word theorem – introduction

It is obvious that context-sensitive grammars are unrestricted as well, because each generative grammar is unrestricted. It is also obvious that regular grammars are context-free as well, because in regular grammars the left hand side of each rule is a nonterminal, which is the only condition to be satisfied for a grammar in order to be context-free. Let us investigate the case of the context-free and context sensitive grammars.

Empty word theorem – introduction

Example

Let the grammar G be

$G = (\{S, A\}, \{x, y\}, S, P)$
$P = \{$
$S \to AxA,$
$A \to SyS,$
$A \to \lambda,$
$S \to \lambda$
$\}.$

This grammar is context-free, because the left hand side of each rule contains one nonterminal. At the same time, this grammar is not context-sensitive, because

- *the rule $A \to \lambda$ is not allowed in context-sensitive grammars,*
- *if $S \to \lambda \in P$, then the rule $A \to SyS$ is not allowed.*

Empty word theorem – introduction

This example shows that there are context-free grammars which are not context-sensitive. Although this statement holds for grammars, we can show that in the case of languages the Chomsky hierarchy is a real hierarchy, because each context-free language is context-sensitive as well. To prove this statement, let us consider the following theorem.

Empty word theorem + proof

Theorem (Empty word theorem)

For each context-free grammar G we can give context-free grammar G', which is context-sensitive as well, such that $L(G) = L(G')$.

Proof. We give a constructive proof of this theorem. We are going to show the necessary steps to receive an equivalent context-sensitive grammar G' for a context-free grammar $G = (N, T, S, P)$.

1. First, we have to collect each nonterminal symbol from which the empty word can be derived. To do this, let the set $U(1)$ be $U(1) = \{A \mid A \in N,\ A \to \lambda \in P\}$. Then let $U(i) = U(i-1) \cup \{A \mid A \to B_1 B_2 ... B_n \in P,\ B_1, B_2, ..., B_n \in U(i-1)\}$. Finally, we have an integer i such that $U(i) = U(i-1) = U$ which contains all of the nonterminal symbols from which the empty word can be derived.

Empty word theorem + proof

2. Second, we are going to create a new set of rules. The right hand side of these rules should not contain the empty word. Let $P' = (P \cup P_1) \backslash \{A \to \lambda \mid A \in N\}$. The set P_1 contains production rules as well. $B \to p \in P_1$ if $B \to q \in P$ and we get p from q by removing some letters contained in set U.
3. Finally, if $S \notin U$, then $G' = (N, T, S, P')$. If set U contains the start symbol, then the empty word can be derived from S, and $\lambda \in L(G)$. In this case, we have to add a new start symbol to the set of nonterminals, and we also have to add two new productions to the set P_1, and $G' = (N \cup \{S'\}, T, S', P' \cup \{S' \to \lambda,\ S' \to S\})$, so G' generates the empty word, and it is context sensitive.

QED. (QED is an initialism of the Latin phrase "quod erat demonstrandum". Traditionally, the abbreviation is placed at the end of a mathematical proof to indicate that the proof or argument is complete.)

Empty word theorem – example

Example

Let the context-free grammar G be

$G = (\{S, A, B\}, \{x, y\}, S, P)$
$P = \{$
$S \to ASxB,$
$S \to AA,$
$A \to \lambda,$
$B \to SyA$
$\}.$

Empty word theorem – example

Example

Now, we create a context-sensitive generative grammar G', such that $L(G') = L(G)$.

1. $U(1) = \{A\}$,
   $U(2) = \{A, S\} = U$.
2. $P' = \{$
   $S \to ASxB,\ S \to SxB,\ S \to AxB,\ S \to xB,$
   $S \to AA,\ S \to A,$
   $B \to SyA,\ B \to yA,\ B \to Sy,\ B \to y$
   $\}$.
3. $G' = (\{S, A, B, S'\}, \{x, y\}, S', P' \cup \{S' \to \lambda,\ S' \to S\})$.

Regular normal form

First of all, we are going to deal with the most simple class of the Chomsky hierarchy, the class of regular languages. In order to be comprehensive we present the definition of regular grammars here again.

Definition (Regular grammars)

A grammar $G = (N, T, S, P)$ is regular if each of its productions has one of the following forms: $A \to u, A \to uB$, where $A, B \in N, u \in T^*$. The languages that can be generated by regular grammars are the regular languages (they are also called type 3 languages of the Chomsky hierarchy).

Regular normal form – definition

We note here that the grammars and languages of the definition above are commonly referred to as right-linear grammars and languages, and regular grammars and languages are defined in a more restricted way:

Definition (Alternative definition of regular grammars)

A grammar $G = (N, T, S, P)$ is regular if each of its productions has one of the following forms: $A \to a, A \to aB, S \to \lambda$, where $A, B \in N, a \in T$. The languages that can be generated by these grammars are the regular languages.

This alternative definition is usually called as **regular normal form**. Now we show that the two definitions are equivalent in the sense that they define the same class of languages.

Regular normal form – theorem + proof

Theorem

The language classes defined by our original definition and by the alternative definition coincide.

Proof. The proof consists of two parts: we need to show that languages generated by grammars of one definition can also be generated by grammars of the other definition.
It is clear the grammars satisfying the alternative definition satisfy our original definition at the same time, therefore, every language that can be generated by the grammars of the alternative definition can also be generated by grammars of the original definition.
Let us consider the other direction. Let a grammar $G = (N, T, S, P)$ may contain rules of types $A \to u$ and $A \to uB$ $(A, B \in N, u \in T^*)$. Then, we give a grammar $G' = (N', T, S', P')$ such that it may only contain rules of the forms $A \to a$, $A \to aB$, $S' \to \lambda$ (where $A, B \in N', a \in T$) and it generates the same language as G, i.e., $L(G) = L(G')$.

Regular normal form – theorem + proof

First we obtain a grammar G'' such that $L(G) = L(G'')$ and $G'' = (N'', T, S, P'')$ may contain only rules of the following forms: $A \to a, A \to aB, A \to B, A \to \lambda$ (where $A, B \in N'', a \in T$). Let us check every rule in P: if it has one of the forms above, then we copy it to the set P'', else we will do the following:

- if the rule is of the form $A \to a_1 \dots a_k$ for $k > 1$, where $a_i \in T, i \in \{1, \dots, k\}$, then let the new nonterminals $X_1, \dots, X_{k-1}$ be introduced (new set for every rule) and added to the set N'' and instead of the actual rule the next set of rules is added to P'': $A \to a_1X_1, X_1 \to a_2X_2, \dots, X_{k-2} \to a_{k-1}X_{k-1}, X_{k-1} \to a_k$.
- if the rule is of the form $A \to a_1 \dots a_kB$ for $k > 1$, where $a_i \in T, i \in \{1, \dots, k\}$, then let the new nonterminals $X_1, \dots, X_{k-1}$ be introduced (new set for every rule) and put to the set N'', and instead of the actual rule the next set of rules is added to P'': $A \to a_1X_1, X_1 \to a_2X_2, \dots, X_{k-2} \to a_{k-1}X_{k-1}, X_{k-1} \to a_kB$.

When every rule is analyzed (and possibly replaced by a set of rules) we have grammar G''. It is clear that the set of productions P'' of G'' may contain only rules of the forms $A \to a, A \to aB, A \to B, A \to \lambda$ (where $A, B \in N'', a \in T$).

Regular normal form – theorem + proof

Now we may have some rules in P'' that do not satisfy the alternative definition. The form of these rules can only be $A \to B$ and $C \to \lambda$ (where $A, B, C \in N'', C \neq S$). The latter types of rules can be eliminated by the Empty-word theorem. Therefore, we can assume that we have a grammar $G''' = (N''', T, S', P''')$ such that $L(G''') = L(G)$ and P''' may contain only rules of the forms $A \to aB, A \to B, A \to a, S' \to \lambda$ (where $A, B \in N''', a \in T$ and in case $S' \to \lambda \in P'''$ the start symbol S' does not occur on the right hand side of any of the rules of P'''). Let us define the following sets of nonterminals:

- let $U_1(A) = \{A\}$,
- let $U_{i+1}(A) = U_i(A) \cup \{B \in N''' \mid \exists C \in U_i(A)$ such that $C \to B \in P'''\}$, for $i > 1$.

Regular normal form – theorem + proof

Since N''' is finite there is a minimal index k such that $U_k(A) = U_{k+1}(A)$. Let $U(A)$ denote the set $U_k(A)$ with the above property. In this way, $U(A)$ contains exactly those nonterminals that can be derived from A by using rules only of the form $B \to C$ (where $B, C \in N'''$). We need to replace the parts $A \Rightarrow^* B \Rightarrow r$ of the derivations in G''' by a direct derivation step in our new grammar, therefore, let $G' = (N''', T, S', P')$, where $P' = \{A \to r \mid \exists B \in N''' \text{ such that } B \to r \in P''', r \notin N''' \text{ and } B \in U(A)\}$. Then G' fulfills the alternative definition, moreover, it generates the same language as G''' and G.
QED.

Examples: book + practice.

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.