Einführung in die Computerlinguistik

Kontextfreie Grammatiken - Formale Eigenschaften

Laura Kallmeyer
Heinrich-Heine-Universität Düsseldorf

Sommersemester 2013

Overview

1. Normalformen
2. Abgeschlossenheitseigenschaften
3. Pumping Lemma

[Hopcroft and Ullman, 1979]

Normal forms (1)

A normal form of a grammar formalism F is a further restriction on the grammars in F that does not affect the set of generated string languages.

Let $G = \langle N, T, P, S\rangle$ be a CFG. A $X \in N \cup T$ is called

- useful if there is a derivation $S \stackrel{*}{\Rightarrow} \alpha X \beta \stackrel{*}{\Rightarrow} w$ with $w \in T^*$.
- useless otherwise.

For each CFG, there exists an equivalent CFG (= a CFG generating the same string language) without useless symbols.

Normal forms (2)

To eliminate all useless symbols two things need to be done:

1. All $X \in N$ need to be eliminated that cannot lead to a terminal sequence.

   This can be done recursively: Starting from the terminals and following the productions from right to left, the set of all symbols leading to terminals can be computed recursively.

   Productions containing symbols that are not in this set are eliminated.
2. In the resulting CFG, the unreachable symbols need to be eliminated.

   This is done starting from S and applying productions. Each time, the symbols from the right-hand sides are added.

   Again, productions containing non-terminals or terminals that are not in the set are eliminated.

Normal Forms (3)

A production of the form $A \to \epsilon$ is called a ϵ-production.

The following holds:

For each CFG G, there is a CFG G' without ϵ-productions such that $L(G') = L(G) \setminus \{\epsilon\}$.

Normal Forms (4)

In order to eliminate ϵ-productions, we

- compute the set $N_\epsilon = \{A \mid A \stackrel{*}{\Rightarrow} \epsilon\}$ recursively:
  1. $N_\epsilon := \{A \in N \mid A \Rightarrow \epsilon\}$.
  2. For all A with $A \to \alpha$, $\alpha \in N_\epsilon^*$: add A to N_ϵ.
  3. Repeat 2. until N_ϵ does not change any more.
- delete the ϵ-productions and for each $A \to X_1 \dots X_n$: add all productions one can obtain by deleting some $X \in N_\epsilon$ from the right-hand side as long as one does not delete all $X_1, \dots, X_n$.

Normal Forms (5)

A production of the form $A \to B$ is called a unary production.

For each CFL that does not contain ϵ, a CFG without unary productions can be found.

Elimination of unary productions for a CFG without ϵ-productions:

- For all $A \stackrel{*}{\Rightarrow} B$ and all $B \to \beta, \beta \notin N$: add $A \to \beta$.
- Delete all unary productions.

Normal Forms (6)

There are two important normal forms for CFGs: A CFG for a language without ϵ is

- in Chomsky normal form iff all productions have either the form $A \to BC$ or $A \to a$ with $A, B, C \in N, a \in T$.
- in Greibach normal form iff all productions have the form $A \to a\alpha$ with $a \in T, \alpha \in N^*$.

Normal Forms (7)

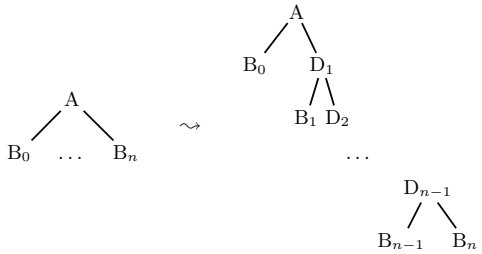
For each CFL L without ϵ, there is a CFG G in Chomsky normal form with $L = L(G)$.

Construction of an equivalent CFG in CNF for a given CFG (after elimination of useless symbols, ϵ-productions and unary productions):

1. For each terminal a: introduce new non-terminal C_a, replace a with C_a in all right-hand sides of length > 1 and add production $C_a \to a$.

Normal Forms (8)

2. For each production $A \to B_0 \dots B_n$ introduce new non-terminals $D_1, \dots, D_{n-1}$ and replace production with productions
   $A \to B_0 D_1$, $D_1 \to B_1 D_2$, $D_2 \to B_2 D_3$, $\dots$, $D_{n-1} \to B_{n-1} B_n$.



Normal Forms (9)

For each CFL L without ϵ, there is a CFG G in Greibach normal form with $L = L(G)$.

For the construction see [Hopcroft and Ullman, 1994].

Closure Properties (1)

CFLs are closed

- under union (construction: add $S' \to S_1 | S_2$ where S_1, S_2 the start symbols of the two CFGs; condition: non-terminals in the two CFGs pairwise disjoint)
- under concatenation and Kleene closure (construction as with regular languages)
- under homomorphisms (construction: replace terminals in productions with their images under the homomorphism)
- under substitution (construction: replace terminals in first CFG with start symbols of corresponding CFGs, make sure non-terminals of the involved CFGs are pairwise disjoint)

Closure Properties (2)

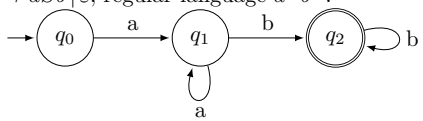
CFLs are closed under intersection with regular languages (construction: take the CFG and the DFA of the regular language; then build a new CFG by replacing non-terminals A with triples $\langle q_1, A, q_2\rangle$ where the triple stands for derivation of the yield of A while traversing a DFA path from q_1 to q_2)

Closure Properties (3)

Example for intersection with regular languages:

CFG: $S \to aSb \,|\, \varepsilon$, regular language a^+b^+.

DFA:



Intersection grammar, start symbol S':

$S' \to \langle q_0, S, q_2\rangle$	(q_2 is the only final state)
$\langle q_0, S, q_2\rangle \to a\langle q_1, S, q_1\rangle b$	(with $\delta(q_0, a) = q_1, \delta(q_1, b) = q_2$)
$\langle q_0, S, q_2\rangle \to a\langle q_1, S, q_2\rangle b$	(with $\delta(q_0, a) = q_1, \delta(q_2, b) = q_2$)
$\langle q_1, S, q_2\rangle \to a\langle q_1, S, q_2\rangle b$	(with $\delta(q_1, a) = q_1, \delta(q_2, b) = q_2$)
$\langle q_1, S, q_2\rangle \to a\langle q_1, S, q_1\rangle b$	(with $\delta(q_1, a) = q_1, \delta(q_1, b) = q_2$)
$\langle q_1, S, q_1\rangle \to \varepsilon$	(since $q_1 = q_1$)

Pumping Lemma (1)

- In a context-free derivation, the expansion of a non-terminal A does not depend on the context A occurs in.
- Consequently, if we have a derivation

$$S \stackrel{+}{\Rightarrow} xAz \stackrel{+}{\Rightarrow} xv_1Av_2z \stackrel{+}{\Rightarrow} xv_1yv_2z$$

then the part $A \stackrel{+}{\Rightarrow} v_1Av_2$ of the derivation can be iterated, i.e., we can also have

$$S \stackrel{+}{\Rightarrow} xv_1^iyv_2^iz$$

for any $i \geq 1$.

Pumping Lemma (2)

Looking at the derivation trees, this is even clearer:

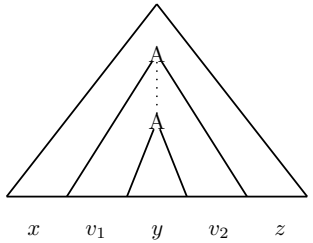
Assume that in a derivation tree, if the derivation tree has a certain minimal height (maximal length of paths from root to leaves), we have a path from the root (symbol S) to a leaf such that

- on this path, a non-terminal A occurs twice, and
- below the higher of these As, there is only a single A and no other non-terminal is repeated on any path.

Since the number of non-terminals is finite, from a certain string length on, every derivation treee of a word in the language is necessarily of this form.

Pumping Lemma (3)

The part of the derivation tree in between the two nodes with the same non-terminal can be iterated. This means that the strings yielded by this part are pumped.



Pumping Lemma (4)

Pumping lemma for context-free languages: Let L be a context-free language. Then there is a constant k such that for all $w \in L$ with $|w| \geq k$: $w = xv_1yv_2z$ with

- $|v_1v_2| \geq 1$,
- $|v_1yv_2| \leq k$, and
- for all $i \geq 0$: $xv_1^iyv_2^iz \in L$.

Pumping Lemma (5)

With the pumping lemma and the closure properties, we can show for a lot of languages that they are not context-free:

$L_1 = \{a^nb^nc^n \mid n \geq 1\}$ is not context-free.

Proof: Assume that L_1 is context-free. Then it must satisfy the pumping lemma with some constant k. Consequently, for every $w \in L_1$ and then in particular for $a^kb^kc^k$, we must find substrings v_1, v_2 that can be iterated. Either they contain each only occurrences of a single terminal. Then the iteration will yield words that have no longer the same numbers of as, bs and cs. Or at least one contains at least two different terminals. Then the iterations necessarily lead to words where the as, bs and cs get mixed.

$\Rightarrow$ L_1 does not satisfy the pumping lemma, contrary to the assumption and therefore L_1 cannot be context-free.

Pumping Lemma (6)

$L_2 = \{w \mid w \in \{a, b, c\}^*, |w|_a = |w|_b = |w|_c\}$ is not context-free.

Proof: Assume that L_2 is context-free. Then its intersection with the regular language $a^+b^+c^+$ must also be context-free. However, this intersection is $L_1 = \{a^nb^nc^n \mid n \geq 1\}$, for which we have just shown that it is not context-free.

$\Rightarrow$ Since L_1 is not context-free, our assmption is false and L_2 is not context-free either.

References

[Hopcroft and Ullman, 1979] Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages and Computation*. Addison Wesley.

[Hopcroft and Ullman, 1994] Hopcroft, J. E. and Ullman, J. D. (1994). *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*. Addison Wesley, 3. edition.