Ambiguity, closure properties, and Chomsky's normal form

Overview of ambiguity, and Chomsky's standard form

The notion of ambiguity

Convention: we assume that in every substitution, the **leftmost remaining variable** is the one that **is replaced**. When this convention is applied to all substitutions in a derivation, we talk about **leftmost derivations**

Definition: A CGF is said to be **ambiguous if a string** in its language **has two different leftmost derivations**.

- In such a case, the string has necessarily two different parsing trees

Equivalence and ambiguity

Definition: Two context-free grammars are declared to be **equivalent** if and only if they produce the same context-free language.

- Remark: An ambiguous context-free grammar **may have** an equivalent unambiguous context-free grammar.

Definition: Given an ambiguous grammar, the process of identifying an equivalent grammar that is not ambiguous is called **disambiguation**. If not such grammar exists, the language is called **inherently ambiguous**

Example of an ambiguous context-free grammar

Let $G = (\{S, A\}, \{0,1\}, R, S)$

$R = \{S \to AS \mid \lambda, A \to A1 \mid 0A1 \mid 01\}$

Then, the string $w = 00111 \in L(G)$ has the following two leftmost derivations:

1. $S \to AS \to 0A1S \to 0A11S$

   $\to 00111\,S \to 00111$

2. $S \to AS \to A1S \to 0A11S$

   $\to 00111\,S \to 00111$

The grammar is not inherently ambiguous

The ambiguity stems from the fact that the rule

$$A \rightarrow A1$$

can be use in the first or in the second substitution indistinctively. In either case the same string $w = 00111 \in L(G)$ is derived.

– However, $L(G)$ is **not inherently ambiguous**. Following is a context free grammar that eliminates the above ambiguity.

Disambiguation

Let $G_1 = (\{S, A, B\}, \{0,1\}, R_1, S)$

$R_1 = \{S \to AS \mid \lambda, A \to 01A \mid B, B \to B1 \mid 01\}$

Claim: $L(G) = L(G_1)$ and under G_1 each string is derived by a unique sequence of leftmost substitutions. This is, the new grammar derives the language unambiguously

Proof: exercise!

Closure properties

Theorem 9.1: Context-free languages **are closed under regular operations**

Scketch of the proof: Let L_1 and L_2 be context-free languages. Let $G_1 = (V_1, A, R_1, S_1)$ and $G_2 = (V_2, A, R_2, S_2)$ be context-free grammars such that $L_1 = L(G_1)$ and $L_2 = L(G_2)$ respectively. We may assume without loss of generality that $V_1 \cap V_2 = \phi$.

Proof (cont.)

Define the context-free grammars:

1. $U = (\{S\} \bigcup V_1 \bigcup V_2, A, R_1 \bigcup R_2 \bigcup \{S \to S_1 \mid S_2\}, S)$
2. $C = (\{S\} \bigcup V_1 \bigcup V_2, A, R_1 \bigcup R_2 \bigcup \{S \to S_1 S_2\}, S)$
3. $T = (\{S\} \bigcup V_1, A, R_1 \bigcup \{S \to \lambda \mid S_1 S\}, S)$

Here we assume $S \notin V_1 \bigcup V_2$

<u>Claim</u>:

$$L(U) = L_1 \bigcup L_2 \wedge L(C) = L_1 L_2 \wedge L(T) = L_1 *$$

The rest of the proof (*i.e.* verification of the claim) is left as an <u>exercise</u>.

Applications

1. The language $M = \{0^{n_1}1^{n_1}0^{n_2}1^{n_2} : n_1, n_2 \geq 0\}$ is context-free, since it is the concatenation of $L = \{0^n1^n : n \geq 0\}$ with itself

2. The language

   $S = \{0^{n_1}1^{n_1} \cdots 0^{n_k}1^{n_k} : k \in N \wedge (\forall i = 1,...,k) n_i \geq 0\}$

   is also context-free, since it is the star-closure of *L*

Chomsky normal form

Theorem 9.2: (Chomsky normal form) Any context-free grammar is equivalent to a grammar $G = (V, A, R, S)$ whose productions are either of the form:

$Z \rightarrow UT$ where $Z, U, T \in V \wedge U \neq S \wedge T \neq S$; or

$Z \rightarrow a$ where $a \in A$; or

$S \rightarrow \lambda$

Method 9.1

Transforming a context-free grammar into a grammar in Chomsky normal form

Given a context-free grammar *G*=(*V*,*A*,*R*,*S*)

1. Add a new start variable S_0 and add the rule $S_0 \rightarrow S$
2. Remove all rules of the form $u \rightarrow \lambda, u \in V$ adding the rules that are necessary to preserve the grammar's productions
3. Eliminate unit rules. These are rules of the form of

$u \rightarrow v, u, v \in V$ and their elimination requires adding the rules $u \rightarrow w$ wherever $v \rightarrow w$ appears, unless $u \rightarrow w$ was already eliminated

4. Convert rules of the form $u \rightarrow a_1 a_2 a_3 ... a_k, k \geq 3$ to a set of rules of the form $u \rightarrow a_1 u_1, u_1 \rightarrow a_2 u_2, ..., u_{k-2} \rightarrow a_{k-1} a_k$ where the u_i's are new variables.
5. Convert rules of the form $u \rightarrow a_1 ... a_k, k \geq 2$ to rules of the form of $u_i \rightarrow a_i$, wherever a_i is a terminal. Here, u_i is a new variable.

Example

The context free grammar:

$$G = (\{S\}, \{0,1\}, R, S)$$

$$R = \{S \to 0S1 \mid \lambda\}$$

is **not** in Chomsky normal form. We will find an equivalent context-free grammar *N,* which is in Chomsky normal form, using the previous procedure (Method 9.1)

Example (cont.)

1. Define a new start variable

$$S_0 \rightarrow S$$

$$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow \lambda \end{array}\right\} \text{Old productions}$$

Example (cont.)

2. Eliminate unitary productions. In our case we have:

$$S_0 \to S \wedge S \to \lambda$$

Recall that the removal is performed by **eliminating them** and **adding all rules necessary to preserve the original grammar's derivations**

Example (cont.)

The derivations that involve these unitary productions (and have to be preserved) are:

$$S_0 \to S \to 0S1 \Rightarrow S_0 \to 0S1$$

$$S_0 \to S \to 01 \Rightarrow S_0 \to 01$$

Thus, the **new set of rules** is:

$$S_0 \to \lambda \mid 0S1 \mid 01$$

$$S \to 0S1$$

$$S \to 01$$

Example (cont.)

The latter set of rules define an equivalent grammar which **is not yet in Chomsky normal form**. Indeed **no rule**, except the one that associate the starting variable with the null string, is in Chomsky normal form. We'll fix this in the next step:

3. Decomposing all remaining productions that are not in Chomsky normal form into **sets of productions** with **two variables** or **a terminal on the right hand side**.

Example (cont.)

This applies to the current productions:

$$S_0 \to 01,$$

$$S \to 0S1,$$

$$S \to 01$$

$$S_0 \to 0S1$$

We transform each of them into a set of productions in Chomsky normal form as follows

Example (cont.)

Consider first: $S \to 01$

This production is clearly equivalent to the combined action of the following productions:

$$S \to 01 \Leftrightarrow$$

$$S \to S_1 S_2 \wedge$$

$$S_1 \to 0 \wedge S_2 \to 1$$

Example (cont.)

As for $S \to 0S1$ we have:

$$S \to 0S1 \Leftrightarrow$$

$$S \to 0S_3 \wedge S_3 \to S1 \Leftrightarrow$$

$$(S \to S_4 S_3 \wedge S_4 \to 0) \wedge$$

$$(S_3 \to SS_5 \wedge S_5 \to 1)$$

Since there are here productions that were already defined, we replace the variables:

$$S_4 \coloneqq S_1 \wedge S_5 \coloneqq S_2$$

Example (cont.)

After replacing we get:

$$S \rightarrow S_1 S_3 \wedge S_3 \rightarrow S S_2$$

Example (cont.)

And similarly, using the already defined productions, we transform

$$S_0 \rightarrow 01 \Leftrightarrow$$

$$S_0 \rightarrow S_1 S_2 \wedge S_1 \rightarrow 0 \wedge S_2 \rightarrow 1$$

and $S_0 \rightarrow 0S1 \Leftrightarrow$

$$S_0 \rightarrow S_1 S_3 \wedge S_3 \rightarrow SS_2$$

Example (cont.)

In summary, we get:

$$N = (\{S_0, S, S_1, S_2, S_3\}, \{0,1\}, R, S_0)$$

$$R = \{S_0 \to \lambda \mid S_1S_2 \mid S_1S_3,$$

$$S \to S_1S_2 \mid S_1S_3,$$

$$S_1 \to 0, S_2 \to 1,$$

$$S_3 \to SS_2\}$$

Does it work?

Let's derive a few strings:

$$S_0 \to \lambda$$

$$S_0 \to S_1S_2 \to 0S_2 \to 01$$

$$S_0 \to S_1S_3 \to 0S_3 \to 0SS_2$$

$$\to 0S_1S_2S_2 \to 00S_2S_2 \to 001S_2 \to 0011$$

What's so important about Chomsky's normal form?

Theorem 9.3: A context-free grammar in Chomsky normal form derives a string of length *n* in **exactly** *2n-1* substitutions

Proof:

Let G be a context - free grammar in Chomsky normal form and let $w = w_1 \cdots w_n \in L(G)$. Since each w_i is a terminal, it could only be produced by substituting a rule of the form $v_i \to w_i$ in a string of variables of the form of $v_1 \cdots v_n$. Thus, the derivation of w involves n applications of such rules. But, the formation of $v_1 \cdots v_n$ can only be done by applying rules of the form $u \to zv, u, z, v$ variables. There are exactly $n-1$ applications of such rules involved in the generation of $v_1 \cdots v_n$.

A note on context-sensitive grammars

<u>Definition:</u> A **context-sensitive** grammar is a quadruple

$$G = (V, A, R, S)$$

where V, A and S are just as in a context-free grammar, but the set of rules R includes rules of the form $aUb \to cVd$, where U and V are variables, and $a, b, c,$ and d are strings of variables and literals.

Example

Let G=({S,R,T,U,V,W},{a, b, c}, *R*, S)

R= {S → aRc,

R → aRT | b, bTc → bbcc,

bTT → bbUT, UT → UU,

UUc → VUc → Vcc,

UV → VV, bVc → bbcc,

bVV → bbWV, WV → WW,

WWc → TWc → Tcc, WT → TT }

This grammar generates the canonical non-context-free language: $\{a^n b^n c^n : n \geq 0\}$

A derivation

The derivation of the string

aaabbbccc

is:

S → aRc → aaRTc → aaaRTTc → aaabTTc → aaabbUTc → aaabbUUc → aaabbVUc → aaabbVcc → aaabbbccc

Note on normal forms

Principle: Every **context-sensitive** grammar which does not generate the empty string can be transformed into an equivalent grammar in **Kuroda normal form**

Remark: The normal form **will not** in general be a context-sensitive grammar, but will be a **non-contracting grammar**