

Section 4.9: Chomsky Normal Form

In this section, we study a special form of grammars called Chomsky Normal Form (CNF). CNF was invented by, and named after, the linguist Noam Chomsky. Grammars in CNF have very nice formal properties. In particular, valid parse trees for grammars in CNF are very close to being binary trees.

Copyright © 2003–4 Alley Stoughton

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

The \LaTeX source of these slides, the associated book, and the distribution of the Forlan toolset are available on the WWW at <http://people.cis.ksu.edu/~stough/forlan/>.

(4.9) Introduction (Cont.)

Any grammar that doesn't generate $\%_0$ can be put in CNF. And, if G is a grammar that does generate $\%_0$, it can be turned into a grammar in CNF that generates $L(G) - \{\%_0\}$. In the next section, we will use this fact when proving the pumping lemma for context-free languages, a method for showing the certain languages are not context-free.

When converting a grammar to CNF, we will first get rid of productions of the form $q \rightarrow \%_0$ and $q \rightarrow r$, where q and r are variables.

(4.9) Removing ϵ -Productions

A ϵ -*production* is a production of the form $q \rightarrow \epsilon$. We will show by example how to turn a grammar G into a simplified grammar with no ϵ -productions that generates $L(G) - \{\epsilon\}$.

Suppose G is the grammar

$$A \rightarrow 0A1 \mid BB,$$

$$B \rightarrow \epsilon \mid 2B.$$

First, we determine which variables q are *nullable* in the sense that $\epsilon \in \Pi_q$, i.e., that ϵ is the yield of a valid parse tree for G whose root label is q . Clearly,

(4.9) Removing ϵ -Productions

A ϵ -production is a production of the form $q \rightarrow \epsilon$. We will show by example how to turn a grammar G into a simplified grammar with no ϵ -productions that generates $L(G) - \{\epsilon\}$.

Suppose G is the grammar

$$A \rightarrow 0A1 \mid BB,$$

$$B \rightarrow \epsilon \mid 2B.$$

First, we determine which variables q are *nullable* in the sense that $\epsilon \in \Pi_q$, i.e., that ϵ is the yield of a valid parse tree for G whose root label is q . Clearly, B is nullable. And, since $A \rightarrow BB \in P_G$, it follows that A is nullable.

(4.9) Removing ϵ -Productions (Cont.)

Since A is nullable, we replace the production $A \rightarrow \epsilon$ with the productions

(4.9) Removing ϵ -Productions (Cont.)

Since A is nullable, we replace the production $A \rightarrow 0A1$ with the productions $A \rightarrow 0A1$ and $A \rightarrow 01$. The idea is that this second production will make up for the fact that A won't be nullable in the new grammar.

Since B is nullable, we replace the production $A \rightarrow BB$ with the productions

(4.9) Removing ϵ -Productions (Cont.)

Since A is nullable, we replace the production $A \rightarrow \epsilon A 1$ with the productions $A \rightarrow 0A1$ and $A \rightarrow 01$. The idea is that this second production will make up for the fact that A won't be nullable in the new grammar.

Since B is nullable, we replace the production $A \rightarrow BB$ with the productions $A \rightarrow BB$ and $A \rightarrow B$ (the result of deleting either one of the B 's).

The production $B \rightarrow \epsilon$ is deleted.

Since B is nullable, we replace the production $B \rightarrow 2B$ with the productions

(4.9) Removing ϵ -Productions (Cont.)

Since A is nullable, we replace the production $A \rightarrow 0A1$ with the productions $A \rightarrow 0A1$ and $A \rightarrow 01$. The idea is that this second production will make up for the fact that A won't be nullable in the new grammar.

Since B is nullable, we replace the production $A \rightarrow BB$ with the productions $A \rightarrow BB$ and $A \rightarrow B$ (the result of deleting either one of the B 's).

The production $B \rightarrow \epsilon$ is deleted.

Since B is nullable, we replace the production $B \rightarrow 2B$ with the productions $B \rightarrow 2B$ and $B \rightarrow 2$.

(4.9) Removing ϵ -Productions (Cont.)

This give us the grammar

$$A \rightarrow 0A1 \mid 01 \mid BB \mid B,$$

$$B \rightarrow 2B \mid 2.$$

In general, we finish by simplifying our new grammar. The new grammar of our example is already simplified, however.

(4.9) Removing Unit Productions

A *unit production* for a grammar G is a production of the form $q \rightarrow r$, where r is a variable (possibly equal to q). We now show by example how to turn a grammar G into a simplified grammar with no ϵ -productions or unit productions that generates $L(G) - \{\epsilon\}$.

Suppose G is the grammar

$$\begin{aligned} A &\rightarrow 0A1 \mid 01 \mid BB \mid B, \\ B &\rightarrow 2B \mid 2. \end{aligned}$$

We begin by applying our algorithm for removing ϵ -productions to our grammar; the algorithm has no effect in this case.

(4.9) Removing Unit Productions (Cont.)

Next, we generate the productions of a new grammar as follows. If

- q and r are variables of G ,
- there is a valid parse tree for G whose root label is q and yield is r ,
- $r \rightarrow w$ is a production of G , and
- w is not a single variable of G ,

then we add $q \rightarrow w$ to the set of productions of our new grammar.

(Determining whether there is a valid parse whose root label is q and yield is r is easy, since we are working with a grammar with no ϵ -productions.)

This process results in the grammar

(4.9) Removing Unit Productions (Cont.)

Next, we generate the productions of a new grammar as follows. If

- q and r are variables of G ,
- there is a valid parse tree for G whose root label is q and yield is r ,
- $r \rightarrow w$ is a production of G , and
- w is not a single variable of G ,

then we add $q \rightarrow w$ to the set of productions of our new grammar.

(Determining whether there is a valid parse whose root label is q and yield is r is easy, since we are working with a grammar with no ϵ -productions.)

This process results in the grammar

$$A \rightarrow 0A1 \mid 01 \mid BB \mid 2B \mid 2,$$

$$B \rightarrow 2B \mid 2.$$

Finally, we simplify our grammar, which has no effect in this case.

(4.9) Chomsky Normal Form

A grammar G is in *Chomsky Normal Form* (CNF) iff each of its productions has one of the following forms:

- $q \rightarrow a$, where a is not a variable;
- $q \rightarrow pr$, where p and r are variables.

We explain by example how a grammar G can be turned into a simplified grammar in CNF that generates $L(G) - \{\epsilon\}$.

Suppose G is the grammar

$$A \rightarrow 0A1 \mid 01 \mid BB \mid 2B \mid 2,$$

$$B \rightarrow 2B \mid 2.$$

We begin by applying our algorithm for removing ϵ -productions and unit productions to this grammar. In this case, it has no effect.

(4.9) Conversion into CNF (Cont.)

Since the productions $A \rightarrow BB$, $A \rightarrow 2$ and $B \rightarrow 2$ are legal CNF productions, we simply transfer them to our new grammar.

Next we add the variables $\langle 0 \rangle$, $\langle 1 \rangle$ and $\langle 2 \rangle$ to our grammar, along with the productions

$$\langle 0 \rangle \rightarrow 0, \quad \langle 1 \rangle \rightarrow 1, \quad \langle 2 \rangle \rightarrow 2.$$

Now, we can replace the production $A \rightarrow 01$ with $A \rightarrow \langle 0 \rangle \langle 1 \rangle$. We can replace the production $A \rightarrow 2B$ with $A \rightarrow \langle 2 \rangle B$. And, we can replace the production $B \rightarrow 2B$ with the production $B \rightarrow \langle 2 \rangle B$.

Finally, we replace the production $A \rightarrow 0A1$ with the productions

$$A \rightarrow \langle 0 \rangle C, \quad C \rightarrow A \langle 1 \rangle,$$

and add C to the set of variables of our new grammar.

(4.9) Conversion into CNF (Cont.)

Summarizing, our new grammar is

$$A \rightarrow BB \mid 2 \mid \langle 0 \rangle \langle 1 \rangle \mid \langle 2 \rangle B \mid \langle 0 \rangle C,$$

$$B \rightarrow 2 \mid \langle 2 \rangle B,$$

$$\langle 0 \rangle \rightarrow 0,$$

$$\langle 1 \rangle \rightarrow 1,$$

$$\langle 2 \rangle \rightarrow 2,$$

$$C \rightarrow A \langle 1 \rangle.$$

The official version of our algorithm names variables in a different way.

(4.9) CNF in Forlan

The Forlan module `Gram` defines the following functions:

```
val removeEmptyProductions      : gram -> gram
val removeEmptyAndUnitProductions : gram -> gram
val chomskyNormalForm           : gram -> gram
```

Suppose `gram` of type `gram` is bound to the grammar with variables `A` and `B`, start variable `A`, and productions

$$\begin{aligned} A &\rightarrow 0A1 \mid BB, \\ B &\rightarrow \% \mid 2B. \end{aligned}$$

(4.9) CNF in Forlan (Cont.)

Here is how Forlan can be used to turn this grammar into a CNF grammar that generates the nonempty strings that are generated by `gram`:

```
- val gram' = Gram.chomskyNormalForm gram;
val gram' = - : gram
- Gram.output("", gram');
{variables}
<1,A>, <1,B>, <2,0>, <2,1>, <2,2>, <3,A1>
{start variable}
<1,A>
{productions}
<1,A> ->
2 | <1,B><1,B> | <2,0><2,1> | <2,0><3,A1> | <2,2><1,B>;
<1,B> -> 2 | <2,2><1,B>; <2,0> -> 0; <2,1> -> 1;
<2,2> -> 2; <3,A1> -> <1,A><2,1>
val it = () : unit
```

(4.9) CNF in Forlan (Cont.)

```
- val gram'' = Gram.renameVariablesCanonically gram';  
val gram'' = - : gram  
- Gram.output("", gram'');  
{variables}  
A, B, C, D, E, F  
{start variable}  
A  
{productions}  
A -> 2 | BB | CD | CF | EB; B -> 2 | EB; C -> 0; D -> 1;  
E -> 2; F -> AD  
val it = () : unit
```