

Derivations

Informatics 2A: Lecture 4

Bonnie Webber

School of Informatics
University of Edinburgh
bonnie@inf.ed.ac.uk

2 October 2009

Review

A **derivation** is the sequence of strings over V produced by a sequence of PS rule applications, starting from a start symbol Σ .

In a **phrase structure grammar** (either **context-free** or **context-sensitive**), only **one** symbol is rewritten at each step in a derivation.

$S \Rightarrow NP VP \Rightarrow NP \text{ verb } NP \Rightarrow NP \text{ verb the book} \Rightarrow NP \text{ took the book} \Rightarrow \text{the man took the book}$

We distinguish those symbols that can be re-written (**non-terminal symbols**) from those that cannot (**terminal symbols**), and take **sentences** of a language to be strings of terminal symbols.

- 1 Context-Free Grammars
 - Review
 - Derivations
 - Tree Diagrams
 - Non-Equivalent Derivations
- 2 Context-Sensitive Grammars
- 3 Normal Forms
 - Chomsky and Greibach Normal Forms
 - Converting to Chomsky Normal Form

Reading: Kozen, ch. 21 (on **Normal Form**)

Derivations in Context-free Grammars

Consider a simple CFG with non-terminal symbols $\{S, A, B\}$ and terminal symbols $\{a, b\}$:

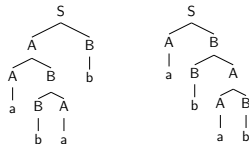
$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow AB \mid a \\ B &\rightarrow BA \mid b \end{aligned}$$

Example 1 – rewriting leftmost NT

$S \Rightarrow AB \Rightarrow \underline{A}B \Rightarrow a\underline{B}B \Rightarrow aB\underline{A}B \Rightarrow aB\underline{a}B \Rightarrow aBa\underline{B} \Rightarrow aBaB\underline{A} \Rightarrow aBaB\underline{a} \Rightarrow aBaBa\underline{B} \Rightarrow aBaBaB \Rightarrow aBaBaB$

Non-equivalent Derivations

Their tree diagrams reveal that the elements of the string **abab** come from different NTs.



When a string has more than one structural analysis with respect to a grammar, it is called **ambiguous** with respect to that grammar. **Ambiguity** is one of the key ideas in Inf2A. Ambiguity is **not** a property of the order of phrase-structure rule applications: Order is still irrelevant.

Quick in-class exercise

Consider a CFG with non-terminals $\{S, NP, VP, Adj, N, V\}$, terminals $\{\text{fish, police, scots}\}$ and the following PS rules:

$$\begin{aligned} S &\rightarrow NP \ VP \\ NP &\rightarrow Adj \ N \mid N \\ VP &\rightarrow V \mid V \ NP \\ Adj &\rightarrow \text{scots} \\ N &\rightarrow \text{fish} \mid \text{police} \mid \text{scots} \\ V &\rightarrow \text{fish} \mid \text{police} \end{aligned}$$

- Does this CFG produce ambiguous strings?

Quick in-class exercise

Consider a CFG with non-terminals $\{S, NP, VP, Adj, N, V\}$, terminals $\{\text{fish, police, scots}\}$ and the following PS rules:

$$\begin{aligned} S &\rightarrow NP \ VP \\ NP &\rightarrow Adj \ N \mid N \\ VP &\rightarrow V \mid V \ NP \\ Adj &\rightarrow \text{scots} \\ N &\rightarrow \text{fish} \mid \text{police} \mid \text{scots} \\ V &\rightarrow \text{fish} \mid \text{police} \end{aligned}$$

- Does this CFG produce ambiguous strings?
- Does this CFG produce unambiguous strings?

Quick in-class exercise

Consider a CFG with non-terminals $\{S, NP, VP, Adj, N, V\}$, terminals $\{\text{fish, police, scots}\}$ and the following PS rules:

$$\begin{aligned} S &\rightarrow NP \ VP \\ NP &\rightarrow Adj \ N \mid N \\ VP &\rightarrow V \mid V \ NP \\ Adj &\rightarrow \text{scots} \\ N &\rightarrow \text{fish} \mid \text{police} \mid \text{scots} \\ V &\rightarrow \text{fish} \mid \text{police} \end{aligned}$$

- Does this CFG produce ambiguous strings?
- Does this CFG produce unambiguous strings?
- Provide an example of each (plus their derivations).

Derivations in Context-Sensitive Grammars

Why do we stress that the order of rule applications doesn't matter with context-free grammars?

There are powerful **context-sensitive grammars**, and also weaker ones. With powerful CGS, order of rule applications can matter:

- 1 Different canonical orders of rule applications can produce different string sets: The same CSG restricted to different canonical orders can produce different languages.
- 2 Alternatively, if order isn't restricted, each rule chosen for use in a derivation can constrain what rules can apply next, so we **can't** just vary the order of rule application.



Derivations in Context-Sensitive Grammars

Consider part of a CSG with non-terminals $\{S, W, X, Y, Z\}$ and terminals $\{a, b, r, s\}$:

$$\begin{aligned} S &\rightarrow aXbY \\ Xb &\rightarrow ZWb \\ XbY &\rightarrow Xbrs \\ WbY &\rightarrow Wbst \end{aligned}$$

Example 6 – rewriting X first

$$S \Rightarrow aXbY \Rightarrow aZWbY \Rightarrow aZWbst \Rightarrow \dots$$

Example 7 – rewriting Y first

$$S \Rightarrow aXbY \Rightarrow aXbrs \Rightarrow aZWbrs \Rightarrow \dots$$



Derivations in Context-Sensitive Grammars

The left-to-right derivation (Ex 6) allows only the second production for Y ($WbY \rightarrow Wbst$). So strings ending in **bst** are in the language.

The right-to-left derivation (Ex 7) allows only the first production for Y ($XbY \rightarrow Xbrs$). So strings ending in **brs** are in the language.

So with CSGs, restricting the derivation order can produce different languages.

Q: Was this the case with CFGs?

N.B. Human languages are thought to be **weakly context-sensitive**. With **weak** forms of CSG, derivation order does **not** matter!



Normal Forms

There are two **canonical** (aka **normal**) forms for PS rules.

Chomsky Normal Form: All productions are of the form:

$$\begin{aligned} A &\rightarrow BC \\ A &\rightarrow a \end{aligned}$$

where A, B, C are NT symbols and a is a terminal symbol.

Greibach Normal Form: All productions are of the form:

$$A \rightarrow aB_1B_2 \dots B_k \quad k \geq 0$$

where A, B_1, \dots, B_k are NT symbols and a is a terminal symbol.

The basic CKY parser (Week 6) assumes all production rules are in Chomsky Normal Form. Other efficient parsers use an extended version of Greibach Normal Form.



Converting to Chomsky Normal Form

Recall the simple CFG used for generating L1:

$$\begin{aligned} V &= \{a, b, S\} \\ \Sigma &= S \\ S &\rightarrow aSb \\ S &\rightarrow ab \end{aligned}$$

Convert this to Chomsky Normal Form by:

- Adding a new non-terminal symbol for each terminal symbol:

$$\begin{aligned} A &\rightarrow a \\ B &\rightarrow b \end{aligned}$$

- Replace terminal symbols in the original rules with these new non-terminals:

$$\begin{aligned} S &\rightarrow ASB \\ S &\rightarrow AB \end{aligned}$$



Converting to Chomsky Normal Form

- For any rule with more than two non-terminal symbols on the RHS, add a new non-terminal that rewrites as the final $k - 1$ symbols on the RHS.

$$\begin{aligned} S &\rightarrow AC \\ C &\rightarrow SB \end{aligned}$$
- Continue introducing such non-terminals until there is no rule whose RHS has more than two non-terminals

Chomsky Normal Form grammar for L1:

$$\begin{aligned} S &\rightarrow AC & C &\rightarrow SB \\ S &\rightarrow AB & A &\rightarrow a \\ B &\rightarrow b \end{aligned}$$

Kozen gives a proof that the two grammars produce the same string set. Do they assign the strings the same structure?



Summary

- Derivation:** sequence of strings produce by applications of grammar rules; can be left-most or right-most.
- Tree structure diagram:** graphs the structure of a string independent of the derivation order.
- Ambiguity:** a string can have more than one structure in a given grammar.
- Normal form:** standardized form for grammar rules; Chomsky and Greibach normal forms most important.

