Fixing Non-LL Grammars

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Objectives

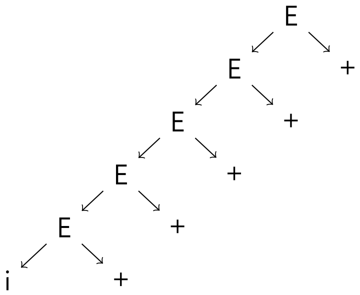
Last time we talked about grammars cannot be parsed using LL. Here we will try to fix them.

- Eliminate left recursion and mutual left recursion from a grammar.
- Eliminate common prefixes from a grammar.
- Detect and eliminate conflicts with FIRST and FOLLOW sets.

The Idea

Consider deriving `i++++` from the following grammar:

$E \rightarrow E +$	"We can have as many `+`s as we want *at the end* of the sentence."
$E \rightarrow i$	"The first word must be an `i`."



More Complicated Example

Consider the following grammar. What does it mean?

$$B \longrightarrow Bxy \mid Bz \mid q \mid r$$

- At the end can come any combination of `x y` or `z`.
- At the beginning can come `q` or `r`.

Eliminating the Left Recursion

We can rewrite these grammars
$$E \to E + \mid i$$
$$B \to Bxy \mid Bz \mid q \mid r$$
using the following transformation:

- Productions of the form $S \to \beta$ become $S \to \beta S'$.
- Productions of the form $S \to S\alpha$ become $S' \to \alpha S'$.
- Add $S' \to \epsilon$.

Result:
$$E \to iE'$$
$$E' \to +E' \mid \epsilon$$
$$B \to qB' \mid rB'$$
$$B' \to xyB' \mid zB' \mid \epsilon$$

Mutual Recursions!

Things are slightly more complicated if we have mutual recursions.

$$
\begin{aligned}
A &\rightarrow Aa \mid Bb \mid Cc \mid q \\
B &\rightarrow Ax \mid By \mid Cz \mid rA \\
C &\rightarrow Ai \mid Bj \mid Ck \mid sB
\end{aligned}
$$

How to do it:

- Take the first symbol (A) and eliminate immediate left recursion.
- Take the second symbol (B) and substitute left recursions to A. Then eliminate immediate left recursion in B.
- Take the third symbol (C) and substitute left recursions to A and B. Then eliminate immediate left recursion in C.

Left Recursion Example

Here is a more complex left recursion.

$A \rightarrow Aa \mid Bb \mid Cc \mid q$

$B \rightarrow Ax \mid By \mid Cz \mid rA$

$C \rightarrow Ai \mid Bj \mid Ck \mid sB$

First we eliminate the left recursion from A.

$A \rightarrow Aa \mid Bb \mid Cc \mid q$

This is the result:

$A \rightarrow BbA' \mid CcA' \mid qA'$

$A' \rightarrow aA' \mid \epsilon$

Left Recursion Example, 2

We substituting in the new definition of A, and now we will work on the B productions.

$A \rightarrow BbA' \mid CcA' \mid qA'$

$A' \rightarrow aA' \mid \epsilon$

$C \rightarrow Ai \mid Bj \mid Ck \mid sB$

First, we eliminate the "backward" recursion from B to A.

Start: $B \rightarrow Ax$

Result: $B \rightarrow BbA'x \mid CcA'x \mid qA'x$

Left Recursion Example, 3

$$A \rightarrow BbA' \mid CcA' \mid qA'$$
$$A' \rightarrow aA' \mid \epsilon$$
$$B \rightarrow BbA'x \mid CcA'x \mid qA'x \mid By \mid Cz \mid rA$$
$$C \rightarrow Ai \mid Bj \mid Ck \mid sB$$

Now we can eliminate the simple left recursion in B:

$$B \rightarrow CcA'xB' \mid qA'xB' \mid CzB' \mid rAB'$$
$$B' \rightarrow bA'xB' \mid yB' \mid \epsilon$$

Left Recursion Example, 4

$$
\begin{aligned}
A &\rightarrow BbA' \mid CcA' \mid qA' \\
A' &\rightarrow aA' \mid \epsilon \\
B &\rightarrow CcA'xB' \mid qA'xB' \mid CzB' \mid rAB' \\
B' &\rightarrow bA'xB' \mid yB' \mid \epsilon
\end{aligned}
$$

Now production C: First, replace left recursive calls to A ...

$$C \rightarrow B\ bA'i \mid CcA'i \mid qA'i \mid B\ j \mid Ck \mid sB$$

Next, replace left recursive calls to B (this gets messy) ...

$$
\begin{aligned}
C \rightarrow\ & CcA'xB'\ bA'i \mid \qquad bA'i \mid CzB'\ bA'i \mid rAB'\ bA'i \\
& CcA'xB'\ j \mid \qquad j \mid \qquad j \mid rAB'\ j \\
& CcA'i \mid qA'i \mid Ck \mid sB
\end{aligned}
$$

Left Recursion Example, 5

Reorganizing C, we have:

$$C \rightarrow \begin{array}{l} qA'xB'bA'i \mid rAB'bA'i \mid qA'xB'j \mid rAB'j \mid qA'i \mid sB \\ CcA'xB'bA'i \mid CzB'bA'i \mid CcA'xB'j \mid CzB'j \mid CcA'i \mid Ck \end{array}$$

Eliminating left recursion gives us:

$$\begin{aligned} C &\rightarrow qA'xB'bA'iC' \mid rAB'bA'iC' \mid qA'xB'jC' \\ &\qquad \mid rAB'jC' \mid qA'iC' \mid sBC' \\ C' &\rightarrow cA'xB'bA'iC' \mid zB'bA'iC' \mid cA'xB'jC' \\ &\qquad \mid zB'jC' \mid cA'iC' \mid kC' \mid \epsilon \end{aligned}$$

The Result

Our final grammar:

$$
\begin{aligned}
A &\rightarrow BbA' \mid CcA' \mid qA' \\
A' &\rightarrow aA' \mid \epsilon \\
B &\rightarrow CcA'xB' \mid qA'xB' \mid CzB' \mid rAB' \\
B' &\rightarrow bA'xB' \mid yB' \mid \epsilon \\
C &\rightarrow qA'xB'bA'iC' \mid rAB'bA'iC' \mid qA'xB'jC' \\
&\quad \mid rAB'jC' \mid qA'iC' \mid sBC' \\
C' &\rightarrow cA'xB'bA'iC' \mid zB'bA'iC' \mid cA'xB'jC' \\
&\quad \mid zB'jC' \mid cA'iC' \mid kC' \mid \epsilon
\end{aligned}
$$

Beautiful, isn't it? I wonder why we don't do this more often?

- Disclaimer: If there is a cycle ($A \rightarrow^+ A$) or an epsilon production ($A \rightarrow \epsilon$) then this technique is not guaranteed to work.

Common Prefix

This grammar has common prefixes.

$$
\begin{aligned}
A &\rightarrow xyB \mid CyC \mid q \\
B &\rightarrow zC \mid zx \mid w \\
C &\rightarrow y \mid x
\end{aligned}
$$

To check for common prefixes, take a nonterminal and compare the FIRST sets of each production.

Production	FirstSet
$A \rightarrow xyB$	$\{x\}$
$A \rightarrow CyC$	$\{x, y\}$
$A \rightarrow q$	$\{q\}$

If we are viewing an A, we will want to look at the next token to see which A production to use. If that token is $\mathtt{x}$, then which production do we use?

Left Factoring

If $A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \gamma$ we can rewrite it as $\begin{array}{l} A \to \alpha A' \mid \gamma \\ A' \to \beta_1 \mid \beta_2. \end{array}$

So, in our example:

$$\begin{array}{lcl} A \to xyB \mid CyC \mid q & \text{becomes} & A \to xA' \mid q \mid yyC \\ B \to zC \mid zx \mid w & & A' \to yB \mid yC \\ C \to y \mid x & & B \to zB' \mid w \\ & & B' \to C \mid x \\ & & C \to y \mid x. \end{array}$$

Sometimes you'll need to do this more than once. Note that this process can destroy the meaning of the nonterminals.

Epsilon Productions

- Epsilon productions have to be handled with care.

$$
\begin{aligned}
A &\rightarrow Bc \\
&\mid x \\
B &\rightarrow c \\
&\mid \epsilon
\end{aligned}
$$

Is this LL?

Epsilon Productions

$$
\begin{array}{rcl}
A & \rightarrow & Bc \\
 & | & x \\
B & \rightarrow & c \\
 & | & \epsilon
\end{array}
$$

- $FOLLOW(B) = \{c\}$, and $FIRST(B) = \{c\}$, so we have a conflict when trying to parse B.
- We can substitute the B rule into the A rule to fix this ...
- Be sure to check if you have introduced a common prefix though!

$$
\begin{array}{rcl}
A & \rightarrow & cc \\
 & | & c \\
 & | & x
\end{array}
\quad \Rightarrow \quad
\begin{array}{rcl}
A & \rightarrow & cA' \\
 & | & x \\
A' & \rightarrow & c \\
 & | & \epsilon
\end{array}
$$