Answering Questions about $LL(1)$ Grammars

Peter McCormick
pdm@cs.toronto.edu

February 27, 2015

1 Definitions

In the following definitions and rules, we will use these conventions:

- Terminals are represented by lower case Roman letters (t, x, y, z)
- Non-terminals are represented by capitalized Roman letters (A, B, X, Y)
- Greek letters (α, β, γ) represent strings that range over both terminals and non-terminals (such as $A\,w\,X\,t$) and can include the empty string λ
- $X \Rightarrow^+ \alpha$ means that there is a valid sequence of derivations starting from the non-terminal X to the string α

1.1 Nullable

α is *nullable* iff $\alpha \Rightarrow^* \lambda$

1.2 First Sets:

$First(z\,\beta)$	$=$	$\{z\}$	z is a terminal symbol
$First(\lambda)$	$=$	$\{\}$	(the empty set)
$First(B\gamma)$	$=$	$First(B)$	(if B is *not* nullable)
	$=$	$First(B) \cup First(\gamma)$	(if B *is* nullable)

1.3 Follow Set Construction Rules

Starting from initially empty follow sets, iteratively apply these rules until the sets no longer change.

#1. If S is the goal symbol	Add $\{\$\}$ to $Follow(S)$	
#2. If $S \Rightarrow^+ \alpha\,X$	Add $\{\$\}$ to $Follow(X)$	
#3. If $S \Rightarrow^+ \alpha\,X\,\mathbf{t}\,\beta$	Add $\{t\}$ to $Follow(X)$	
#4. If $S \Rightarrow^+ \alpha\,X\,\mathbf{Y}\,\beta$	Add $First(Y)$ to $Follow(X)$	(if Y is *not* nullable)
	Add $First(Y) \cup First(\beta)$ to $Follow(X)$	(if Y *is* nullable)
#5. If $X \rightarrow \alpha\,\mathbf{Y}$	Add $Follow(X)$ to $Follow(Y)$	

1.4 Predict Sets for $LL(1)$ Grammars

Given a non-terminal A defined with several alternate productions:

$$\begin{array}{lcl} A & \rightarrow & \alpha_1 \\ & \rightarrow & \alpha_2 \\ & & \ldots \\ & \rightarrow & \alpha_n \end{array}$$

The predict set for each production $A \rightarrow \alpha_i$ is defined as

$$\begin{array}{lcll} Predict\,(A \rightarrow \alpha_i) & = & First\,(\alpha_i) & (\text{if } \alpha_i \text{ is } \textit{not} \text{ nullable}) \\ & = & First\,(\alpha_i) \cup Follow\,(A) & (\text{if } \alpha_i \textit{ is} \text{ nullable}) \end{array}$$

1.5 Factoring Out Repeated Prefixes

For a prospective $LL(1)$ grammar, a repeated common prefix in alternate productions is problematic because the predict sets of the alternatives are not disjoint. For example, in the definition of

$$\begin{array}{lcl} A & \rightarrow & B\,\alpha \\ & \rightarrow & B\,\beta \end{array}$$

an $LL(1)$ parser could not distinguish between the two alternatives for A since

$$\begin{array}{lcl} Predict\,(A \rightarrow B\,\alpha) & = & First\,(B\,\alpha) \\ & & (\text{if } B \text{ is not nullable}) \\ & = & First\,(B) \\ & = & First\,(B\,\beta) \\ & = & Predict\,(A \rightarrow B\,\beta) \end{array}$$

A similar argument holds even when B is nullable (why?)
Fortunately a mechanical transformation can alleviate this.

Given a non-$LL(1)$ definition of the form

$$\begin{array}{rcl} X & \rightarrow & \alpha\,\beta_1 \\ & \rightarrow & \ldots \\ & \rightarrow & \alpha\,\beta_n \\ & \rightarrow & \sigma_1 \\ & \rightarrow & \ldots \\ & \rightarrow & \sigma_m \end{array}$$

The common prefix can be factored out into additional helper non-terminals:

$$\begin{array}{rcl} X & \rightarrow & \alpha\,Xtail \\ & \rightarrow & \sigma_1 \\ & \rightarrow & \ldots \\ & \rightarrow & \sigma_m \\ Xtail & \rightarrow & \beta_1 \\ & \rightarrow & \ldots \\ & \rightarrow & \beta_n \end{array}$$

1.6 Factoring Out Left Recursion

Left recursion is similarly not directly expressible in $LL(1)$ grammars, but a slightly more elaborate transformation can rectify this.

For a non-$LL(1)$ left recursive definition of the form,

$$\begin{array}{rcl} X & \rightarrow & X\,\alpha \\ & \rightarrow & \sigma_1 \\ & \rightarrow & \ldots \\ & \rightarrow & \sigma_k \end{array}$$

break the recursion in the first production by transforming it into

$$\begin{array}{rcl} X & \rightarrow & \sigma_1\,Xtail \\ & \rightarrow & \ldots \\ & \rightarrow & \sigma_k\,Xtail \\ Xtail & \rightarrow & \alpha\,Xtail \\ & \rightarrow & \lambda \end{array}$$

Repeat this process until all left recursive productions are removed.

For example, given the definition

$$
\begin{array}{rcl}
X & \rightarrow & X\,\alpha_1 \\
 & \rightarrow & X\,\alpha_2 \\
 & \rightarrow & \beta_3 \\
 & \rightarrow & \ldots \\
 & \rightarrow & \beta_k
\end{array}
$$

After the first transformation,

$$
\begin{array}{rcl}
X & \rightarrow & X\,\alpha_2\,X_1 \\
 & \rightarrow & \beta_3\,X_1 \\
 & \rightarrow & \ldots \\
 & \rightarrow & \beta_k\,X_1 \\
X_1 & \rightarrow & \alpha_1\,X_1 \\
 & \rightarrow & \lambda
\end{array}
$$

Then after the second,

$$
\begin{array}{rcl}
X & \rightarrow & \beta_3\,X_1\,X_2 \\
 & \rightarrow & \ldots \\
 & \rightarrow & \beta_k\,X_1\,X_2 \\
X_1 & \rightarrow & \alpha_1\,X_1 \\
 & \rightarrow & \lambda \\
X_2 & \rightarrow & \alpha_2\,X_1\,X_2 \\
 & \rightarrow & \lambda
\end{array}
$$

Try applying this procedure to $E \rightarrow E + T | E - T | T$.

2 Example

Consider the following grammar:

$$
\begin{array}{rcl}
S & \rightarrow & S\,z\,A \\
 & \rightarrow & z\,B \\
 & \rightarrow & B \\
A & \rightarrow & y\,A \\
 & \rightarrow & w \\
B & \rightarrow & B\,x\,A \\
 & \rightarrow & A\,y\,A \\
 & \rightarrow & A
\end{array}
$$

There are two clues that tell us that these definitions do not constitute a valid $LL\,(1)$ grammar:

1. In the last two productions for B, $B \to A\,y\,A$ and $B \to A$, they have a common prefix of A.

2. The left recursion in the productions $S \to S\,z\,A$ and $B \to B\,x\,A$.

We will apply the transformation procedures and verify that the resulting grammar is $LL\,(1)$.

First we factor out the common prefix in the B productions:

$$\begin{array}{rcl} B & \to & B\,x\,A \\ & \to & Bhead\,Btail \\ Bhead & \to & A \\ Btail & \to & y\,A \\ & \to & \lambda \end{array}$$

After simplifying and a rename, the grammar has become:

$$\begin{array}{rcl} S & \to & S\,z\,A \\ & \to & z\,B \\ & \to & B \\ A & \to & y\,A \\ & \to & w \\ B & \to & B\,x\,A \\ & \to & A\,B_1 \\ B_1 & \to & y\,A \\ & \to & \lambda \end{array}$$

Next we break up the left recursive productions for S and B:

$$\begin{array}{rcl} S & \to & z\,B\,S_1 \\ & \to & B\,S_1 \\ S_1 & \to & z\,A\,S_1 \\ & \to & \lambda \\ A & \to & y\,A \\ & \to & w \\ B & \to & A\,B_1\,B_2 \\ B_1 & \to & y\,A \end{array}$$

$$\begin{array}{rcl} & \to & \lambda \\ B_2 & \to & x\,A\,B_2 \\ & \to & \lambda \end{array}$$

We note that S_1, B_1 and B_2 are nullable, since their definitions directly include $\to \lambda$ productions. If there had been a production $X \to S_1$, it too would be nullable by transitivity, since $X \Rightarrow S_1 \Rightarrow \lambda$.

We compute the First sets for each non-terminal:

$$\begin{array}{rcl} First\,(S) & = & First\,(z\,B\,S_1) \cup First\,(B\,S_1) \\ & = & \{z\} \cup First\,(B) \\ & & \text{(since } B \text{ is not nullable)} \\ First\,(S_1) & = & First\,(z\,A\,S_1) \cup First\,(\lambda) \\ & = & \{z\} \cup \{\} = \{z\} \\ First\,(A) & = & First\,(y\,A) \cup First\,(w) \\ & = & \{y\} \cup \{w\} = \{w, y\} \\ First\,(B) & = & First\,(A\,B_1\,B_2) \\ & = & First\,(A) \\ & & \text{(since } A \text{ is not nullable)} \\ & = & \{w, y\} \\ First\,(B_1) & = & First\,(y\,A) \cup First\,(\lambda) \\ & = & \{y\} \\ First\,(B_2) & = & First\,(x\,A\,B_2) \cup First\,(\lambda) \\ & = & \{x\} \end{array}$$

Finally, $First\,(S) = \{w, y, z\}$.

	w	x	y	z
$First\,(S)$	✓		✓	✓
$First\,(S_1)$				✓
$First\,(A)$	✓		✓	
$First\,(B)$	✓		✓	
$First\,(B_1)$			✓	
$First\,(B_2)$		✓		

Next we compute the Follow sets. The process of iteratively applying the construction rules is entirely mechanical, but we will attempt to inspect the rules and the grammar in order to do this efficiently.

Since S is the goal symbol, we apply rule #1 and start by adding {\$} to its follow set, so $\{\$\} \subset Follow\,(S)$.

Since the two alternatives for S can be derived in one step, we will begin by applying the rules recursively from there.

- Since $S \Rightarrow z\,B\,S_1$ and $S \Rightarrow B\,S_1$, we can apply rule #2 to the same effect in both cases ($X = S_1$ and $\alpha = z\,B$ or $\alpha = B$, respectively), so we add $\{\$\}$ to $Follow\,(S_1)$, thus $\{\$\} \subset Follow\,(S_1)$
- Rule #4 similarly applies to both one step derivations of S to the same effect, so in the case of $S \Rightarrow z\,B\,S_1$ (letting $\alpha = z$, $X = B$, $Y = S_1$, $\beta = \lambda$), since S_1 is nullable, we add

$$\begin{aligned} First\,(Y) \cup First\,(\beta) &= First\,(S_1) \cup First\,(\lambda) \\ &= \{z\} \cup \{\} \end{aligned}$$

  to $Follow\,(B)$, so $\{z\} \subset Follow\,(B)$.

At this point we know that $\{\$\} \subset Follow\,(S)$, $\{\$\} \subset Follow\,(S_1)$ and $\{z\} \subset Follow\,(B)$.

As per rule #3, we look in the grammar definition for anywhere that a terminal appears immediately after a non-terminal (a sub-string sequence of the form $X\,t$.) There are not any in this example.

Considering rule #4, we look for all other immediately adjacent non-terminals in the grammar productions:

1. A and S_1 in $S_1 \rightarrow z\,A\,S_1$
2. A and B_1 in $B \rightarrow A\,B_1\,B_2$
3. Also B_1 and B_2 in $B \rightarrow A\,B_1\,B_2$
4. A and B_2 in $B_2 \rightarrow x\,A\,B_2$.

While rule #4 is defined in terms of strings which are derivable from the start symbol, if there is any path from the start symbol to a given non-terminal (i.e. it is not orphaned), we can always find a suitable α and β to fit the pattern.

For example, we can derive a string involving adjacent A and B_2 (the last in the list) via the following sequence of derivations:

$$\begin{aligned} S &\Rightarrow z\,B\,S_1 \\ &\Rightarrow z\,\underbrace{A\,B_1\,B_2}\,S_1 \\ &\qquad \text{(expanding } B\text{)} \\ &\Rightarrow z\,\overbrace{A\,B_1\,\underbrace{x\,A\,B_2}}\,S_1 \\ &\qquad \text{(expanding } B_2 \text{ once)} \end{aligned}$$

In this case rule #4 applies with $\alpha = z\,A\,B_1\,x$, $X = A$, $Y = B_2$ and $\beta = S_1$. Since B_2 is nullable, we add $First\,(B_2) \cup First\,(S_1) = \{x\} \cup \{z\}$ to $Follow\,(A)$, so $\{x, z\} \subset Follow\,(A)$.

Since B_2 was nullable, from the definition of rule #4 we can see that the contents of β, and thus $First\,(\beta)$, were significant, so we should check if there

were any other ways to derive an expression involving B_2 such that something other than S_1 would appear immediately afterward. B_2 appears as the right most symbol in its own definition and in the definition of B, which in turn only appears preceding an S_1.

Back to the list of adjacent non-terminals:

1. $S \Rightarrow z\,B\,S_1 \Rightarrow z\,B\,\underbrace{z\,A\,S_1}$, applying rule #4 ($\alpha = z\,B\,z$, $X = A$, $Y = S_1$, $\beta = \lambda$), since S_1 is nullable add $First\,(S_1) \cup First\,(\lambda) = \{z\}$ to $Follow\,(A)$, so $\{z\} \subset Follow\,(A)$.

2. $S \Rightarrow B\,S_1 \Rightarrow \underbrace{A\,B_1\,B_2}\,S_1$, applying rule #4 ($\alpha = \lambda$, $X = A$, $Y = B_1$, $\beta = B_2\,S_1$), since B_1 is nullable add

$$\begin{aligned} First\,(B_1) \cup First\,(B_2\,S_1) &= \{y\} \cup [First\,(B_2) \cup First\,(S_1)] \\ &\quad \text{(since } B_2 \text{ is nullable)} \\ &= \{y\} \cup \{x\} \cup \{z\} \end{aligned}$$

   to $Follow\,(A)$, so $\{x, y, z\} \subset Follow\,(A)$.

3. Same derivation, applying rule #4 ($\alpha = \lambda\,A$, $X = B_1$, $Y = B_2$, $\beta = S_1$), since B_2 is nullable add $First\,(B_2) \cup First\,(S_1) = \{x\} \cup \{z\}$ to $Follow\,(B_1)$, so $\{x, z\} \subset Follow\,(B_1)$.

At this point, we know that $\{\$\} \subset Follow\,(S)$, $\{\$\} \subset Follow\,(S_1)$, $\{x, y, z\} \subset Follow\,(A)$, $\{z\} \subset Follow\,(B)$, $\{x, z\} \subset Follow\,(B_1)$.

Finally, we look to apply rule #5 to propagate the follow sets:

- Since $S \to B\,S_1$, add $Follow\,(S)$ to $Follow\,(S_1)$, so $\{\$\} \subset Follow\,(S_1)$
- Since $B \to A\,B_1\,B_2$, add $Follow\,(B)$ to $Follow\,(B_2)$, so $\{z\} \subset Follow\,(B_2)$
- Since $B_1 \to y\,A$, add $Follow\,(B_1)$ to $Follow\,(A)$, so $\{x, z\} \subset Follow\,(A)$

We now have our completed Follow sets:

	w	x	y	z	\$
$Follow\,(S)$					✓
$Follow\,(S_1)$					✓
$Follow\,(A)$		✓	✓	✓	
$Follow\,(B)$				✓	
$Follow\,(B_1)$		✓		✓	
$Follow\,(B_2)$				✓	

We are ready to compute the predict sets:

$$\begin{aligned} Predict\,(S \to z\,B\,S_1) &= First\,(z\,B\,S_1) \\ &= \{z\} \end{aligned}$$

$$
\begin{aligned}
Predict\,(S \to B\,S_1) &= First\,(B\,S_1) \\
&\quad \text{(since } B \text{ is not nullable)} \\
&= \{w, y\} \\
Predict\,(S_1 \to z\,A\,S_1) &= First\,(z\,A\,S_1) \\
&= \{z\} \\
Predict\,(S_1 \to \lambda) &= First\,(\lambda) \cup Follow\,(S_1) \\
&\quad \text{(since } \lambda \text{ is nullable)} \\
&= \{\$\} \\
Predict\,(A \to y\,A) &= First\,(y\,A) \\
&= \{y\} \\
Predict\,(A \to w) &= First\,(w) \\
&= \{w\} \\
Predict\,(B \to A\,B_1\,B_2) &= First\,(A\,B_1\,B_2) \\
&\quad \text{(since } A \text{ is not nullable)} \\
&= First\,(A) \\
&= \{w, y\} \\
Predict\,(B_1 \to y\,A) &= First\,(y\,A) \\
&= \{y\} \\
Predict\,(B_1 \to \lambda) &= First\,(\lambda) \cup Follow\,(B_1) \\
&\quad \text{(since } \lambda \text{ is nullable)} \\
&= \{x, z\} \\
Predict\,(B_2 \to x\,A\,B_2) &= First\,(x\,A\,B_2) \\
&= \{x\} \\
Predict\,(B_2 \to \lambda) &= First\,(\lambda) \cup Follow\,(B_2) \\
&\quad \text{(since } \lambda \text{ is nullable)} \\
&= \{z\}
\end{aligned}
$$

Summarizing:

	w	x	y	z	\$
$Predict\,(S \to z\,B\,S_1)$				✓	
$Predict\,(S \to B\,S_1)$	✓		✓		
$Predict\,(S_1 \to z\,A\,S_1)$				✓	
$Predict\,(S_1 \to \lambda)$					✓
$Predict\,(A \to y\,A)$			✓		
$Predict\,(A \to w)$	✓				
$Predict\,(B \to A\,B_1\,B_2)$	✓		✓		
$Predict\,(B_1 \to y\,A)$			✓		
$Predict\,(B_1 \to \lambda)$		✓		✓	
$Predict\,(B_2 \to x\,A\,B_2)$		✓			
$Predict\,(B_2 \to \lambda)$				✓	

For every non-terminal in the grammar, the predict sets for each of their alternatives are mutually disjoint, and thus this is a valid $LL\,(1)$ grammar. □