Computing PREDICT and FOLLOW sets

SI 413 - Programming Languages and Implementation

Dr. Daniel S. Roche

United States Naval Academy

Fall 2011

Set Definitions

Given any CFG in BNF (no alternation | or Kleene $*$,$+$), we can construct the following sets.

Each set contains *only tokens*, and FIRST(A), FOLLOW(A), and PREDICT(A) are defined for every non-terminal A in the language.

- EPS: All non-terminals that could expand to ϵ, the empty string
- FIRST(A): The set of tokens that could appear as the first token in an expansion of A
- FOLLOW(A): The set of tokens that could appear immediately after A in an expansion of S, the start symbol.
- PREDICT(A): The set of tokens that could appear next in a valid parse of a string in the language, when the next non-terminal in the parse tree is A.

Each set is defined using *mutual recursion*.

Do we need FIRST?

For top-down parsing, the only sets we really need are the PREDICT sets.

It turns out the FIRST sets aren't necessary to compute PREDICT.

So (for now) we will forget about FIRST and just worry about the other three (EPS, PREDICT, FOLLOW).

EPS

Definition: EPS contains all non-terminals that could expand to ϵ, the empty string

EPS contains:

1. Any non-terminal that has an epsilon production
2. Any non-terminal that has a production containing only non-terminals that are all in EPS

PREDICT

Definition: PREDICT(A) contains all tokens that could appear next on the token stream when we are expecting an A.

PREDICT(A) contains:

1. Any token that appears as the leftmost symbol in a production rule for A.
2. For every non-terminal B that appears as the leftmost symbol in a production rule for A, every token in PREDICT(B).
3. If $A \in$ EPS, every token in FOLLOW(A).

FOLLOW

Definition: FOLLOW(A) contains all tokens that could come right after A in a valid parse.

FOLLOW(A) contains:

1. Any token that immediately follows A on any right-hand side of a production
2. For any non-terminal B that immediately follows A on any right-hand side of a production, every token in PREDICT(B)
3. For any non-terminal B such that A appears right-most in a right-hand side of a production of B, every token in FOLLOW(B).

There is also the special rule for the *start symbol* that FOLLOW(S) always contains \$, the end-of-file token.

Example: Calculator Language

Here's the LL(1) grammar for the calculator language:

$$
\begin{aligned}
S &\rightarrow \textit{exp}\ \texttt{STOP} \\
\textit{exp} &\rightarrow \textit{term}\ \textit{exptail} \\
\textit{exptail} &\rightarrow \epsilon \mid \texttt{OPA}\ \textit{term}\ \textit{exptail} \\
\textit{term} &\rightarrow \textit{sfactor}\ \textit{termtail} \\
\textit{termtail} &\rightarrow \epsilon \mid \texttt{OPM}\ \textit{factor}\ \textit{termtail} \\
\textit{sfactor} &\rightarrow \texttt{OPA}\ \textit{factor} \mid \textit{factor} \\
\textit{factor} &\rightarrow \texttt{NUM} \mid \texttt{LP}\ \textit{exp}\ \texttt{RP}
\end{aligned}
$$

Computing PREDICT for the calculator language

EPS $= \{\}$

Non-terminal	PREDICT	FOLLOW
S		$
exp		
exptail		
term		
termtail		
sfactor		
factor		

Computing PREDICT for the calculator language

EPS $= \{exptail, termtail\}$

Non-terminal	PREDICT	FOLLOW
S		$
exp		
exptail		
term		
termtail		
sfactor		
factor		

- First construct EPS; we just have to use Rule 1.

Computing PREDICT for the calculator language

EPS $= \{exptail, termtail\}$

Non-terminal	PREDICT	FOLLOW
S		$
exp		
exptail	OPA	
term		
termtail	OPM	
sfactor	OPA	
factor	NUM, LP	

- Apply Rule 1 for PREDICT in four places

Computing PREDICT for the calculator language

EPS $= \{exptail, termtail\}$

Non-terminal	PREDICT	FOLLOW
S		$
exp		STOP,RP
exptail	OPA	
term		
termtail	OPM	
sfactor	OPA	
factor	NUM, LP	

- Apply Rule 1 for FOLLOW in two places

Computing PREDICT for the calculator language

EPS $= \{exptail, termtail\}$

Non-terminal	PREDICT	FOLLOW
S		$
exp		STOP,RP
exptail	OPA	STOP,RP
term		
termtail	OPM	
sfactor	OPA	
factor	NUM, LP	

- Apply Rule 3 for FOLLOW to *exptail*

Computing PREDICT for the calculator language

EPS $= \{exptail, termtail\}$

Non-terminal	PREDICT	FOLLOW
S		$
exp		STOP,RP
exptail	OPA,STOP,RP	STOP,RP
term		
termtail	OPM	
sfactor	OPA	
factor	NUM, LP	

- Apply Rule 3 for PREDICT to *exptail*

Computing PREDICT for the calculator language

EPS $= \{exptail, termtail\}$

Non-terminal	PREDICT	FOLLOW
S		$
exp		STOP,RP
exptail	OPA,STOP,RP	STOP,RP
term		OPA,STOP,RP
termtail	OPM	
sfactor	OPA	
factor	NUM, LP	

- Apply Rule 2 for FOLLOW to *term*

Computing PREDICT for the calculator language

EPS $= \{exptail, termtail\}$

Non-terminal	PREDICT	FOLLOW
S		$
exp		STOP,RP
exptail	OPA,STOP,RP	STOP,RP
term		OPA,STOP,RP
termtail	OPM	OPA,STOP,RP
sfactor	OPA	
factor	NUM, LP	

- Apply Rule 3 for FOLLOW to *termtail*

Computing PREDICT for the calculator language

EPS $= \{exptail, termtail\}$

Non-terminal	PREDICT	FOLLOW
S		$
exp		STOP,RP
exptail	OPA,STOP,RP	STOP,RP
term		OPA,STOP,RP
termtail	OPM,OPA,STOP,RP	OPA,STOP,RP
sfactor	OPA	
factor	NUM, LP	

- Apply Rule 3 for PREDICT to *termtail*

Computing PREDICT for the calculator language

EPS $= \{exptail, termtail\}$

Non-terminal	PREDICT	FOLLOW
S		$
exp		STOP,RP
exptail	OPA,STOP,RP	STOP,RP
term		OPA,STOP,RP
termtail	OPM,OPA,STOP,RP	OPA,STOP,RP
sfactor	OPA	OPM,OPA,STOP,RP
factor	NUM, LP	OPM,OPA,STOP,RP

- Apply Rule 2 for FOLLOW in two places

Computing PREDICT for the calculator language

EPS $= \{exptail, termtail\}$

Non-terminal	PREDICT	FOLLOW
S	OPA,NUM,LP	$
exp	OPA,NUM,LP	STOP,RP
exptail	OPA,STOP,RP	STOP,RP
term	OPA,NUM,LP	OPA,STOP,RP
termtail	OPM,OPA,STOP,RP	OPA,STOP,RP
sfactor	OPA,NUM,LP	OPM,OPA,STOP,RP
factor	NUM, LP	OPM,OPA,STOP,RP

- Apply Rule 2 for PREDICT from the bottom up

Computing PREDICT for the calculator language

EPS $= \{exptail, termtail\}$

Non-terminal	PREDICT	FOLLOW
S	OPA,NUM,LP	$
exp	OPA,NUM,LP	STOP,RP
exptail	OPA,STOP,RP	STOP,RP
term	OPA,NUM,LP	OPA,STOP,RP
termtail	OPM,OPA,STOP,RP	OPA,STOP,RP
sfactor	OPA,NUM,LP	OPM,OPA,STOP,RP
factor	NUM, LP	OPM,OPA,STOP,RP

- Check that none of the rules add anything to any set.

Tagging set elements

For use in top-down parsing, every symbol in EPS and in any PREDICT set is *tagged* with a production rule, as follows:

- Every non-terminal in EPS is tagged with the rule that gives an ϵ-production for that non-terminal.
- Every token in PREDICT(A) that was added from Rule 1 or Rule 2 on Slide 5 is tagged with the production of A that brought in that token.
- Every token in PREDICT(A) that was added from Rule 3 on Slide 5 is tagged with the tag on A in EPS; that is, the rule that gives the ϵ-production of A.

These tags indicate what the top-down parser should do when it expects A and sees a token in PREDICT(A).