CSI4101 Compiler Design
2nd Semester, 2019

Homework 2

This homework has been provided for your self-study. There is nothing to hand in with this homework. The solution to the example in Section 3 will be provided ahead of the exam.

1 Algorithm to Compute First Sets

To initialize, set FIRST(X) ={} for each terminal and non-terminal symbol X in grammar G. The algorithm then proceeds according to the following three steps; the third step is repeated until no more changes to the FIRST sets occur, i.e., until a fixed-point is reached[1].

1. If $X \in V_t$ then FIRST(X) = $\{X\}$
2. If $X ::= \epsilon$ then add ϵ to FIRST(X)
3. If $X ::= Y_1 Y_2 \ldots Y_k$:
   (a) Put FIRST(Y_1) $- \{\epsilon\}$ in FIRST(X)
   (b) $\forall i : 1 < i \leq k$, if $\epsilon \in$ FIRST(Y_1) $\cap \cdots \cap$ FIRST(Y_{i-1})
   (i.e., $Y_1 \ldots Y_{i-1} \rightarrow^* \epsilon$)
   then put FIRST(Y_i) $- \{\epsilon\}$ in FIRST(X)
   (c) if $\epsilon \in$ FIRST(Y_1) $\cap \cdots \cap$ FIRST(Y_k) then put ϵ in FIRST(X)

   Repeat Step 3 until no more additions can be made.

1.1 Example

Consider the following context-free grammar. The FIRST-sets for the terminals V_t (Rule 1 above) are given in Table I. Table II shows the FIRST-sets of the non-terminals after initialization and application of Rule 2.

S' ::= S
S ::= AB
S ::= C
A ::= ef
A ::= ϵ
B ::= hg
C ::= DD
C ::= fi
D ::= g

Terminal	FIRST
e	{e}
f	{f}
g	{g}
h	{h}
i	{i}

Table I

Non-Terminal	FIRST
S'	{}
S	{}
A	$\{\epsilon\}$
B	{}
C	{}
D	{}

Table II

[1] The abbreviation "i.e." stands for the Latin "it est", meaning "that is". Please note the use of a comma before and after "i.e.". In particular, no blank is used between a word and a following comma.

We go through one iteration of Rule 3 to modify the FIRST-sets as follows:

(1) S' ::= S: Put {} in FIRST(S'), no addition.
(2) S ::= AB: Put {} in FIRST(S), no addition.
(3) S ::= C: Put {} in FIRST(S), no addition.
(4) A ::= ef: Put {e} in FIRST(A).
(5) A ::= ϵ: Put {ϵ} in FIRST(A), no addition.
(6) B ::= hg: Put {h} in FIRST(B).
(7) C ::= DD: Put {} in FIRST(C), no addition.
(8) C ::= fi: Put {f} in FIRST(C).
(9) D ::= g: Put {g} in FIRST(D).

Non-Terminal	FIRST
S'	{}
S	{}
A	{e, ϵ}
B	{h}
C	{f}
D	{g}

Because the previous step resulted in additions (changes) to the FIRST-sets, we iterate once more:

(1) S' ::= S: Put {} in FIRST(S'), no addition.
(2) S ::= AB: Put {e, h} in FIRST(S).
(3) S ::= C: Put {f} in FIRST(S).
(4) A ::= ef: Put {e} in FIRST(A), no addition.
(5) A ::= ϵ: Put {ϵ} in FIRST(A), no addition.
(6) B ::= hg: Put {h} in FIRST(B), no addition.
(7) C ::= DD: Put {g} in FIRST(C).
(8) C ::= fi: Put {f} in FIRST(C), no addition.
(9) D ::= g: Put {g} in FIRST(D), no addition.

Non-Terminal	FIRST
S'	{}
S	{e, f, h}
A	{e, ϵ}
B	{h}
C	{f, g}
D	{g}

Because the previous step resulted in further additions, we iterate once more:

(1) S' ::= S: Put {e, f, h} in FIRST(S').
(2) S ::= AB: Put {e, h} in FIRST(S), no addition.
(3) S ::= C: Put {f, g} in FIRST(S).
(4) A ::= ef: Put {e} in FIRST(A), no addition.
(5) A ::= ϵ: Put {ϵ} in FIRST(A), no addition.
(6) B ::= hg: Put {h} in FIRST(B), no addition.
(7) C ::= DD: Put {g} in FIRST(C), no addition.
(8) C ::= fi: Put {f} in FIRST(C), no addition.
(9) D ::= g: Put {g} in FIRST(D), no addition.

Non-Terminal	FIRST
S'	{e, f, h}
S	{e, f, g, h}
A	{e, ϵ}
B	{h}
C	{f, g}
D	{g}

Because of the further additions, we iterate once more:

(1) S' ::= S: Put {e, f, g, h} in FIRST(S').
(2) S ::= AB: Put {e, h} in FIRST(S), no addition.
(3) S ::= C: Put {f, g} in FIRST(S), no addition.
(4) A ::= ef: Put {e} in FIRST(A), no addition.
(5) A ::= ϵ: Put {ϵ} in FIRST(A), no addition.
(6) B ::= hg: Put {h} in FIRST(B), no addition.
(7) C ::= DD: Put {g} in FIRST(C), no addition.
(8) C ::= fi: Put {f} in FIRST(C), no addition.
(9) D ::= g: Put {g} in FIRST(D), no addition.

Non-Terminal	FIRST
S'	{e, f, g, h}
S	{e, f, g, h}
A	{e, ϵ}
B	{h}
C	{f, g}
D	{g}

Further iterations do not yield additional changes.

2 Algorithm to Compute Follow Sets

To initialize, set FOLLOW(X) $=\{\}$ for all non-terminals X in grammar G. The algorithm then proceeds according to the below two steps; the second step is repeated until no more changes to the FOLLOW sets occur (i.e., until a fixed-point is reached).

- **Note 1:** α and β represent arbitrary strings consisting of terminals and nonterminals, including ϵ. But B represents a single non-terminal. We introduced this notation on page 74 of the lecture slides on "Syntax Analysis".
- **Note 2:** In Rule 2 ($A ::= \alpha B \beta$), think of B as the non-terminal on which we "anchor" the "$\alpha B \beta$" pattern in the right-hand side of a production. This is explained in detail on pages 87–88 of the lecture slides on "Syntax Analysis".
- **Note 3:** With Rule 2, keep in mind that β in FIRST(β) **may** denote an entire **string** of terminals and non-terminals.

1. Put \$ in FOLLOW(S).
2. If $A ::= \alpha B \beta$:
   (a) Put FIRST(β) $- \{\epsilon\}$ in FOLLOW(B)
   (b) if $\beta = \epsilon$ (i.e., $A ::= \alpha B$) or $\epsilon \in$ FIRST(β) (i.e., $\beta \rightarrow^* \epsilon$) then put FOLLOW($A$) in FOLLOW($B$)

   Repeat Step 2 until no more additions can be made.

2.1 Example

We consider again the context-free grammar from Section 1. The FIRST-sets for the noterminals of this grammar are given in the following table.

S' ::= S
S ::= AB
S ::= C
A ::= ef
A ::= ϵ
B ::= hg
C ::= DD
C ::= fi
D ::= g

Non-Terminal	FIRST
S'	$\{e, f, g, h\}$
S	$\{e, f, g, h\}$
A	$\{e, \epsilon\}$
B	$\{h\}$
C	$\{f, g\}$
D	$\{g\}$

We then initialize the FOLLOW-sets and proceed according to Steps 1 and 2 stated above.

(1)		Put \$ in FOLLOW(S').
(2)	S' ::= S:	Put FOLLOW(S'), i.e., {\$} in FOLLOW(S).
(3)	S ::= AB:	Put FIRST(B), i.e., {h} in FOLLOW(A).
		Put FOLLOW(S), i.e., {\$}, in FOLLOW(B).
(4)	S ::= C:	Put FOLLOW(S), i.e., {\$}, in FOLLOW(C).
(5)	A ::= ef:	no addition.
(6)	A ::= ϵ:	no addition.
(7)	B ::= hg:	no addition.
(8)	C ::= DD:	Put FIRST(D), i.e., {g}, in FOLLOW(D).
		Put FOLLOW(C), i.e., {\$}, in FOLLOW(D).
(9)	C ::= fi:	no addition.
(10)	D ::= g:	no addition.

Non-Terminal	FOLLOW
S'	{\$}
S	{\$}
A	{h}
B	{\$}
C	{\$}
D	{g, \$}

Further iterations do not yield additional changes.

3 Homework Example

Compute the FIRST- and FOLLOW-sets for the following grammar.

S ::= aABbCD
S ::= ϵ
A ::= ASd
A ::= ϵ
B ::= SAc
B ::= eC
B ::= ϵ
C ::= Sf
C ::= Cg
C ::= ϵ
D ::= aBD
D ::= ϵ