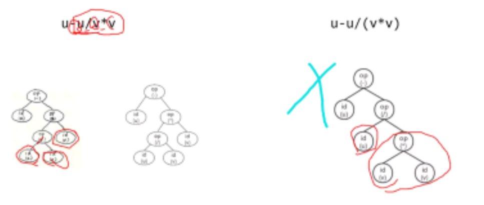
罗 week8 理论课

2020年4月13日 星期一 下午1:56

不想改写算法就可以人工向前看即可。



eg.
First (F) = { (, i }
First (T) = { (, i }
First (E) = { (, i }

e.g.
(E) = { (, i }
(E') = { +, ε }
(T) = { (, i }
(T') = { *, ε }
(F) = { (, i }

e.g.
(S) = { a, d }
(A) = { e, a, d, ε }
(B) = { a, d, c, ε }
(D) = { a, d, ε }

Follow
First(e) ∩ Follow(A) + {First(D) − ε} + $ = Follow(A) + a, d, $ = a, b, c, d, e, $
c, b
a, d,
a, d, c, b, e.

$S \to aAbDe \mid d$
$A \to BSD \mid e$
$B \to SAc \mid cD \mid \varepsilon$
$D \to Se \mid \varepsilon$

只有非终结符上定义非终结符.
FOLLOW(A), $A \in V_N$
跟随A的右侧出现的终结符的集合.
$S \overset{*}{\Rightarrow} \alpha A a \beta$
$S \overset{*}{\Rightarrow} \ldots Aa \ldots$

FOLLOW(A) = { a | $S \Rightarrow^* \alpha A a \beta$, $a \in V_T$ }

If $S \Rightarrow^* \ldots A$, then $\$ \in$ FOLLOW(A).

用递归的思路计算

FOLLOW(A)是文法G的某些句型中紧随在A之后出现的终结符或$。

*终结符的First集合就是自己.

1. $ 放在 Follow (S) 中
2. if $A \to \alpha B \beta$, then 将{First(β) − ε}放入Follow(B)中.
3. if $A \to \alpha B$ or $A \to \alpha B \beta$, First(β) 包含 ε, 将Follow (A) 放入Follow (B) 中

→ 用 αB 替换A后.出现在αB后的,也在A后

First 与 Follow 的算法伪代码.

结束标记.

E → TE′	FIRST(E) ={ (, i }	), $
E′ → +TE′ | ε	FIRST(E′)={ +, ε }	$)
T → FT′	FIRST(T) ={ (, i }	+ $)
T′ → *FT′ | ε	FIRST(T′)={ *, ε }	+ $)
F → (E) | i	FIRST(F) ={ (, i }	* + $)

statement → *if-stmt* | *repeat-stmt* | *assign-stmt* | *read-stmt* | *write-stmt*

E → TE′
E′ → +TE′ | ε　　E → TE′
T → FT′
T′ → *FT′ | ε　　E → FT′E′
F → (E) | i

i + i

→ iT′E′
→ iE′
→ i+TE′
→ i+FT′E′
→ i+iT′E′
→ i+iE′
→ i+i

$A \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$

意义: 若 First(αi) ∩ First(αj) = ∅, 且 a ∈ First(αi) then A → αi

若 ε : A → ε, or ε ∈ First(αi)

LL(1), 从左→右扫描, 以+最左推导

S → aSe | B
B → bBe | C
C → cCe | d

S → (S) S | ε
First(S) = { (, ε }
Follow(S) = { $,), (}
Follow(S) = { $,) }
产生冲突
S → S(S)S | ε
First(S) = { ε, (}
Follow(S) = { $, (,) }

→ First不相交

- A grammar G is LL(l) if and only if the following conditions hold:
  1. For every $A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$, $FIRST(\alpha_i) \cap FIRST(\alpha_j) = \phi$, where $1 \le i, j \le n$, $i \ne j$.
  2. For every $A \in V_N$, if $\varepsilon \in FIRST(A)$, then $FIRST(A) \cap FOLLOW(A) = \phi$.

两两不相交

A grammar G is LL(1) if and only if whenever $A \to \alpha \mid \beta$ are two distinct productions of G, the following conditions hold:

1. For no terminal a do both α and β derive strings beginning with a.
2. At most one of α and β can derive the empty string.
3. If $\beta \overset{*}{\Rightarrow} \epsilon$, then α does not derive any string beginning with a terminal in FOLLOW(A). Likewise, if $\alpha \overset{*}{\Rightarrow} \epsilon$, then β does not derive any string beginning with a terminal in FOLLOW(A).

eg.
S → ABBA　　First(S) = {a, b, ε}　　Follow(S) = {$}
A → a | ε　　First(a) = {a, ε}　　Follow(A) = {$, b} {$, a, b}
B → b | ε　　First(b) = {b, ε}　　Follow(B) = {$, a} {$, a, b}

First(BBA) − ε + First(A …

左递归文法不会是 LL(1)文法
但经过修改可以得到需要的文法.

S → iEt SS′ | a　　First(S) = i, a　　Follow(S) = ($, e)
S′ → eS | ε　　First(S′) = e, ε　　Follow(S′) = ($, e)
E → b　　First(E) = b　　Follow(E) = t,

First (S′) ∩ Follow (S′) ≠ ∅ → 不是 LL(1)文法.

- 如果一个文法是LL(1)文法，则可以进行确定的自顶向下分析。
- 非终结符A在分析过程中，面临的输入符号为 a，A的产生式为
  $A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$
  1. 若 $a \in FIRST(\alpha_i)$，则选择 $A \to \alpha_i$;
  2. 若 $a \notin FIRST(\alpha_i)$，则:
     ① 若 $\varepsilon \in FIRST(\alpha_i)$且 $a \in FOLLOW(A)$，则选择 $A \to \varepsilon$;
     ② 否则，a 的出现是一种语法错误。

每个预测的都是相应

Production	FIRST	FOLLOW
S → iEtSS′ | a		
S′ → eS | ε	{ e, ε }	{ e, $ }
E → b		

就是这个推导里面算follow(S′),
按照规则follow(S′) = follow(S)
然后算follow(S), follow(S) = first(S′) − {ε} + follow(S′) = {e} + follow(S′)

- 非终结符A在分析过程中，面临的输入符号为 a，A的产生式为

$$A \to \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$$

1. 若 $a \in FIRST(\alpha_i)$，则选择 $A \to \alpha_i$；
2. 若 $a \notin FIRST(\alpha_i)$，则：

① 若 $\varepsilon \in FIRST(\alpha_i)$且 $a \in FOLLOW(A)$，则选择 $A \to \varepsilon$；

② 否则，a 的出现是一种语法错误。

有一个预测分析表.

Production	FIRST	FOLLOW
S → iEtSS′ | a		
S′ → eS | ε	{ e, ε }	{ e, $ }
E → b		

就是这个推导里面算follow(S′),
按照规则follow(S′) = follow(S)
然后算follow(S), follow(S) = first(S′) - {ε} + follow(S′) = {e} + follow(S′)
follow(S) = {e} + follow(S′),又因为前面得到S和S′的follow集合相等

- Algorithm 4.31 can be applied to any grammar G to produce a parsing table M.
- For every LL(1) grammar, each parsing-table entry uniquely identifies a production or signals an error.
- For some grammars, however, M may have some entries that are multiply defined. For example, if G is left-recursive, or if G is ambiguous.
- 并非所有的文法都能改写为LL(1)文法。

NON-TERMINAL	id	+	*	(	)	$
E	E → TE′			E → TE′		
E′		E′ → +TE′			E′ → ε	E′ → ε
T	T → FT′			T → FT′		
T′		T′ → ε	T′ → *FT′		T′ → ε	T′ → ε
F	F → id			F → (E)		

Figure 4.17: Parsing table M for Example 4.32

Steps	Parsing Stack	Input	Action
1	$E	$i_1*i_2+i_3\$$	E→TE′
2	$ET	$i_1*i_2+i_3\$$	T→FT′
3	$E′T′F	$i_1*i_2+i_3\$$	F→i
4	$E′T′i	$i_1*i_2+i_3\$$	match
5	$E′T′	$*i_2+i_3\$$	T′→*FT′
6	$E′T′F*	$*i_2+i_3\$$	match
7	$E′T′F	$i_2+i_3\$$	F→i
8	$E′T′i	$i_2+i_3\$$	match
9	$E′T′	$+i_3\$$	T′→ε
10	$E′	$+i_3\$$	E′→+TE′

	i	+	*	(	)	$
E	E → TE′			E → TE′		
E′		E′ → +TE′			E′ → ε	E′ → ε
T	T → FT′			T → FT′		
T′		T′ → ε	T′ → *FT′		T′ → ε	T′ → ε
F	F → i			F → (E)		

E′	$+i_3\$$	E′→+TE′
E′T+	$+i_3\$$	match
E′T	$i_3\$$	T→FT′
E′T′F	$i\$$	F→i
E′T′i	$i\$$	match.
$E′T′	$	T′→ε
$E′	$	E′→ε
$	$	accept.