4.6.2 项和LR(0)自动机 →∪Σεαβ∈∉•

4.6.4 构造SLR语法分析表

练习: 1. 已知下面文法G:

S → L = R | R

L → \* R | id

R → L

(1) 求文法G的拓广文法G';

S' → S

S → L = R | R

L → \* R | id

R → L

(2)求G'对应的LR(0)自动机;

(3) 生成对应的 SLR分析表.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Action | | | | GOTO | | |
|  | = | \* | id | $ | S | L | R |
| 0 |  | s4 | s5 |  | 1 | 2 | 3 |
| 1 |  |  |  | acc |  |  |  |
| 2 | s6/r5 |  |  | r5 |  |  |  |
| 3 |  |  |  | r2 |  |  |  |
| 4 |  | s4 | s5 |  |  | 8 | 7 |
| 5 | r4 |  |  | r4 |  |  |  |
| 6 |  | s7 | s5 |  |  | 8 | 9 |
| 7 | r3 |  |  | r3 |  |  |  |
| 8 | r5 |  |  | r5 |  |  |  |
| 9 |  |  |  | r1 |  |  |  |

为什么要对文法G进行拓广?

保证文法开始符号不出现在任何产生式右部.

E → E + T

F → ( E )

即使某个时候可以归约为E, 也不能断定可以结束分析？

E' → E

根本目的是保证文法的起始符号 FOLLOW(?) = {$}

为什么称为LR(0)自动机?

构造自动机时，对就的项又称为LR(0)项,

即向前看符号为 0(即不必知道当前输入符号).

由这样的LR(0)项构成的自动机也即LR(0)自动机

为什么称LR(0)自动机为 识别文法活前缀/可行前缀的DFA?

最右句型的前缀，并且不超过句柄的最右端

I0 (状态0开始状态)

所有状态都是接受状态 (I0, I1, …. )

最右句柄总是出现在栈的最顶端

最右句型:

1. E + T E E + E + T

2. (id + id) \* id ( ( id

3. E + ( E + id ) E E + E + ( E + ( E E + ( E + ..

什么是LR(0)文法, 什么是SLR(1)文法?

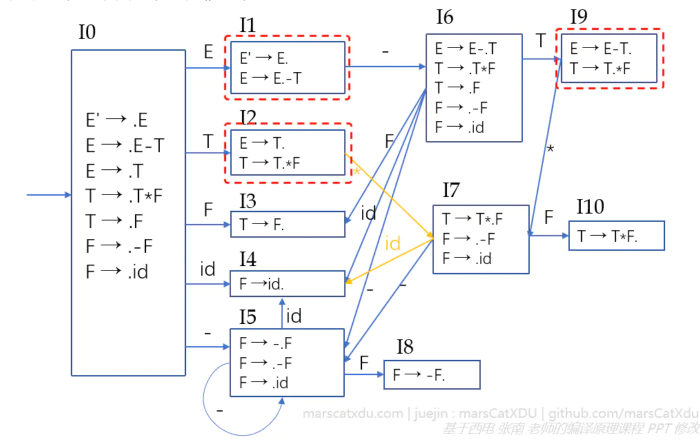
若一个文法 G 的拓广文法 G' 的识别活前缀的

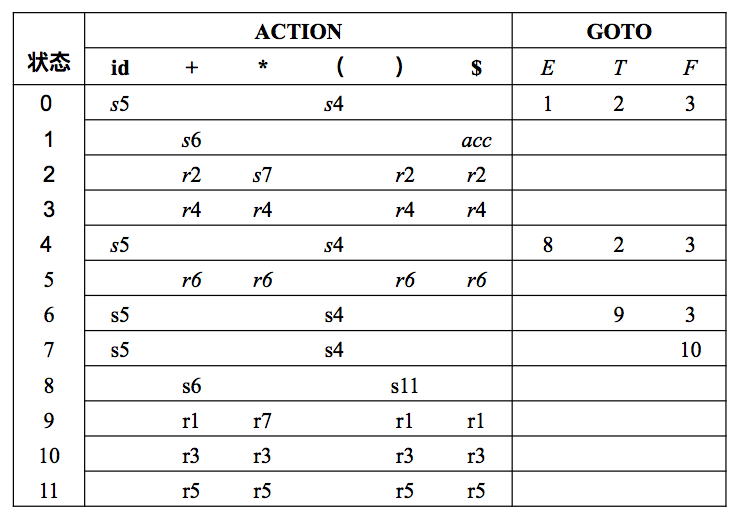
自动机中的每个状态（项目集）均不存在下述情况：

1. 既有移进项目又有规约项目（移进-规约冲突）；

2. 含有多个规约项目（规约-规约冲突），

则称 G 为一个 LR(0) 文法





例(1)已下下面文法G:

E → aA | bB

A → cA | d

B → cB | d

FOLLOW(E') = {$}

FOLLOW (E) = {$}

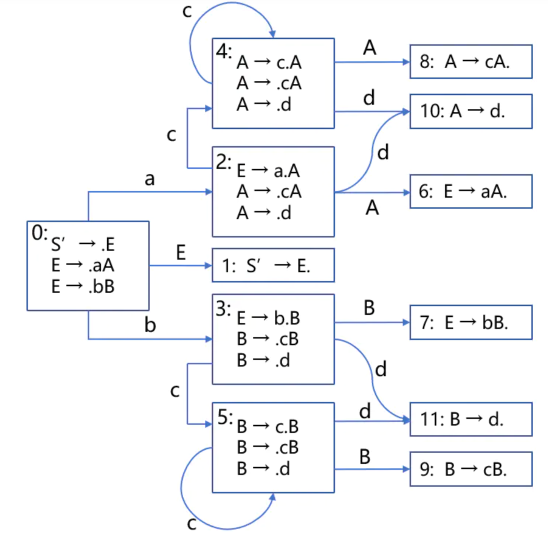
FOLLOW(A) = {$}

FOLLOW(B) = {$}

(1) 求文法G的拓广文法G';

(2)求G'对应的LR(0)自动机;

(3) 生成对应的 LR(0)分析表.



任何二义性文法都不是SLR(1)文法

例(2) 悬空else

A → S

S → iCtSS' | a

S' → eS | ε

C → b

任何非二义性文法都是SLR(1)文法 错

不是二义文法的非 SLR(1) 文法

任何二义文法都不可能是LL(1)文法，也不可能是SLR(1)文法

例(3) 说明文法

S → AaAb | BbBa

A → ε

B → ε

是LL(1)文法, 但不是SLR(1)文法.

例(4) 求文法

S → ( L ) | a

L → L, S | S

的活前缀的一个DFA

例(5) 求文法

S → Aa | bAc | dc | bda

A → d

活前缀的DFA. 并构造该文法的SLR(1)分析表，并说明该文法是否为SLR(1)文法.

例(5) 证明文法

S → AaAb | BbBa

A → ε

B → ε

是LL(1)文法, 但不是SLR(1)文法.

0: S’ → S

1: S → AaAb

2: S → BbBa

3: A → ε

4: B → ε

SLR(1)分析表

I0: S' → • S, S → •AaAb, S → • BbBa, A → •, B → •;

I0的操作:

对于下一个符号a或b时，进行A归约;

对于下一个符号a或b时，进行B归约;

归约-归约冲突,

即当下一个符号是a时，进行A归约, 进行B归约(完全可以丢弃这个选项);

当前未匹配任意字符, 符号栈中的内容是$(即空串), 如果进行B归约, 符号栈中的内容是 $ B, 下一个输入符号是a; 由于不能单独将B进行归约, 那么下一个动作必定是移进a, 移进a之后符号栈的内容是 $ B a , 但是Ba 不是一个活前缀, 所以这个一定不能够生成正确的推导.

LR(1)

I0: S' → • S, $; S → •AaAb, $; S → • BbBa, $; A→ •, a; B→ •, b

当下一个输入符号是a时, 采用A→ •进行归约;

当下一个输入符号是b时, 采用B→ •进行归约;

即当下一个符号是b时，进行A归约, 进行B归约;