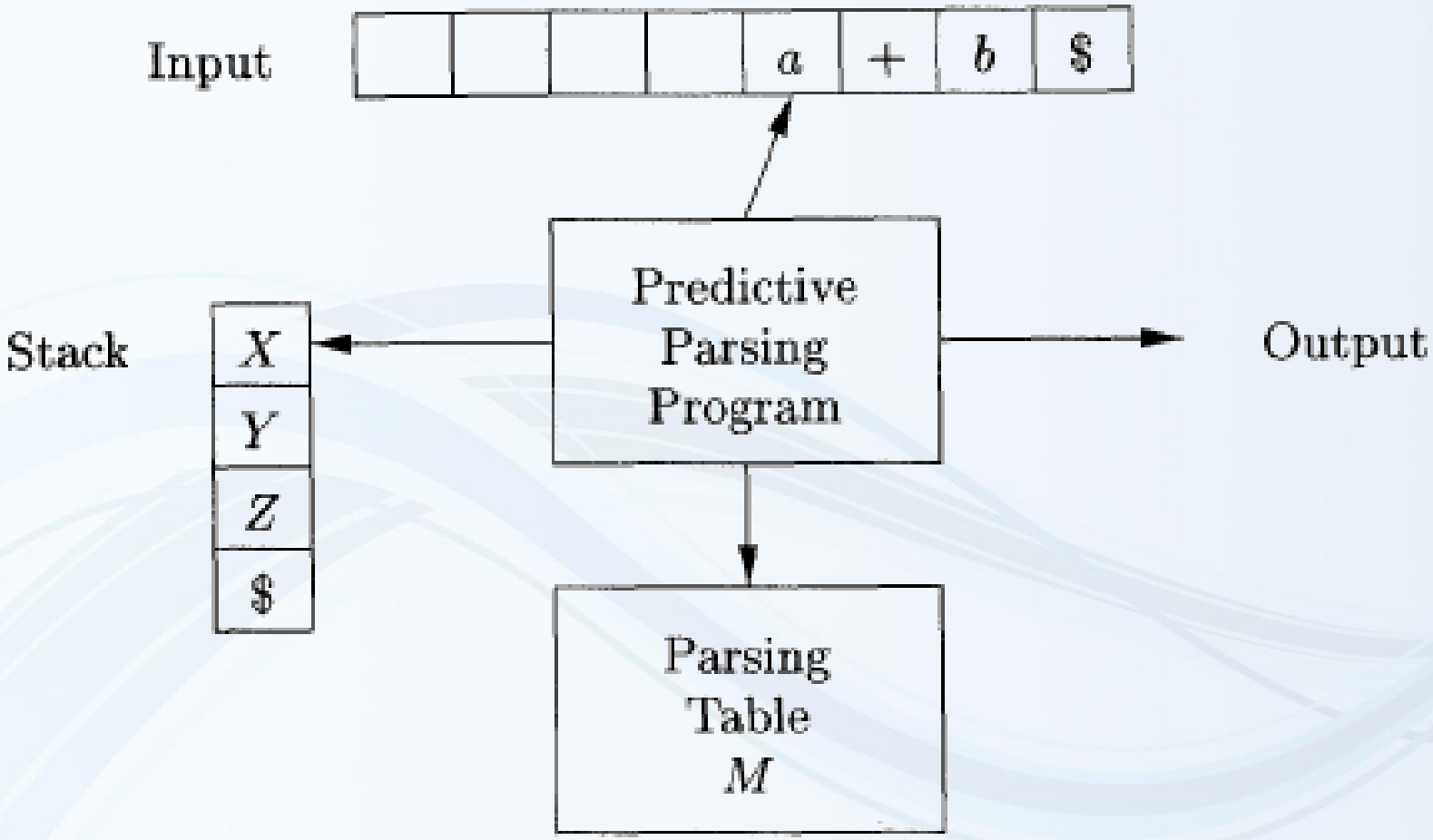
分析表驱动的预测分析器（LL(1)分析器）



预测分析算法(LL(1)分析算法)

让*ip*指向第一个输入符号a;　//输入结尾要加上结束标志$

指向栈顶X;　//开始运行时,栈为S$

```
while ( X≠ $ ){
    if ( X is a ) pop栈顶，并让 ip指向下一个输入符号;
    else if ( X 为终结符 ) error( );
        else if ( M[X, a] 是错误项目 ) error( );
            else if ( M[X,a] = X→Y1Y2…Yk ) {
                    输出规则 X→Y1Y2…Yk;
                    pop 栈顶;
                    push Yk, Yk-1,. . . , Y1 onto the stack, with Y1 on top;
                }
    指向栈顶X;
}
```

预测分析过程（例）

NON-TERMINAL	INPUT SYMBOL					
	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$	$T' \to \epsilon$
F	$F \to \mathbf{id}$			$F \to (E)$		

输入：id+id*id$

预测分析过程（例）

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	output $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	output $T \to FT'$
	$\mathbf{id}\ T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	output $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match **id**
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	output $T' \to \epsilon$
$\mathbf{id}$	$+\,TE'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	output $E' \to +\,TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$
$\mathbf{id} +$	$FT'E'\$$	$\mathbf{id} * \mathbf{id}\$$	output $T \to FT'$
$\mathbf{id} +$	$\mathbf{id}\ T'E'\$$	$\mathbf{id} * \mathbf{id}\$$	output $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$T'E'\$$	$*\,\mathbf{id}\$$	match **id**
$\mathbf{id} + \mathbf{id}$	$*\,FT'E'\$$	$*\,\mathbf{id}\$$	output $T' \to *\,FT'$
$\mathbf{id} + \mathbf{id} *$	$FT'E'\$$	$\mathbf{id}\$$	match $*$
$\mathbf{id} + \mathbf{id} *$	$\mathbf{id}\ T'E'\$$	$\mathbf{id}\$$	output $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$T'E'\$$	$\$$	match **id**
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$E'\$$	$\$$	output $T' \to \epsilon$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$\$$	$\$$	output $E' \to \epsilon$

LL(1)分析（例）

对于文法S→(L)|a　L →L,S|S

(1)消除左递归

(2)构造预测分析器

(3)分析(a,(a,a))，并给出分析树。

如果顺序设为L,S 会怎么样？

解：设定非终结符的顺序为S,L。

S无左递归

L存在逆序L→S，将S代入得：L→L,S|(L)|a

L存在直接左递归L→L,S，引入非终结符L’，消除左递归得：

L→(L)L’|aL’　　L’ →,SL’| ε

故文法为：　S→(L)|a　　L→(L)L’|aL’　　L’ →,SL’| ε

LL(1)分析（例）

FOLLOW(L')=FOLLOW(L)={) }

可构建预测分析表如下

非终结符	输入符号				
	(	)	a	,	$
S	S→(L)		S→a		
L	L→(L)L'		L→aL'		
L'		L' →ε		L' →,SL'	

S→(L)|a　　L→(L)L'|aL'　　L' →,SL'| ε

LL(1)分析（例）

Matched	Stack	Input	Action
	S$	(a,(a,a))$	
	(L)$	(a,(a,a))$	output S→(L)
(	L)$	a,(a,a))$	Match (
(	aL')$	a,(a,a))$	Output L→aL'
(a	L')$	,(a,a))$	Match a
(a	,SL')$	,(a,a))$	Output L' →,SL'
(a,	SL')$	(a,a))$	Match ,
(a,	(L)L')$	(a,a))$	output S→(L)
(a,(	L)L')$	a,a))$	Match (
(a,(	aL')L')$	a,a))$	Output L→aL'
(a,(a	L')L')$	,a))$	Match a

LL(1)分析（例）

Matched	Stack	Input	Action
(a,(a	,SL')L')$	,a))$	Output L' →,SL'
(a,(a,	SL')L')$	a))$	Match ,
(a,(a,	aL')L')$	a))$	Output S→a
(a,(a,a	L')L')$	))$	Match a
(a,(a,a	)L')$	))$	Output L' →ε
(a,(a,a)	L')$	)$	Match)
(a,(a,a)	)$	)$	Output L' →ε
(a,(a,a))	$	$	Match)

分析树略

预测分析中的错误恢复（恐慌模式）

- 忽略输入中的符号，直到出现栈顶的同步词法单元（也称同步符号）。
  - 同步词法单元集合常简称同步集合。
  - 终结符a的同步符号就是a，出现同步符时弹栈+忽略输入。
  - 非终结符A的同步符号是FOLLOW(A)中的所有符号，出现同步符时仅弹栈。

考虑错误处理的预测分析表

非终结符号	输入符号					
	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$	synch	synch
E'		$E' \to +TE'$			$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to FT'$	synch		$T \to FT'$	synch	synch
T'		$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$	$T' \to \epsilon$
F	$F \to$ id	synch	synch	$F \to (E)$	synch	synch

FOLLOW(E) = {$,)}　　FOLLOW(E') = {$,)}　　FOLLOW(T) = {$, +,)}
FOLLOW(T') = {$, +,)}　FOLLOW(F) = {$, +, *,)}

结合错误恢复的预测分析过程

stack	input	action
E$	+id*+id$	
E$	id*+id$	Error, skip +
TE'$	id*+id$	Output E →TE'
FT'E'$	id*+id$	Output T →FT'
idT'E'$	id*+id$	Output F →id
T'E'$	*+id$	Match id
*FT'E'$	*+id$	Output T'→*FT'
FT'E'$	+id$	Match *
T'E'$	+id$	Error M[F,+]=synch
E'$	+id$	Output T'→ ε
+TE'$	+id$	Output E'→ +TE'

动作1：分析表空白，忽略输入

动作2：分析表同步符，弹栈

结合错误恢复的预测分析过程（续）

stack	input	action
TE'$	id$	Match +
FT'E'$	id$	Output T →FT'
idT'E'$	id$	Output F →id
T'E'$	$	Match id
E'$	$	Output T'→ ε
$	$	Output E'→ ε

LL(1)分析（例续）

对于文法S→(L)|a　　L→(L)L’|aL’　　L’ →,SL’| ε

（1）构建考虑错误恢复的预测分析表

（2）给出(aa,(a,))的分析过程

解：FOLLOW(S)={$}∪(FIRST(L’)-{ε})∪FOLLOW(L’)={$, , ,)}

FOLLOW(L’)=FOLLOW(L)={)}

构建预测分析表

非终结符	输入符号				
	(	)	a	,	$
S	S→(L)	synch	S→a	synch	synch
L	L→(L)L’	synch	L→aL’		
L’		L’ →ε		L’ →,SL’	

LL(1)分析（例续）

stack	input	action
S$	(aa,(a,))$	
(L)$	(aa,(a,))$	Output S→(L)
L)$	aa,(a,))$	Match (
aL')$	aa,(a,))$	Output L→aL'
L')$	a,(a,))$	Match a
L')$	,(a,))$	Error, skip a
,SL')$	,(a,))$	Output L' →,SL'
SL')$	(a,))$	Match ,
(L)L')$	(a,))$	Output S→(L)
L)L')$	a,))$	Match (
aL')L')$	a,))$	Output L→aL'

LL(1)分析（例续）

stack	input	action
L')L')$	,))$	Match a
,SL')L')$	,))$	Output L' →,SL'
SL')L')$	))$	Match ,
L')L')$	))$	Error M[S,)]=synch
)L')$	))$	Output L' →ε
L')$	)$	Match)
)$	)$	Output L' →ε
$	$	Match)

4.5 自底向上的语法分析

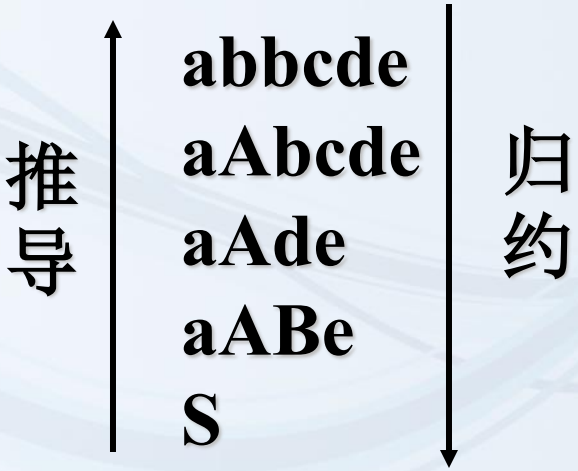
- 自顶向下分析的逆过程：把一个串归约到文法的开始符.
  - 归约：先将一个子串与某规则的右部匹配，然后将该子串替换为该规则的左部。

文法:

$S \rightarrow aABe$

$A \rightarrow Abc \mid b$

$B \rightarrow d$



最右推导:

$S \Rightarrow aABe \Rightarrow aAde \Rightarrow aAbcde \Rightarrow abbcde$

句柄(*Handle*)

- 与某规则右部一致的子串，并且将它归约成该规则的左部就是某最右推导的逆过程。
  - 一个句型可能有多个不同的句柄（二义文法）
  - 一个无二义文法的右句型只有唯一的句柄。
    - ✓右句型(规范句型)：通过最右推导（也称规范推导）得到的句型。

短语（*Phrase*）

- 对于文法G[S]，αβδ是G的一个句型，如果 $S \overset{*}{\Rightarrow} \alpha A \delta$，且：
  - $A \overset{+}{\Rightarrow} \beta$，则称β是句型αβδ相对于A的短语。
  - $A \rightarrow \beta$，则称β是句型αβδ相对于$A \rightarrow \beta$的直接短语（简单短语）。

对于最右推导过程：

$E \Rightarrow E+E \Rightarrow E+E*E \Rightarrow E+E*i \Rightarrow E+i*i$

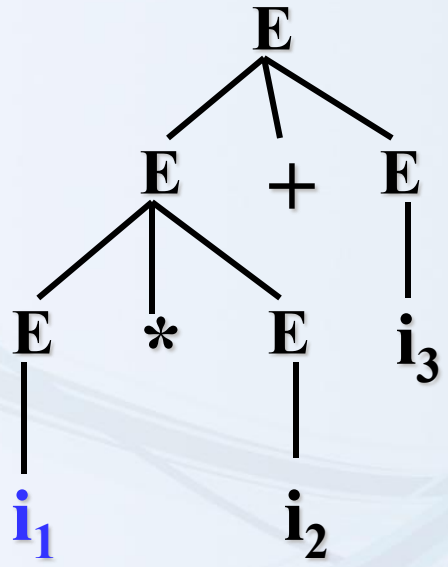
i是句型E+i*i相对于非终结符E的短语。

i是句型E+i*i相对于规则$E \rightarrow i$的直接短语。

短语和句柄（分析树视角）

- 子树的末端结点符号串是相对于子树根的短语
  - 一个内部结点对应一棵子树
  - 一棵子树对应一个短语
- 直接短语对应的子树高度为1。

一个规范句型的最左直接短语称为该句型的句柄



注：有5个短语：i_1，i_2，i_3，i_1*i_2，$i_1*i_2+i_3$

短语和句柄（例）

对于文法S → SS+ | SS* | a

试指出右句型SSS+a*+的句柄

解法一：S ⇒ SS+ ⇒ SSS*+ ⇒ SSa*+

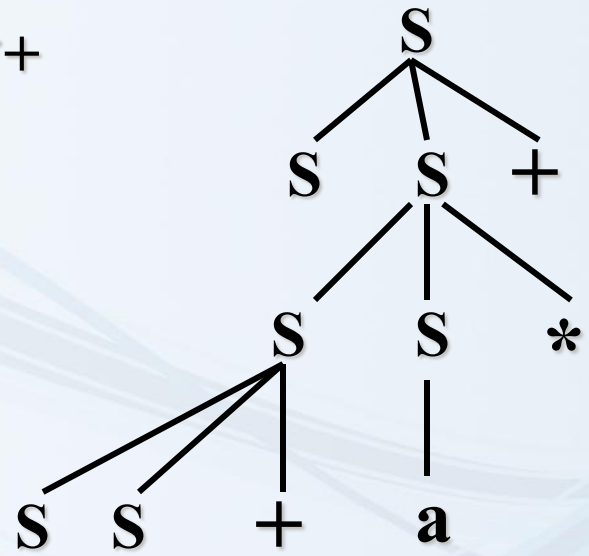
⇒ SSS+a*+

显然句柄为SS+

解法二（推荐！）：

可构建分析树如右：

显然句柄为SS+



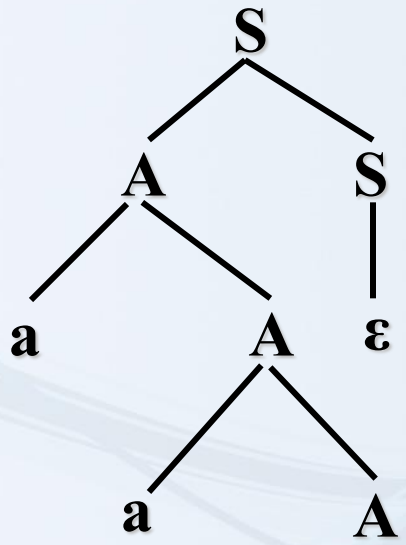
本句型有哪些短语，直接短语？

短语和句柄（例2）

对于文法S → AS | ε　A→aA | b

试指出右句型aaA的短语，简单短语和句柄

解：可构建分析树如右：



短语：aaA，aA, ε

简单短语：aA, ε

句柄：aA

Bottom-Up Parsing

For the given grammar, what is the correct series of reductions for the string: -(id + id) + id

$E \rightarrow E' \mid E' + E$

$E' \rightarrow -E' \mid id \mid (E)$

A

-(id + id) + id
-(id + E') + id
-(id + E) + id
-(E' + E) + id
-(E) + id
-E' + id
E' + id
E' + E'
E' + E
E

B

-(id + id) + id
-(E' + id) + id
-(E' + E') + id
-(E' + E) + id
-(E) + id
-E' + id
E' + id
E' + E'
E' + E
E

C

-(id + id) + id
-(E' + id) + id
-(E' + E') + id
-(E' + E') + E'
-(E' + E) + E'
-(E) + E'
-E' + E'
E' + E'
E' + E
E

D

-(id + id) + id
-(id + id) + E'
-(id + id) + E
-(E' + id) + E
-(E' + E') + E
-(E' + E) + E
-(E) + E
-E' + E
E' + E
E

最左归约(例)

有一语法制导定义如下：

产生式	语义规则
S→SS+	**print “1”**
S→SS*	**print “2”**
S→a	**print “3”**

若输入序列为**aa+aa*+**，且采用自底向上（最左归约）的分析方法，则输出序列为:

3313321

可行前缀与可归前缀

- 前缀：符号串的首部　　（复习）

  abc　　ε, a, ab, abc

- 活前缀：对于文法G[S]的一个规范句型αβ, 其中β为终结符号串, 如果α不含句柄后的任何符号,则称α为活前缀（也称可行前缀）
  - 如果α是含有句柄的活前缀，则称α为可归前缀。

可归前缀是最长的活前缀

可行前缀与可归前缀

P163　4.6.5

可以出现在一个移入－归约语法分析器的栈中的最右句型前缀被称为可行前缀（viable prefix）。它们的定义如下：一个可行前缀是一个最右句型的前缀，并且它没有越过该最右句型的最右句柄的右端。根据这个定义，我们总是可以在一个可行前缀之后增加一些终结符号来得到一个最右句型。

可行前缀与可归前缀

文法G[S]:

S → CbBA　　A → Aab|ab　　B → C|Db

C → a　　D → a

句子ababab的规范推导过程为：

S⇒CbBA⇒CbBab⇒CbDbab⇒Cbabab⇒ababab

(1)句型ababab的活前缀是ε、a, 可归前缀是a.

(2)句型Cbabab的活前缀是ε、C、Cb、Cba, 可归前缀是Cba.

(3)句型CbDbab的活前缀是ε、C、Cb、 CbD、CbDb, 可归前缀是CbDb.