Формални Езици и Езикови Процесори
ТУ, кат. КС, летен семестър 2012

Лекция 12

Тема:

Синтактичен Анализ (Низходящ LL(k) метод)

Съдържание:

- Формални методи
- Низходяща Top-down СА стратегия
- Проба-грешка, груба сила /brute force/
- LL(k) СА анализ
- Demo програми
- Примери

СА – формални методи

Дадено: КСГ граматика G, описваща език L(G) и входен низ string ω.

Задача: Изречение ли е вх. низ в езика L(G)? Принадлежи ли низът на езика $\omega \in L(G)$?

Ако Да, да се построи syntax (parse) tree.

Иначе, грешка

Две стратегии за строеж на синтактично дърво:

Низходяща top-down (from root to leaves);

Възходяща bottom-up (from leaves to root).

Top-down низходящ подход

Постановка :

Входна лента

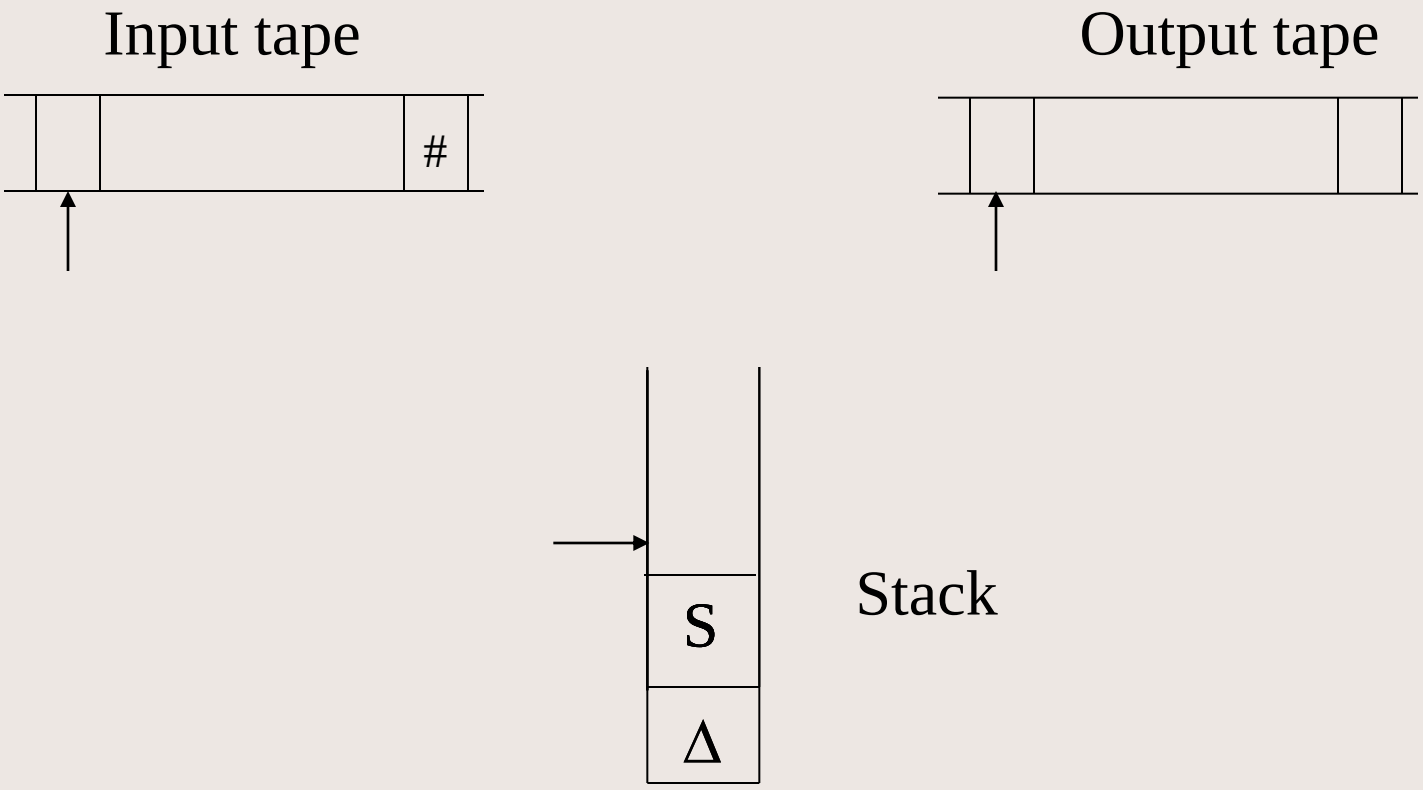
Изх. лента

Стек

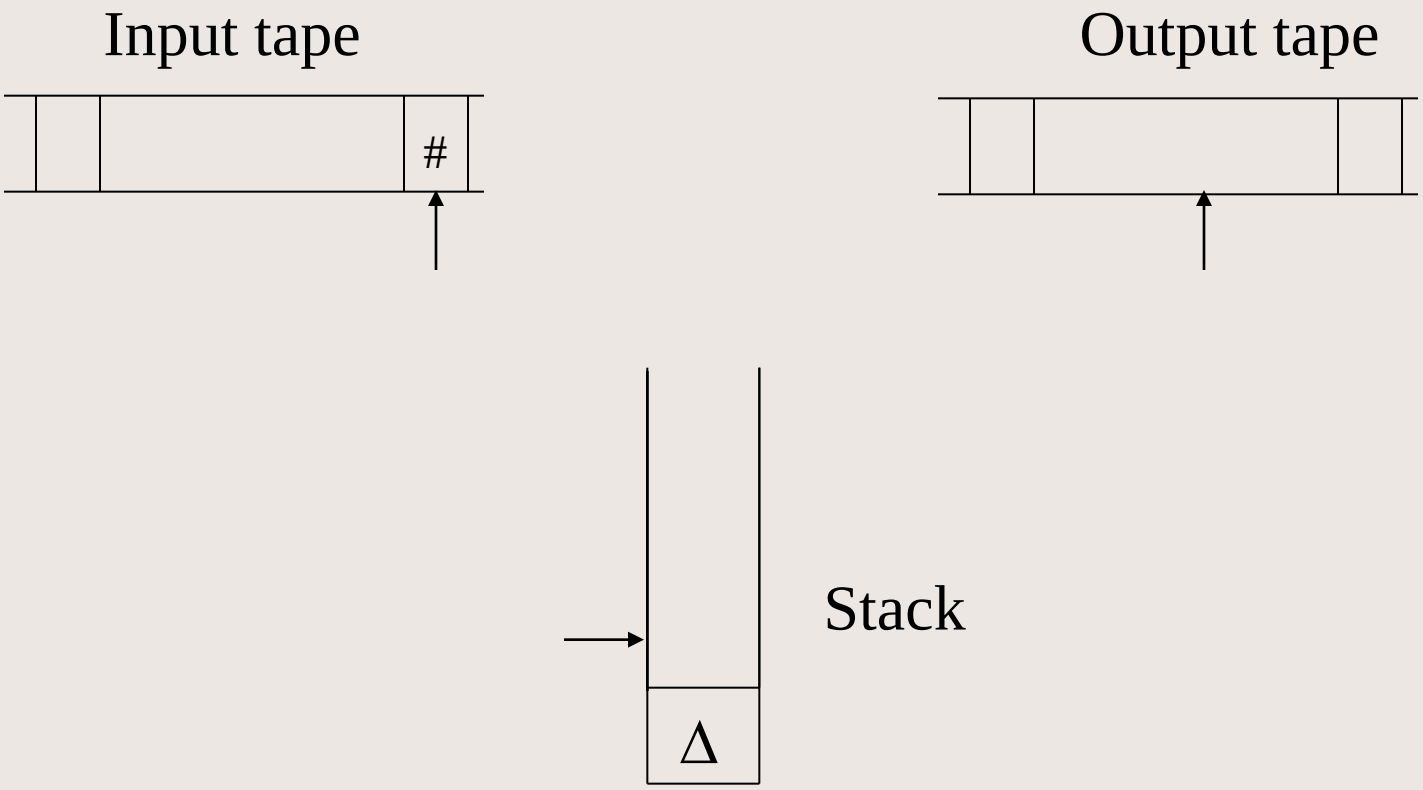
Статус преди СА
скан показалец
top stack: старт. нетерминал

Статус след СА
скан показалец
празен стек

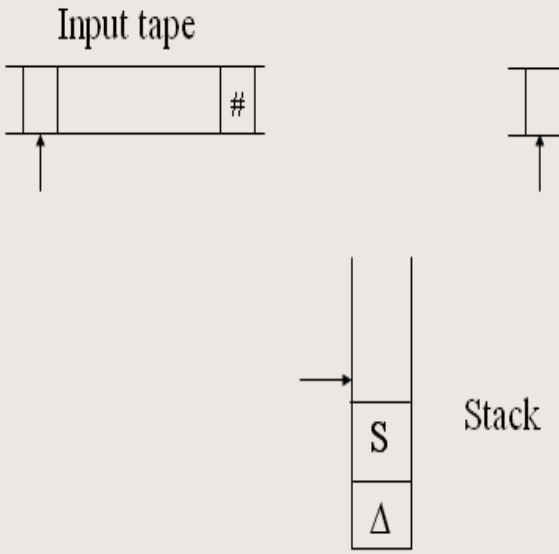
Top-down SA strategy
Статус преди CA



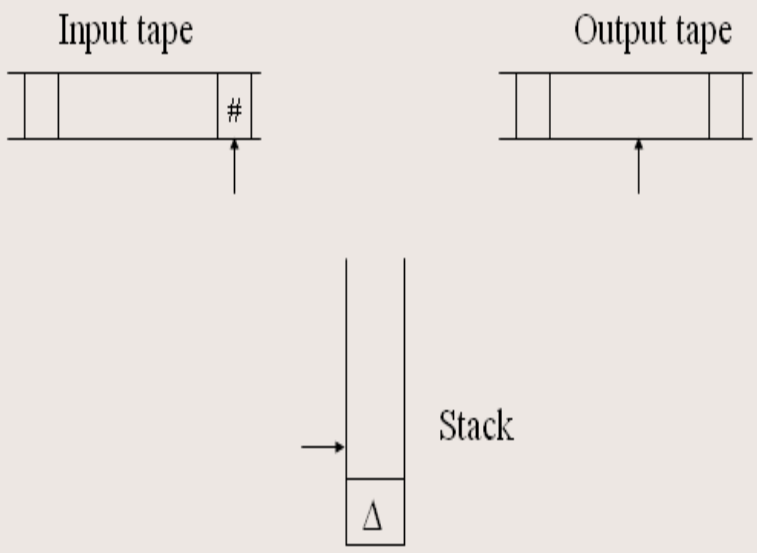
Top-down SA strategy
Статус след успешен CA



Top-down SA strategy status before parsing



Top-down SA strategy status after successful parsing



СА – постановка на проблем

Дадено: КСГ граматика G, описваща език L(G) и входен низ string ω.

Задача: Изречение ли е вх. низ в езика L(G)? Принадлежи ли низът на езика $\omega \in L(G)$?

Ако Да, да се построи syntax (parse) tree.

Иначе, грешка

Как се строи синт. дърво? В каква форма се представя? Генериране и съхранение на продукции такива, че приложени в прав ред ще потвърдят пораждането/деривацията от S до ω или приложени в обратен ред ще редуцират/свият входния низ от ω до стартовия нетерминал S.

СА като функция

Дадено: КСГ граматика G, описваща език L(G) и входен низ string ω.

Задачата за СА представена като функция:

$$SA(\omega) = k_1\ k_2\ k_3\ \ldots\ k_n$$

Вх. Низ ω е аргумент на функцията.

Функцията SA връща последователност от номера на продукции (т.е абстрактна форма на дървото на разбора).

Пример 1

a + a * a

Пример: Top-down СА на АИ *a+a*a*

Дадено: еднозначна КСГ на АИ

1. $E \rightarrow T$
2. $E \rightarrow E + T$
3. $T \rightarrow F$
4. $T \rightarrow T * F$
5. $F \rightarrow a$
6. $F \rightarrow (E)$

$$E \rightarrow \underline{E} + T \rightarrow \underline{T} + T \rightarrow \underline{F} + T \rightarrow a + \underline{T} \rightarrow$$

$$a + \underline{T} * F \rightarrow a + \underline{F} * F \rightarrow a + a * \underline{F} \rightarrow a + a * a$$

SA (a+a*a) = 2 1 3 5 4 3 5 5

Ляв каноничен анализ

a + a * a

$E \rightarrow E + T$

a + a * a

$E \rightarrow E + T$

$\rightarrow T + T$

a + a * a

$E \rightarrow E + T$

$\rightarrow T + T$

$\rightarrow F + T$

a + a * a

$$
\begin{aligned}
E &\rightarrow E + T \\
&\rightarrow T + T \\
&\rightarrow F + T \\
&\rightarrow a + T
\end{aligned}
$$

a + a * a

$E \rightarrow E + T$

$\rightarrow T + T$

$\rightarrow F + T$

$\rightarrow a + T$

$\rightarrow a + T * F$

a + a * a

$E \rightarrow E + T$

$\rightarrow T + T$

$\rightarrow F + T$

$\rightarrow a + T$

$\rightarrow a + T * F$

$\rightarrow a + F * F$

a + a * a

$E \rightarrow E + T$

$\rightarrow T + T$

$\rightarrow F + T$

$\rightarrow a + T$

$\rightarrow a + T * F$

$\rightarrow a + F * F$

$\rightarrow a + a * F$

a + a * a

$$
\begin{aligned}
E &\rightarrow E + T \\
&\rightarrow T + T \\
&\rightarrow F + T \\
&\rightarrow a + T \\
&\rightarrow a + T * F \\
&\rightarrow a + F * F \\
&\rightarrow a + a * F \\
&\rightarrow a + a * a
\end{aligned}
$$

Top-down СА методи

Класификация:

a/ Груба сила – недетерминиран метод;

b/ LL(k) – детерминиран метод;

c/ Рекурсивен спуск Recursive descent.

Brute force – не детерминиран метод

Осн. Идея: Няма формален критерий за избор на продукция. По метода на пробата и грешката се прилагат всички възможни продукции.

Пример:

Симетрични низове КСГ: 1. $S \rightarrow a\ S\ b$

2. $S \rightarrow \varepsilon$

Низ за анализ/разбор: aabb#

Brute force – non deterministic method

1. S → a S b
2. S → ε

Input string: aabb#
^

Initial status:

Start non terminal is on top of stack.

Scanning pointer points the first a.

S
▽

Brute force – non deterministic method

1. S → a S b
2. S → ε

Input string: **a**abb#
^

Process:
Top stack char is terminal.
Compare top stack and current input char.
Coincidence: top stack char is popped,
scanning pointer is incremented

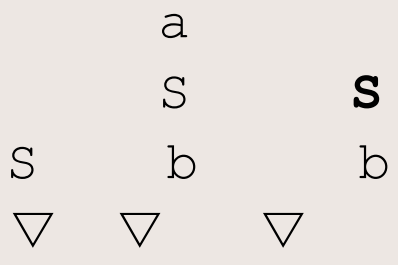
a
S
S b
▽ ▽

Brute force – non deterministic method

1. S → a S b
2. S → ε

Input string: aabb#
^

Process:
Top stack char is non terminal.
Select rule 1 to apply on top stack symbol.

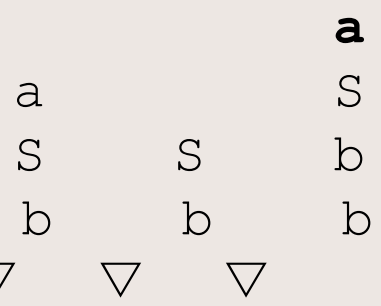


Brute force – non deterministic method

1. S → a S b
2. S → ε

Input string: a**a**bb#
^

Process:
Top stack char is terminal
Compare top stack and current input char.
Coincidence: top stack char is popped,
scanning pointer is incremented



Brute force – non deterministic method

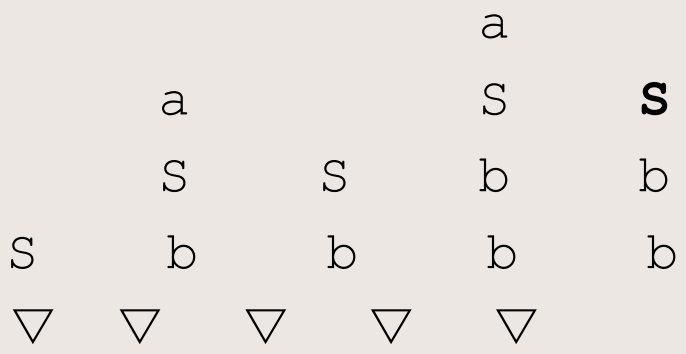
1. S → a S b
2. S → ε

Input string: aabb#
^

Process:

Top stack char is non terminal.

Select rule 1 to apply on top stack symbol.



Brute force – non deterministic method

1. S → a S b
2. S → ε

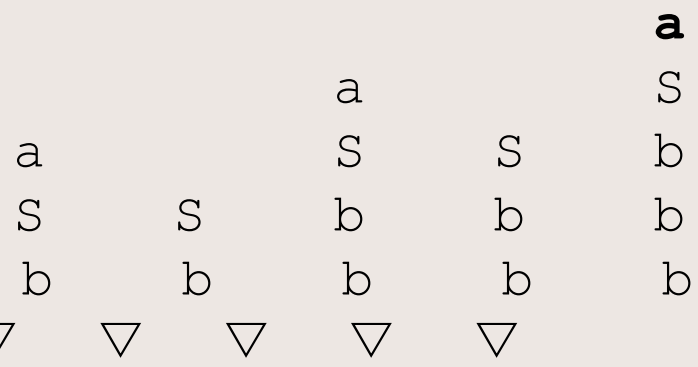
Input string: aa**b**b#
^

Process:

Top stack char is terminal

Compare top stack and current input char.

No Coincidence: try to apply alternate rule on the top stack non terminal, one step back

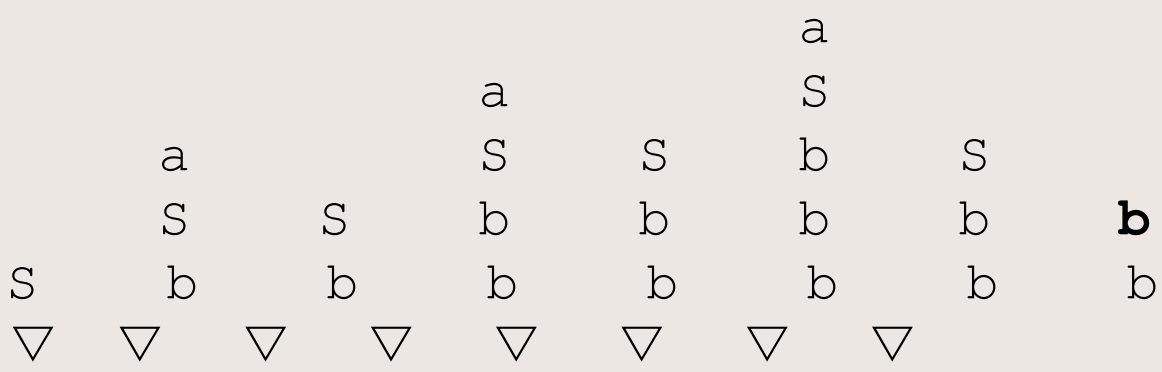


Brute force – non deterministic method

1. S → a S b
2. S → ε

Input string: aa**b**b#
^

Process:
Top stack char is terminal
Compare top stack and current input char.
Coincidence: top stack char is popped,
scanning pointer is incremented

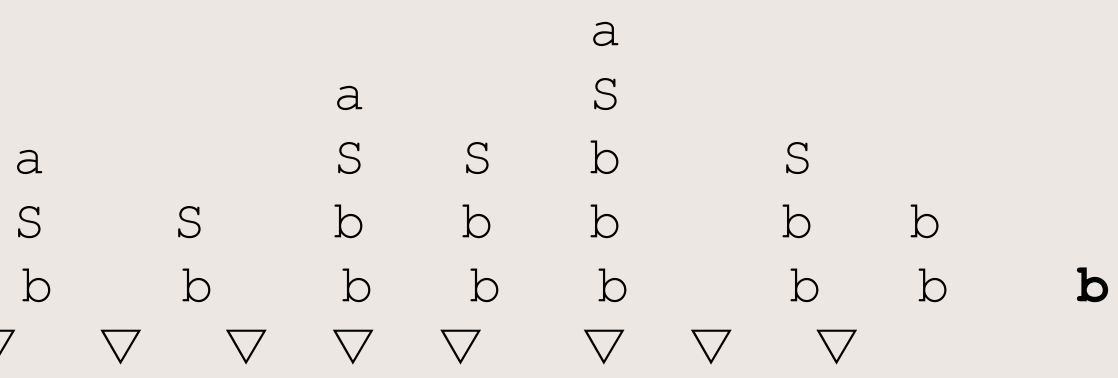


Brute force – non deterministic method

1. S → a S b
2. S → ε

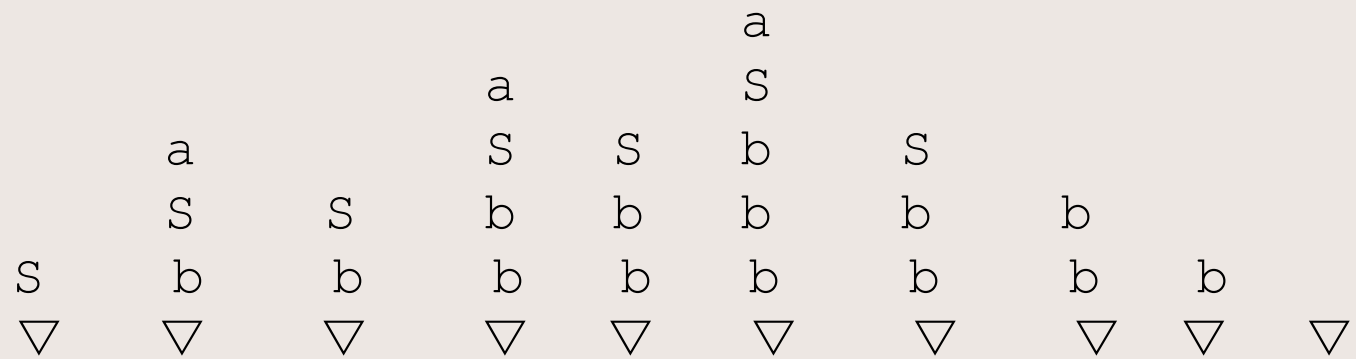
Input string: aab**b**#
^

Process:
Top stack char is terminal
Compare top stack and current input char.
Coincidence: top stack char is popped,
scanning pointer is incremented



Brute force – non deterministic method

```
1. S → a S b          Input string: aabb#
2. S → ε                                ^
Process:
 stack is empty
 Current input char is EOD(#)
```



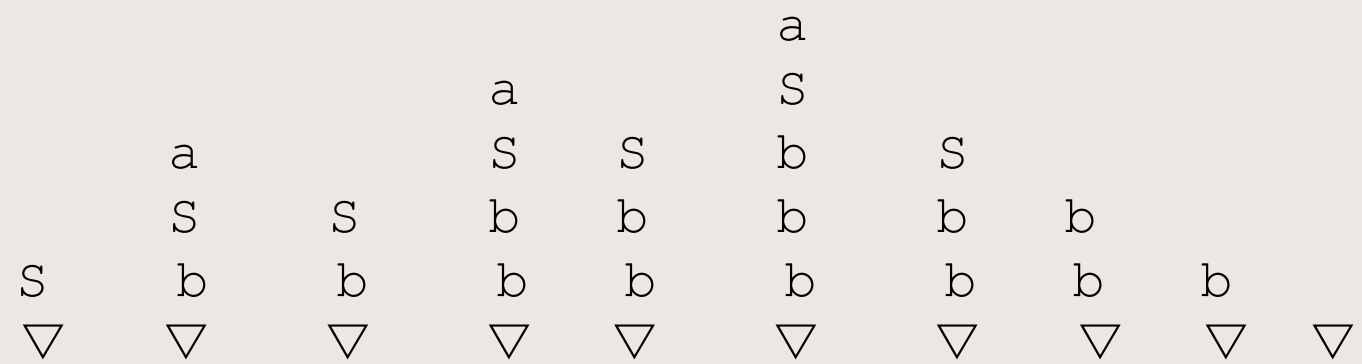
Brute force – non deterministic method

1. S → a S b
2. S → ε

Input string: aabb#
^

Process:

Success: empty stack
exhausted input tape



Top-down СА методи

Класификация:

a/ Груба сила – недетерминиран метод;

b/ LL(k) – детерминиран метод;

c/ Рекурсивен спуск Recursive descent.

LL(k) top-down парсер

Какво значи терминът LL(k)?

- **L**L(k) – вх. Низ се сканира Л > Д
- L**L**(k) – Провежда се ляв каноничен анализ
- LL(**k**) – СА е детерминиран, ако са известни *k* входни символа сочени от показалеца на вх. низ
- LL(1) – СА е детерминиран, ако е известен само 1 (текущият) входен символ

LL(1) top-down парсер

Принципна схема/постановка:

Вх лента Изходна лента

Стек

LL(1) управляваща таблица

Низходяща LL(1) SA стратегия

LL(1) управляваща таблица

LL(1) управл таблица служи да детерминира низходящия СА разбор.

LL(1) управл таблица: размер pow(N) x pow(Σ)

LL(1) управл таблица: съдържание:

- TAB(X,a) = k_i – номер на продукция, която следва да се приложи, ако на върха на стека е не терминал ***X*** и текущият сканиран символ е терминалът ***a***;
- TAB(X,a) = 0 – грешка, от невалидна комбинация топ стек нетерминал и входен терминален символ

LL(1) top-down парсер

Принципна схема/постановка:

Класически LL(1) алгоритъм за СА анализ – блок схема.

LL(1) граматики

5. Класификация на КСГ граматики, подходящи за низходящ СА анализ:

a/ *s-grammars* (simple grammars без ε-правила);

b/ *q-grammars* (s-grammars с ε-правила/rules)

c/ *LL(1) grammars* (без ограничения върху външното представяне на правилата).

s - граматики

а/ *s-grammars* (simple) grammars без ε- правила;

Определение:

- Дясната страна на всяка продукция започва с терминал.
- Десните страни на две или повече правила с еднаква лява част започват с различен терминал.

Управляващата таблица се строи тривиално. Пример:

s – граматики

Пример – управл таблица за s – grammar

1. $S \rightarrow a\ S$
2. $S \rightarrow b\ A$
3. $A \rightarrow d$
4. $A \rightarrow c\ c\ A$

LL(1) управляваща таблица:

	a	b	c	d
S	x	x	x	x
A	x	x	x	x

s – граматики

Пример – управл таблица за s – grammar

1. $S \rightarrow a\ S$
2. $S \rightarrow b\ A$
3. $A \rightarrow d$
4. $A \rightarrow c\ c\ A$

LL(1) управл таблица

	a	b	c	d
S	1	2	0	0
A	0	0	4	3

q - граматики

b/ *q-grammars* (s-grammars with ε- (empty) rules);

За да се построи управл таблица, въвежда се множество FOLLOW(X):

FOLLOW(X) е множество термнални символи, които следват нетерминала X в коя да е сентенциална форма.

q – граматики

q-grammars правилата са два вида:

- s-grammar продукция: $\mathbf{X \rightarrow b\ \alpha'}$

Тогава: $\mathbf{SELECT(X \rightarrow b\ \alpha') = \{ b \}}$

- празно правило: $\mathbf{X \rightarrow \varepsilon}$

Тогава: $\mathbf{SELECT(X \rightarrow \varepsilon) = FOLLOW(X)}$

Пример

q – граматики

Пример – управл таблица за q – grammar

1. $S \rightarrow a\ A\ S$
2. $S \rightarrow b$
3. $A \rightarrow c\ A$
4. $A \rightarrow \varepsilon$

LL(1) управляваща таблица:

	a	b	c
S	x	x	x
A	x	x	x

q – граматики

пример – управл таблица за q – grammar

1. $S \rightarrow a\ A\ S$ SELECT($S \rightarrow a\ A\ S$) = { a }
2. $S \rightarrow b$ SELECT($S \rightarrow b$) = { b }
3. $A \rightarrow c\ A$ SELECT($A \rightarrow c\ A$) = { c }
4. $A \rightarrow \varepsilon$ SELECT($A \rightarrow \varepsilon$) = FOLLOW(A) = {a, b}

LL(1) управл таблица	a	b	c
S	x	x	x
A	x	x	x

q – граматики

пример – управл таблица за q – grammar

1. $S \rightarrow a\ A\ S$ SELECT($S \rightarrow a\ A\ S$) = { a }
2. $S \rightarrow b$ SELECT($S \rightarrow b$) = { b }
3. $A \rightarrow c\ A$ SELECT($A \rightarrow c\ A$) = { c }
4. $A \rightarrow \varepsilon$ SELECT($A \rightarrow \varepsilon$) = FOLLOW(A) = {a, b}

LL(1) управл таблица	a	b	c
S	1	2	0
A	4	4	3

LL(1) граматики

c/ *LL(1) grammars* (без ограничеия в правилата).

За изграждане на таблицата, се въвеждат две нови множества FIRST(α) и SELECT(X→α):

FIRST(α) е множество от терм символи, които стоят в началото на низове, породени от α.

SELECT(X→α) е множество ИЗБОР_НА_ПРОДУКЦИЯ.

LL(1) граматики

Въвеждат се две нови означения:

Анулиращ низ α. α е анулиращ низ, ако той може да се сведе до празния низ ε.

Анулираща продукция X→α. Продукцията X→α е анулираща, ако дясната ѝ страна α е анулиращ низ.

LL(1) граматики

Как се съставя множеството FIRST(α)?

1/ If $\alpha = \varepsilon$ then FIRST(α) = FIRST(ε) = ∅

2/ If $\alpha = b\,\alpha'$ then FIRST(α) = FIRST(b α') = { b }

3/ If $\alpha = X\,\alpha''$ and $X \rightarrow \beta1 \mid \beta2 \mid \ldots \mid \beta n$ then

FIRST(α)= FIRST(X α'') = $\cup$ FIRST(βi), I=1..n

If поне едно anul(βi) then

FIRST(α) = FIRST(α) $\cup$ FIRST(α'')

LL(1) граматики

Как се строи множеството FOLLOW(X)?

1. Начало. Инициализация м-во M = ∅; I=1
2. Сканиране на продукции for I=1,n, step+1
3. Ако X не се среща в дясна страна на правило, няма действие. Преход към 2.
4. Ако X се среща в дясна страна $Y \rightarrow \beta 1 X \beta 2$, then $M = M \cup First(\beta 2)$
5. If (β2=празен низ/ε/ или anul(β2)) и X<>Y, then $M = M \cup Follow(Y)$
6. Край. Follow(X) = M

LL(1) граматики

Как се определят множествата за избор на продукция SELECT($X \to \alpha$)?

a/ случай не анулираща продукция $X \to \alpha$

SELECT($X \to \alpha$) = FIRST(α)

b/ случай анулираща продукция $X \to \alpha$

SELECT($X \to \alpha$) = FIRST(α) $\cup$ FOLLOW(X)

c/ Случай ε- продукция $X \to \varepsilon$

SELECT($X \to \varepsilon$) = FOLLOW(X)

LL(1) граматики

Пример – управл таблица за LL(1) grammar

1. $S \rightarrow A\ a\ C$	FIRST(A a C)	= { e, d, b, a }
2. $S \rightarrow c\ d$	FIRST(c d)	= { c }
3. $A \rightarrow B\ A\ a$	FIRST(B A a)	= { d, e }
4. $A \rightarrow C$	FIRST(C)	= { b }
5. $B \rightarrow e$	FIRST(e)	= { e }
6. $B \rightarrow d$	FIRST(d)	= { d }
7. $C \rightarrow \varepsilon$	FIRST(ε)	= ∅
8. $C \rightarrow b\ S\ d$	FIRST(b S d)	= { b }

LL(1) граматики

Пример – управл таблица за LL(1) grammar

1. $S \rightarrow A\ a\ C$
2. $S \rightarrow c\ d$
3. $A \rightarrow B\ A\ a$
4. $A \rightarrow C$
5. $B \rightarrow e$
6. $B \rightarrow d$
7. $C \rightarrow \varepsilon$
8. $C \rightarrow b\ S\ d$

FOLLOW(A) = { a }

FOLLOW(C) = { a, d }

LL(1) граматики

Пример – управл таблица за LL(1) grammar

LL(1) управляваща таблица:

	a	b	c	d	e
S	x	x	x	x	x
A	x	x	x	x	x
B	x	x	x	x	x
C	x	x	x	x	x

LL(1) граматики

Пример – управл таблица за LL(1) grammar

SELECT(1)= SELECT(S→A a C) = FIRST(A a C) = { e,d,b,a}

SELECT(2)= SELECT(S→c d) = FIRST(c d) = { c }

SELECT(3)= SELECT(A→B A a) = FIRST(B A a) = { d, e }

SELECT(4)= SELECT(A→C)=FIRST(C)∪FOLLOW(A)={b, a}

SELECT(5)= SELECT(B → e) = FIRST(e) = { e }

SELECT(6)= SELECT(B → d) = FIRST(d) = { d }

SELECT(7)= SELECT(C→ε) = FOLLOW(C) = { a, d }

SELECT(8)= SELECT(C→b S d) = FIRST(b S d) = { b }

LL(1) граматики

Пример – управл таблица за LL(1) grammar

LL(1) LL(1) управляваща таблица:

	a	b	c	d	e
S	1	1	2	1	1
A	4	4	0	3	3
B	0	0	0	6	5
C	7	8	0	7	0

Ограничения

Условия за прилагане на низходящ LL(1) СА метод:

a/ Без лява рекурсия в правилата;

b/ Множествата SELECT на две или повече правила с еднакъв ляв нетерминален символ, да не се пресичат, т.е.

$$\mathbf{SELECT(X \rightarrow \alpha) \cap SELECT(X \rightarrow \beta) = \varnothing}$$

Демо програми

програми: llka1.cpp, llkexpr1.cpp.

Благодаря
За
Вниманието