Классическая теория компиляторов

Лекция 3

КОНТЕКСТНО-СВОБОДНЫЕ ГРАММАТИКИ

P: $A \to \alpha$, где $A \in N$, $\alpha \in (N \cup \Sigma)+$

N = {группа_существ, глагол, прилагательное, существительное, предлог, фраза}

Σ = {печатать, стереть, зеленый, первый, последний, символ, строка, страница, в}

S = фраза

P = { Фр $\to$ Г, Гс

Гс $\to$ Прил, С

Гс $\to$ Прил, С, Предлог, Гс

Г $\to$ печатать

Г $\to$ стереть

Прил $\to$ зеленый

Прил $\to$ первый

Прил $\to$ последний

С $\to$ символ

С $\to$ строка

С $\to$ страница

Предлог $\to$ в}

"Печатать зеленый строка",

"Стереть первый символ в последний строка в первый страница" и т.п.

Анализатор на Прологе

```
/* Фраза -> Глагол, Группа_сущ
Группа_сущ -> Прилагат, Существ
Группа_сущ -> Прилагат, Существ, Предлог, Группа_сущ
Глагол -> ...
Прилагат -> ...
Существ -> ...
Предлог -> ...*/
goal
FRAZA =
     ["печатать","первый","символ","в","последний","строка"],
print_list(FRAZA),
s(FRAZA).

clauses
s(A) :- sentence(A), write("\nФРАЗА РАСПОЗНАНА\n").
s(_) :- write("\nОШИБКА: ФРАЗА НЕ РАСПОЗНАНА\n").
% СЛУЖЕБНЫЕ И СЕРВИСНЫЕ ПРЕДИКАТЫ
append([],L,L).
append([H|A],B,[H|C]) :- append(A,B,C).
print_list([]).
print_list([Head|Tail]) :- write(Head,"."),
     print_list(Tail).

% СИНТАКСИС
sentence(S) :-
append(S1,S2,S),
glagol(S1),
GS(S2),
write("\nГлагол:"), writel(S1),
write("\nГс: "), writel(S2).

GS(S) :-
append(S1,S2,S),
prilagat(S1),
noun(S2),
write("\n*** Разбор ГС-1:"),
write("\nПрилагательное: "),writel(S1),
write("\nСуществительное:"),writel(S2).
```

```
GS(S) :-
append(S1,Stmp1,S),
append(S2,Stmp2,Stmp1),
append(S3,S4,Stmp2),
prilagat(S1),
noun(S2),
predlog(S3),

GS(S4),
write("\n*** Разбор ГС-2:"),
write("\nПрилагательное: "),
    print_list(S1),
write("\nСуществительное:"),
    print_list(S2),
write("\nПредлог: "), print_list(S2),
write("\nГс: "), print_list(S2).


% СЛОВАРЬ
noun(["символ"]). noun(["строка"]).
    noun(["страница"]).
glagol(["печатать"]).
    glagol(["стереть"]).
prilagat(["первый"]).
    prilagat(["второй"]).
    prilagat(["последний"]).
predlog(["в"]).
```

Грамматика арифметического выражения

$G_0=(\{E,T,F\},\{a,+,*,(,)\},P,E)$

P = {

$E \rightarrow E+T$

$E \rightarrow T$

$T \rightarrow T*F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow a$ }

Анализатор на Прологе

```
1. E(L)  :- T(L).
2. E(L)  :- a3(L1,["+"],L3,L),
3.             E(L1),
4.             T(L3).
5. T(L)  :- F(L).
6. T(L)  :- a3(L1,["*"],L3,L),
7.             T(L1), F(L3).
8. F(L)  :- L=["a"].
9. F(L)  :- a3(["("],L2,[")"],L),
10.            E(L2).
```

ОК-ГРАММАТИКИ

Грамматики определенных клауз.
Наличие *контекстуальных аргументов*.

Гс → Мест(k), Прил(k), Сущ(k), ГС

k - контекстуальный аргумент, отвечающий за согласование родов.

Мест(муж) → этот
Мест(жен) → эта
Прил(муж) → второй
Прил(жен) → вторая
Сущ(жен) → строка
Сущ(муж) → пароль
и т.д.

На Прологе:
mest(“муж”, ”этот”).
pril ("жен", "вторая").
и т.п.

СИНТАКСИЧЕСКИ УПРАВЛЯЕМЫЙ ПЕРЕВОД

$S \to E$

$E \to T,\ E \to E+T,\ T \to F,\ T \to T*F,\ F \to a,\ F \to (E)$

Левый вывод цепочки:

Пусть дана фраза: a+a*a.

S→E→E+T→E+T*F→T+T*F→F+F*F→a+a*a

⇒фраза принадлежит нашему языку.

Но: 1) долго, 2) неопределенно, 3) надо сделать что-то полезное (сформировать ПФ)

Идея СУ-перевода

Применение каждого правила грамматики будет вызывать выполнение той или иной семантической процедуры.

Правило	Элемент перевода
$E \to E+T$	E = ET+
$E \to T$	E = T
$T \to T*F$	T = TF*
$T \to F$	T = F
$F \to (E)$	F = E
$F \to a$	F = a

$(E,E) \Rightarrow (E+T, ET+)$
$\Rightarrow (T+T, TF+)$
$\Rightarrow (F+T, FT+)$
$\Rightarrow (a+T, aT+)$
$\Rightarrow (a+T*F, aTF*+)$
$\Rightarrow (a+F*F, aFF*+)$
$\Rightarrow (a+a*F, aaF*+)$
$\Rightarrow (a+a*a, aaa*+).$

Перевод инфиксной формы записи в ПФ

Z ::= E
E ::= T | E+T | E-T | -T
T ::= F | T*F | T/F
F ::= a | (E)

№	Правило	Семантическая программа	Условие применимости
1	Z ::= E	нет	(*)
2	E ::= T	нет	(**)
3	E ::= E+T	Push('+')	(**)
4	E ::= E-T	Push('-')	(**)
5	E ::= -T	Push('@')	
6	T ::= F	Нет	
7	T ::= T*F	Push('*')	
8	T ::= T/F	Push('/')	
9	F ::= a	Push(a)	
10	F ::= (E)	Нет	

Алгоритм СУ-перевода

"a*(b+c)#"

Признак нормального окончания работы алгоритма:

- когда в стеке остался единственный символ Z, а текущим символом является ‘#’, то процедура завершена успешно.
- В противном случае фраза построена неверно.

Недостаток: плохая диагностика

Стек S	R	ω_k...	Номер правила	Польская цепочка
#	a	*(b+c)#		
#a	*	(b+c)#	9	a
#F	*	(b+c)#	6	a
#T	*	(b+c)#		a
#T*	(	b+c)#		a
#T*(	b	+c)#		a
#T*(b	+	c)#	9	ab
#T*(F	+	c)#	6	ab
#T*(T	+	c)#	2	ab
#T*(E	+	c)#		ab
#T*(E+	c	)#		ab
#T*(E+c	)	#	9	abc
#T*(E+F	)	#	6	abc
#T*(E+T	)	#	3	abc+
#T*(E	)	#		abc+
#T*(E)	#		10	abc+
#T*F	#		7	abc+*
#T	#		2	abc+*
#E	#		1	abc+*
#Z	#		STOP	abc+*

Замечание об унарных операторах

Считается, что унарные операторы имеют наивысший приоритет. Исходя из этого, соответствующий правила грамматики должны записываться в конце списка. Тогда получаем:

...
T→-F
T→!F
...

Т.е. действие унарного оператора (‘-’ или ‘!’) будет направлено на F (F→(E), F→a). Если же поместить унарный оператор в правило

E→-T (или E→!T),

то оператор будет действовать уже на терм.

Это означает, что в выражении

"!a*b"

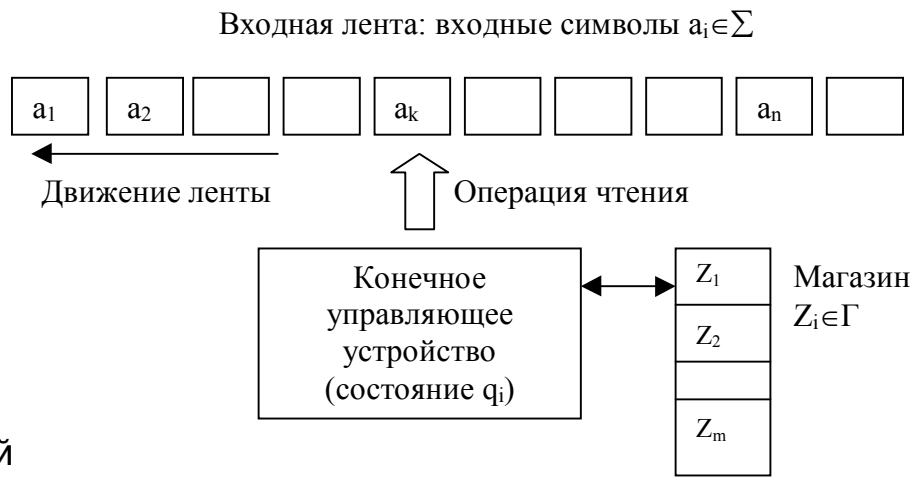
сначала будет вычислено произведение a*b, а затем только будет взято его отрицание.

АВТОМАТЫ С МАГАЗИННОЙ ПАМЯТЬЮ

- Для КСГ создать конечный распознающий автомат уже невозможно.
- Более сложные по своей структуре автоматы – *автоматы с магазинной памятью*, которые применяются для распознавания фраз КСГ.

МП = ($\sum$, Q, Г, q_0, z_0, T, P), где

- $\sum$ – конечный входной алфавит (входной словарь);
- Q – конечное множество состояний;
- Г – конечный алфавит магазинных символов;
- q_0 – начальное состояние управляющего устройства ($q_0 \in Q$);
- z_0 – символ, находящийся в магазине в начальный момент времени (начальный символ), $z_0 \in Г$;
- T – множество терминальных (заключительных) состояний, $T \subset Q$;
- P: $Q \times (\sum \cup \{e\}) \times Г \rightarrow Q \times Г^*$



Понятия, связанные с АМП

- *Конфигурацией* МП-автомата называется тройка $(q,\omega,\alpha)\in Q\times\Sigma^*\times\Gamma^*$, где
  - q – текущее состояние устройства;
  - ω – неиспользованная часть входной цепочки; первый символ цепочки ω находится под входной головкой; если ω = e, то считается, что вся входная лента прочитана;
  - α – содержимое магазина; самый левый символ цепочки a считается верхним символом магазина; если a=e, то магазин считается пустым.
- *Такт работы МП-автомата:*

  $(q,a\omega,Z\alpha) \rightarrow (q', \omega,\gamma\alpha)$

  Если a=e, то этот такт называется *e-тактом*. В e-такте входной символ не принимается во внимание и входная головка не сдвигается. Если γ=e, то верхний символ удаляется из магазина (магазинный список *сокращается*).
- *Начальная конфигурация* МП-автомата – это тройка

  (q_0, ω, Z_0), $\omega\in\Sigma^*$
- Говорят, что цепочка ω *допускается* МП-автоматом, если

  $(q_0, \omega, Z_0) \rightarrow^*(q, e, \alpha)$ для некоторых $q\in T$ и $\alpha\in\Gamma^*$
- *Расширенный МП-автомат*

  $МП_r = (\Sigma, Q, \Gamma, q_0, Z_0, T, P_r)$, где

  P_r – отображение множества $Q\times(\Sigma\cup\{e\})\times\Gamma^* \rightarrow Q\times\Gamma^*$.
- Расширенный МП-автомат обладает способностью продолжать работу и тогда, когда **магазин пуст**.
- ***Теорема***. Пусть $G = (N,\Sigma,P,S)$ – КС-грамматика. По грамматике G можно построить такой МП-автомат R, что $L_e(R)=L(G)$.

Пример МП-автомата

Дано: G_0=({E,T,F},{a,+,*,(,),#},P,E)

P = { E $\rightarrow$ E+T|T
T $\rightarrow$ T*F|F
F $\rightarrow$ (E)|a}

Здесь '#' – символ конца входной последовательности.

R = (Σ, Q, Г, q_0, Z_0, T, P), где

Q = {q, r},

Г={E, T, F, \$} $\cup$ Σ

q_0 = q,

Z_0 = \$,

T = {r},

P:

(1) (q,e,E+T) $\rightarrow$ (q,E) (*)
(2) (q,e,T) $\rightarrow$ (q,E) (*)
(3) (q,e,T*F) $\rightarrow$ (q,T)
(4) (q,e,F) $\rightarrow$ (q,T)
(5) (q,e,(E)) $\rightarrow$ (q,F)
(6) (q,e,a) $\rightarrow$ (q,F)
(7) (q,#,\$E) $\rightarrow$ (r,e)
(8) (q,b,e) $\rightarrow$ (q,b) для всех b$\in${a,+,*,(,)};

Замечание 1. Правила (*), применимы, если следующим символом входной цепочки является '+', или '-', ')' или входная цепочка пуста.

Замечание 2. Приоритет выполнения правил определяется содержимым стека.

Пример разбора

Вход: "a+a*a".

(q,a+a*a#,$) → (q, +a*a#, $a)
→ (q, +a*a#, $F)
→ (q, +a*a#, $T)
→ (q, +a*a#, $E)
→ (q, a*a#, $E+)
→ (q, *a#, $E+a)
→ (q, *a#, $E+F)
→ (q, *a#, $E+T)
→ (q, a#, $E+T*)
→ (q, e, $E+T*a)
→ (q, e, $E+T*F)
→ (q, e, $E+T)
→ (q, #, $E)
→ (r, #, e)

- Схема СУ-перевода и разбор КС-грамматики с помощью АМП *эквивалентны*.
- Можно дополнить АМП набором семантических процедур. Тогда можно формировать ПФ.
- АМП можно считать просто более формальным изложением алгоритма СУ-перевода. Например:

1) алгоритм СУ-перевода: если ни одно из правил не применимо к содержимому стека, то следует поместить в стек очередной символ входной последовательности. В МП-автомате вместо этого используется правило (8)

((q,b,e) → (q,b) для всех b∈{a,+,*,(,)};)

2) Условие завершения СУ-перевода формулируется правилом (7) ((q,#,$E) → (r,e)).