Лекция 12

19 ноября

7.ВОСХОДЯЩИЙ ГРАММАТИЧЕСКИЙ РАЗБОР

7.1.Распознавание методом “перенос-опознание”

- Восходящий разбор, как уже отмечалось ранее, базируется на понятии основы. Рассмотрим механизм такого разбора на примере.
- Пусть дана следующая грамматика:
-
- ***1. <S> → (<A> <S>)***
- ***2. <S> → (b)***
- ***3. <A> → (<S> a <A>)***
- ***4. <A> → (a)***
-
- Начальный символ грамматики ***<S>***.

Распознавание методом “перенос-опознание” (продолжение)

- Теперь рассмотрим цепочку
- ***((a) (b))***
- Правый вывод для этой цепочки имеет следующий вид:
- ***<S> Þ (<A> <S>) Þ (<A> (b)) Þ ((a) (b))***
- Восходящий разбор представляет собой обращение правого вывода: мы находим основу (т.е. последнее применение правила), и заменяем её левой частью соответствующего правила (основа подчёркивается):
- ***((a) (b)) - - > (<A> (b)) - - > (<A> <S>) - - > <S>***

МП-автомат

- Для преобразования цепочки будем применять МП-автомат.
- Наш МП-автомат будет использовать пять операций: *Перенос, Свёртка(1), Свёртка(2), Свёртка(3), Свёртка(4)*.
- Операция *Перенос* переносит текущий входной символ на верх магазина и осуществляет сдвиг по входной цепочке.
- Операция *Свёртка(p)* выполняется в тех случаях, когда верхние символы магазина совпадают с правой частью правила *p.*
- Операция *Свёртка(p)* выталкивает из магазина все символы правой части правила *p* и вталкивает в него левую часть правила (т.е. заменяет основу цепочки левой частью соответствующего правила).
- Рассмотрим работу автомата при распознавании приведённой ранее цепочки:

Работа МП-автомата при распознавании цепочки

Содержимое магазина	Текущая входная цепочка	Операция
∇	((a)(b))⊥	*Перенос*
∇(	(a)(b))⊥	*Перенос*
∇((	a)(b))⊥	*Перенос*
∇((a	)(b))⊥	*Перенос*
∇((a)	(b))⊥	*Свёртка(4)*
∇(<A>	b))⊥	*Перенос*
∇(<A>(	))⊥	*Перенос*
∇(<A>(b	)⊥	*Перенос*
∇(<A>(b)	)⊥	*Свёртка(2)*
∇(<A><S>	)⊥	*Перенос*
∇(<A><S>)	⊥	*Свёртка(1)*
∇<S>	⊥	*Допустить*

Таблица переходов для МП-автомата

Магазинные символы	(	)	a	b	⊥
<S>	*Перенос*	*Перенос*	*Перенос*	*Перенос*	*Опознать 2*
<A>	*Перенос*	*Перенос*	*Перенос*	*Перенос*	*Опознать 2*
(	*Перенос*	*Перенос*	*Перенос*	*Перенос*	*Опознать 2*
)	*Опознать 1*	*Опознать 1*	*Опознать 1*	*Опознать 1*	*Опознать 1*
a	*Перенос*	*Перенос*	*Перенос*	*Перенос*	*Опознать 2*
b	*Перенос*	*Перенос*	*Перенос*	*Перенос*	*Опознать 2*
∇	*Перенос*	*Перенос*	*Перенос*	*Перенос*	*Опознать 2*
Начальное содержимое магазина: ∇					

Элементы таблицы

- *Опознать 1*: Если на верху магазина ***(<A> <S>)***, то *Свёртка(1)*;
- иначе если на верху магазина ***(b)***, то *Свёртка(2)*; иначе если на
- верху магазина ***(<S> α <A>)***, то *Свёртка(3)*; иначе если на верху
- магазина ***(α)***, то *Свёртка(4)*; иначе *Отвергнуть*.
- *Опознать 2*: Если на верху магазина ∇ ***<S>***, то *Допустить*,
- иначе *Отвергнуть*.
- *Перенос*: *Втолк(текущий входной символ), Сдвиг*.
- *Свёртка(1)*: *Вытолк, Вытолк, Вытолк, Вытолк, Вытолк,*
- *Втолк(<S>)*.
- *Свёртка(2)*: *Вытолк, Вытолк, Вытолк, Втолк(<S>)*.
- *Свёртка(3)*: *Вытолк, Вытолк, Вытолк, Вытолк, Вытолк, Вытолк,*
- *Втолк(<A>)*.
- *Свёртка(4)*: *Вытолк, Вытолк, Втолк(<A>)*.

Комментарий

- Построить автомат для данной грамматики было несложно, поскольку можно заметить, что каждая основа заканчивается правой скобкой, причём правая скобка может быть только концом основы.
- Автоматы такого рода мы будем называть автоматами типа “перенос-опознание”.
- Далее мы будем предполагать, что грамматики не содержат $\boldsymbol{\varepsilon}$-правил.

7.2.Управление автоматом типа “перенос-опознание”

- Разделим задачу построения автомата на две части.
- Первая состоит в том, чтобы решить, какие элементы управляющей таблицы автомата должны содержать операции *Перенос*, какие – процедуры опознания и какие – операции *Отвергнуть*.
- Вторая часть задачи состоит в отыскании подходящих процедур опознания.
- В данном параграфе мы рассмотрим первую часть задачи построения МП-автомата.

Управление автоматом типа “перенос-опознание” (продолжение)

- Предположим, что у нас есть автомат типа “перенос-опознание” для данной грамматики с начальным символом ***<S>***.
- На каждом шаге обработки этим автоматом некоторой входной цепочки справедливо в точности одно из следующих утверждений:
- 1. Входная цепочка допустима, и верхняя часть магазина является основой цепочки, представляемой магазином и ещё не обработанными входными символами.
- 2. Входная цепочка допустима, содержимое магазина ∇***<S>***, и текущий входной символ – концевой маркер.
- 3. Входная цепочка допустима, текущий входной символ – не концевой маркер, и верхняя часть магазина не является основой представленной в магазине цепочки.
- 4. Входная цепочка не допустима.

Управление автоматом типа “перенос-опознание” (продолжение)

- Каждый раз, когда автомат достигает конфигурации, описываемой одним из первых двух утверждений, он должен выбрать процедуру опознания. Выбор осуществляется с помощью управляющей таблицы, исходя из верхнего символа магазина и текущего входного символа. Элемент управляющей таблицы для данной комбинации входного и магазинного символов должен содержать процедуру опознания.
- Каждый раз при достижении автоматом конфигурации, описываемой утверждением 3, он должен выбрать операцию *Перенос*. Следовательно, элементы таблицы для комбинаций входных и магазинных символов, встречающихся в конфигурациях, описываемых утверждением 3, должны содержать *Перенос*.
- Элементы таблицы, соответствующие утверждению для конфигурации 4, могут содержать операцию *Отвергнуть*, но могут содержать и операцию *Перенос* или процедуру опознания (в этом случае цепочка будет отвергаться позже).

Управление автоматом типа “перенос-опознание” (продолжение)

- Для того, чтобы решить, какие элементы управляющей таблицы должны содержать перенос, а какие – процедуры опознания, необходимо проанализировать грамматику выяснить, какие элементы таблицы могут встретиться в конфигурациях, описываемых утверждениями 1 или 2, а какие – в конфигурациях, соответствующих утверждению 3.
- Для каких-то грамматик такой анализ может показать, что некоторые элементы таблицы могут встречаться как в ситуациях, описываемых утверждениями 1 или 2, так и в ситуациях, описываемых утверждением 3.
- То есть табличный элемент должен содержать и перенос, и процедуру опознания. Это означает, что управляющий механизм типа “перенос-опознание” не подходит для данной грамматики, и необходимо обратиться или к другой грамматике, описывающий тот же язык, или к другому типу автомата.
- Такую ситуацию будем называть конфликтом переноса-опознания.

Управление автоматом типа “перенос-опознание” (продолжение)

- Как средство анализа грамматики мы будем использовать множества *Перв* и *След*.
- Для данной контекстно-свободной грамматики с начальным символом ***<S>*** и для символа ***X*** определим:
- Множество *След(X)* – множество входных символов, которые могут непосредственно следовать за ***X*** в некоторой промежуточной цепочке, выводимой из ***<S>*** $\perp$ **.**
- Множество *Перв(**X**)* – множество тех символов грамматики, которые встречаются в начале промежуточных цепочек, выводимых из ***X***.
- Эти определения расширяют ранее введённые понятия *Перв* и *След*.
- Рассмотрим пример. Пусть дана следующая грамматика:

Пример

1. <S> → b <A> <S> <B>
2. <S> → b <A>
3. <A> → d <S> c a
4. <A> → e
5. <B> → c <A> a
6. <B> → c

Пример (продолжение)

- Множество *Перв* для символов данной грамматики вычисляется тривиально и имеет следующий вид:
-
- *Перв(<S>) = { b, <S> }*
- *Перв(<A>) = { d, e, <A> }*
- *Перв(<B>) = { c, <B> }*
- *Перв(a) = { a }*
- *Перв(b) = { b }*
- *Перв(c) = { c }*
- *Перв(d) = { d }*
- *Перв(e) = { e }*

Пример (продолжение)

- Вычислим множества *След*:
-
- *След(<S>) = { c, ⊥}*
- *След(<A>) = { a, b, c, ⊥}*
- *След(<B>) = { c, ⊥}*
- *След(a) = { a, b, c, ⊥}*
- *След(b) = { d, e}*
- *След(c) = { a, c, d, e, ⊥}*
- *След(d) = { b }*
- *След(e) = { a, b, c, ⊥}*

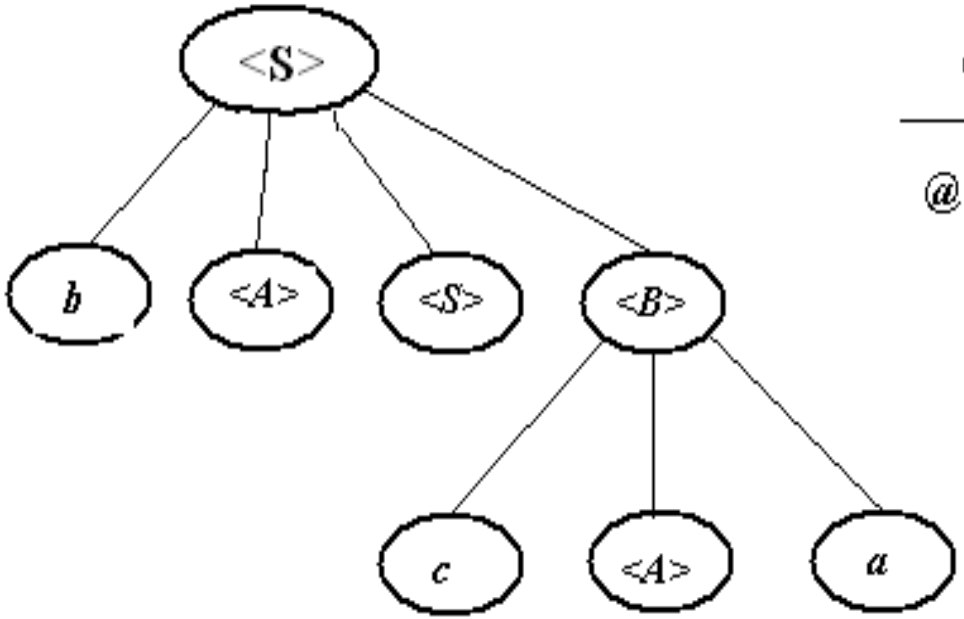
О заполнении управляющей таблицы

- Обсудим теперь управляющую таблицу типа “перенос-опознание” для нашей грамматики.
- Многие элементы управляющей таблицы можно заполнить на основе простого наблюдения, какие из символов в правых частях правил встречаются только в самой правой позиции, а какие – только в других позициях.
- Например, символы ***b, d*** и ***<S>*** ни в одной правой части правил не встречаются в качестве самых правых символов, и, поэтому ***b, d*** , ***<S>***, а также ∇ на верху магазина не могут быть крайними правыми символами основы.
- Следовательно, если один из этих символов находится на верху магазина, автомат должен выполнить перенос любого входного символа (кроме концевого маркера), и, значит можно соответствующим образом заполнить элементы управляющей таблицы.

О заполнении управляющей таблицы (продолжение)

- Символы ***a, e*** и ***<B>*** встречаются только в крайних правых позициях правил, следовательно, появление ***a, e*** или ***<B>*** на верху магазина может соответствовать лишь самому правому символу основы, и, значит, автомат должен вызвать процедуру опознания. Таким образом, вызовы этих процедур нужно включить в управляющую таблицу.
- Подобные простые рассуждения относительно заполнения управляющей таблицы неприменимы лишь в тех случаях, когда на верху магазина находятся ***<A>*** или ***c***. Нетерминал ***<A>*** встречается один раз в крайней правой позиции (правило 2) и дважды в других позициях (правила 1 и 5). Аналогично и для символа ***c***.
- Рассмотрим несколько конфигураций, в которых символ ***<A>*** находится на верху магазина.

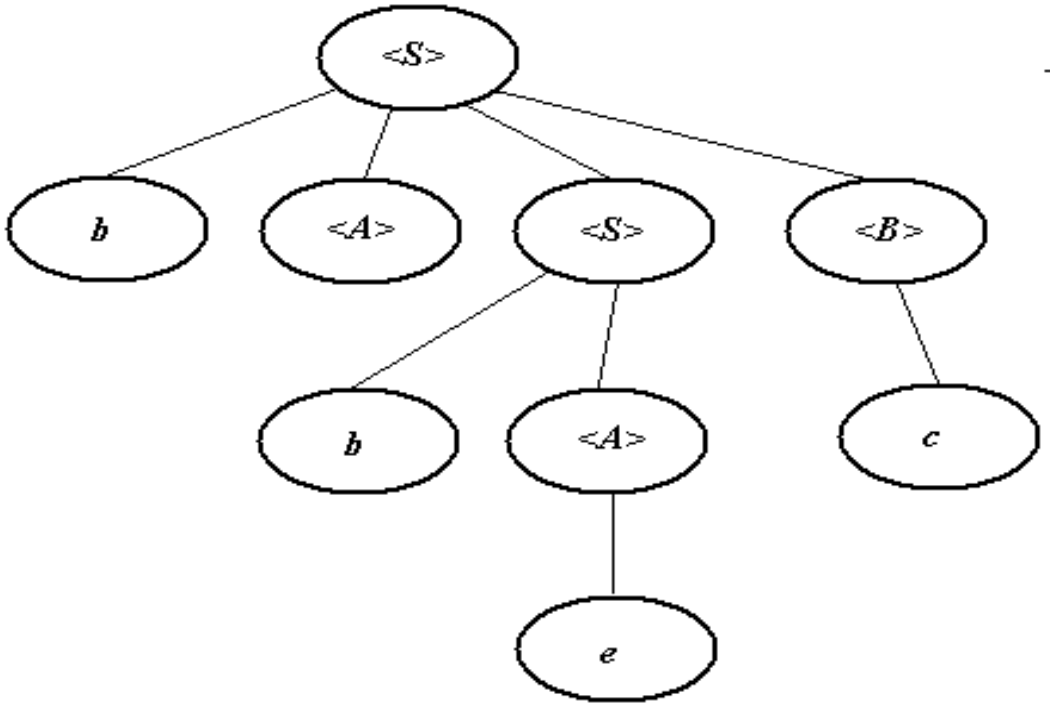
Конфигурация 1



Сод. магазина	Тек. вх. цеп.
@ *b* <A> <S> *c* <A>	*a* #

Конфигурация 1

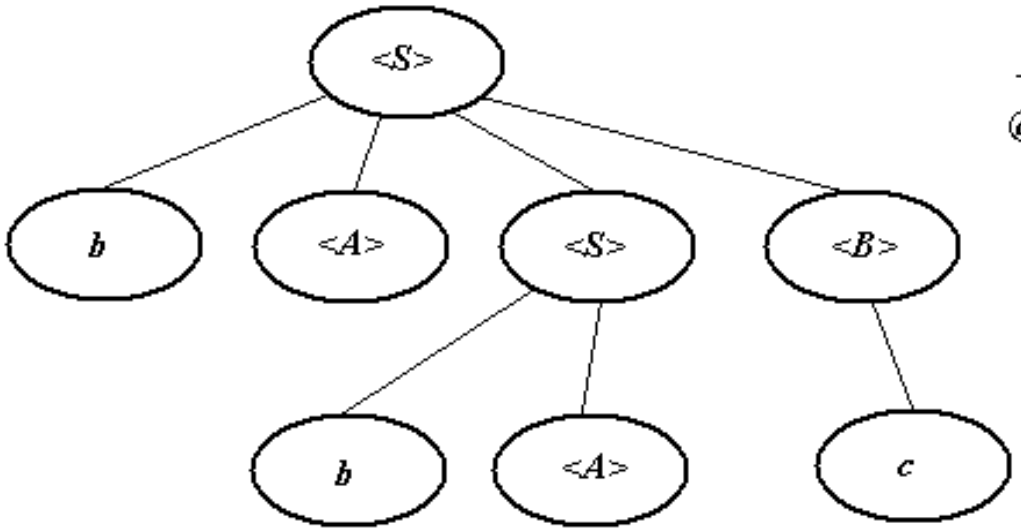
Конфигурация 2



Сод. магазина	Тек. вх. цеп.
@ b <A>	b e c #

Конфигурация 2

Конфигурация 3



Сод. магазина	Тек. вх. цеп.
@ b <A> b <A>	c #

Конфигурация 3

Обсуждение конфигураций

- В ситуации, соответствующей конфигурации 3, верно утверждение 1 (т.е. верхний магазинный символ ***<A>*** является концом основы); а в ситуациях, соответствующих конфигурациям 1 и 2, справедливо утверждение 3 (т.е. верхний магазинный символ ***<A>*** является концом основы).
- Таким образом, конфигурация 3 показывает, что элемент строки управляющей таблицы, соответствующий символу ***<A>***, для входного символа ***c*** должен содержать процедуру опознания; а из конфигураций 1 и 2 следует, что элементы этой строки для входных символов ***a*** и ***b*** должны содержать *Перенос*.
- Возникает вопрос, возможны ли другие ситуации, противоречащие такому выбору элементов управляющей таблицы для входных символов ***a, b*** и ***c***. Кроме того, надо знать какой выбор можно сделать для входных символов ***d*** и ***e***.

Выбор операции

- Итак, пусть на верху магазина находится нетерминал ***<A>***. Выясним, в каких случаях нужно применять процедуру опознания, в каких – перенос, а когда следует отвергать цепочку.
- Процедура опознания применяется, когда на верху магазина находится основа цепочки, представляемой магазином и текущей входной цепочкой, и нетерминал ***<A>*** находится на конце основы, т.е. правой части правила.
- Мы имеем такое правило – это правило 2 с нетерминалом ***<S>*** в левой части. Текущий входной символ следует за этой правой частью, т.е. он следует за нетерминалом ***<S>***, если мы редуцируем основу.
- Таким образом, процедура опознания должна применяться (при верхнем магазинном символе ***<A>***), когда текущий входной символ принадлежит множеству *След(<S>)*, т.е. для ***c*** и ***#.***

Выбор операции (продолжение)

- Перенос осуществляется, когда верхний магазинный символ ***<A>*** не является концом основы, т.е. не является концом правой части некоторого правила, а находится внутри правой части правила.
- В этом случае текущий входной символ принадлежит множеству *Перв(<X>)*, где *X* – символ, расположенный сразу после ***<A>*** в этой правой части.
- В нашем случае символ ***<A>*** встречается внутри правых частей правил 1 и 5.
- Таким образом, операция *Перенос* должна выполняться, когда текущий входной символ принадлежит объединению множеств *Перв(<S>)* и *Перв(a)*, т.е. для ***a*** и ***b***.
- Видим, что требования переноса и опознания не противоречат друг другу.

Выбор операции (продолжение)

- Можно также заключить, что при обработке допустимой цепочки элементы строки ***<A>*** для входных символов ***d*** и ***e*** не будут использоваться, и каждый из этих элементов может содержать либо операцию *Отвергнуть*, либо *Перенос*, либо операцию опознания.
- Приведённые рассуждения иллюстрируют два принципа, которые называются “*принцип свёртывания*” и “*принцип переноса*”.

Принцип свёртывания

- *“Принцип свёртывания”* формулируется следующим образом:
- Для данной грамматики управляющую таблицу автомата типа “перенос-опознание” нужно строить так, чтобы она обеспечивала выбор процедуры опознания для магазинного символа ***A*** и текущего входного символа ***t*** в тех случаях, когда выполняется одно из следующих условий:
-
- 1. Существует нетерминал ***L*** и цепочка ***x***, такие что
- $L \rightarrow xA$
- есть правило грамматики, и ***t*** принадлежит множеству *След(L)*.
-
- 2***. A*** – начальный символ грамматики, ***t*** – концевой маркер.

Принцип свёртывания (продолжение)

- Часто бывает удобно рассматривать принцип свёртывания в терминах отношения *Свёртывается-По*, определяемого следующим образом:
- Если ***A*** – символ грамматики, а ***t*** – входной символ, то
- ***A*** *Свёртывается-По* ***t***
- тогда и только тогда, когда выполняется одно из условий принципа свёртывания.
- Таким образом, управляющий механизм типа “перенос-опознание” нужно строить так, чтобы он содержал процедуру опознания для всех комбинаций магазинного символа ***A*** и входного символа ***t***, таких что
- ***A*** *Свёртывается-По* ***t***

Принцип переноса

- “*Принцип переноса*” формулируется следующим образом:
- Для данной грамматики управляющую таблицу автомата типа “перенос-опознание” нужно строить так, чтобы она содержала операцию *Перенос* для магазинного символа ***A*** и текущего входного символа ***t*** в тех случаях, когда выполняется одно из следующих условий:
- 1. Имеется символ грамматики ***B***, такой что в правую часть некоторого правила входит цепочка ***AB***, и ***t*** принадлежит множеству *Перв(B)*.
- 2. ***A*** – маркер дна магазина, а ***t*** принадлежит множеству *Перв(S)*, где *S* – начальный символ грамматики.

Принцип переноса (продолжение)

- Часто бывает удобно рассматривать принцип переноса в терминах отношения *Под*, определяемого следующим образом:
- Если ***A*** – магазинный символ, а ***t*** – входной символ, то
- ***A*** *Под* ***t***
- тогда и только тогда, когда выполняется одно из условий принципа переноса.
- Таким образом, управляющую таблицу автомата типа “перенос-опознание” нужно строить так, чтобы она содержала операцию *Перенос* для всех комбинаций магазинного символа ***A*** и входного символа ***t***, таких что
- ***A*** *Под* ***t***