Лекція 9. Види синтаксичного розбору

Необходимость использования автоматов с магазинной памятью

Синтаксический разбор осуществляется с применением более сложных грамматик, обеспечивающих иерархическое определение одних правил через другие. Поэтому, для построения распознавателей, мощности конечных автоматов, используемых при лексическом анализе, уже не хватает. Необходим более мощный и универсальный автомат, поддерживающий построение дерева разбора и выстраивающий его как сверху вниз, так и снизу вверх. Из предыдущих тем известно, что конечный автомат можно расширить дополнительной рабочей памятью, чтобы обеспечить распознавание цепочек, порождаемых практически любыми грамматиками. Организация и поведение такого автомата определяется классом грамматик. Как определено в классификации Хомского, контекстно-свободным грамматикам можно поставить в соответствие автомат с магазинной памятью (АМП).

То, что даже простые цепочки проблематично распознать с использованием конечного автомата, можно проиллюстрировать решением следующей элементарной задачи: необходимо построить распознаватель правильной вложенности круглых скобок. Такая вложенность скобок часто встречается в различных языках программирования при построении арифметических выражений. Для решения данной задачи необходимо:

1. определять равенство скобок, то есть количество открывающихся и закрывающихся скобок должно быть равным;
2. следить за правильностью их вложения, то есть, чтобы любой закрывающейся скобке предшествовала соответствующая открывающаяся.

Невозможность использования конечного автомата подтверждается и тем, что грамматику, порождающую рассматриваемые цепочки, нельзя свести к праволинейной (тому виду, который, по классификации Хомского, эквивалентен конечному автомату). Ее также нельзя представить только итеративными диаграммами Вирта, непосредственно соответствующими конечному автомату. Это определяется наличием в грамматике правил с рекурсией в середине. А такая рекурсия не может быть сведена к итерации. Грамматика $\mathbf{G_{7.1}}$ Может быть определена следующим образом:

$$\mathbf{G_{7.1} = (\{A, S\}, \{ (,) \}, P, S)} \ ,$$

Где P определяется как:

1. S →(A) A
2. A →S
3. A →ε

Одним из путей решения задачи является добавление счетчика скобок, который, при просмотре цепочки увеличивается на 1, если встретится открывающаяся скобка. Закрывающаяся скобка уменьшает значения счетчика на единицу. Начальное состояние

счетчика (перед просмотром цепочки) равно 0. По завершении просмотра цепочки значение счетчика должно быть равным 0. При этом, в ходе просмотра цепочки счетчик не может принимать отрицательные значения. Подход обеспечивает решение поставленной задачи. Однако, наличие счетчика уже определяет автомат с дополнительно рабочей памятью, которому также необходимо наличие арифметического устройства. Поэтому, использование стека вряд ли будет сложнее. А экономить ячейки памяти мы давно уже разучились. Кроме того, в реальной ситуации могут быть более сложные зависимости. Например, может быть несколько видов скобок со своими правилами вложенности и их взаимным расположением. Значит, необходим универсальный подход, обеспечивающий преодоление различных ситуаций. Таким универсальным подходом и является использование автомата с магазинной памятью.

Организация автомата с магазинной памятью

АМП в качестве рабочей памяти использует стек, называемый также магазином. Данная память поддерживает только ограниченные операции доступа, в то же время достаточные для решения сложных задач, включая и задачи распознавания цепочек. Автомат с магазинной памятью определяется следующими пятью объектами:

1. Конечным **множеством входных символов**, включающим концевой маркер (⊣).
2. Конечным **множеством магазинных символов**, включающим маркер дна (▽).
3. Конечным множеством состояний, включающим начальное состояние.
4. **Устройством управления** (УУ), которое каждой комбинации входного символа, магазинного символа и состояния ставит в соответствие выход или переход. Переход, в отличие от выхода, заключается в выполнении операций над магазином, состоянием и входом. Операции, запрашивающие входной символ после концевого маркера или выталкивания из магазина после маркера дна, а также операция вталкивания маркера дна, исключаются.

5. **Начальным содержимым магазина**, содержащим маркер дна и, возможно пустую, цепочку магазинных символов.

Автомат с магазинной памятью называется **распознавателем**, если у него 2 выхода: **"Допустить"** и **"Отвергнуть"**.

Операции автомата

Динамическое поведение АМП описывается через его операциями над входной цепочкой и стеком, а также переходами из одного состояния в другое. К операциям над стеком относятся:

1. **"Вытолкнуть"** - выталкивает из стека верхний символ (будем также использовать сокращенное обозначение **"↑"**).
2. **"Втолкнуть A"** - вталкивает в стек магазинный символ A (будем также использовать сокращенное обозначение **"↓A"**).
3. **"Заменить XYZ"** - используется для сокращения записи, когда необходимо вытолкнуть верхний символ и вместо него втолкнуть несколько других (в данном случае X, Y, Z). Эквивалентна:

↑↓X↓Y↓Z (сокращенно обозначим: **↕XYZ**).

Переход АМП из одного состояния в другое указывается явно операцией **"Состояние t"**, где **t** - новое состояние автомата (будем сокращать текст данной операции до **"[t]"**).

Сдвиг входной головки на один символ вправо относительно входной ленты задается операцией **"Сдвиг"** (сократим до **"→"**). После ее выполнения текущим символом становится следующий символ на входной ленте. Другой операцией над входной головкой является **"Держать",** которая не изменяет положение входной головки до следующего шага (можно просто не писать, если нет сдвига).

Переход или шаг автомата - это выполнение операций над стеком и входной головкой, а также изменение состояния. При этом необязательно, чтобы за один шаг происходили все изменения. Возможно: или входная головка останется на месте, или не произойдет операции над стеком, или не изменится состояние.

Распознаватель скобочных выражений

Рассмотрим одну из возможных реализация АМП, для задачи проверки корректности вложенности круглых скобок. Кратко опишем общий принцип работы автомата. Если входная головка читает "**(**", то в магазин заталкивается символ **A**. Если входная головка читает "**)**", то из магазина выталкивается содержащийся там символ. Цепочка отвергается, если:

1. На входе остаются правые скобки, а магазин пуст.
2. Входная цепочка прочитана до конца, а в магазине остаются символы **A**, соответствующие левым скобкам, для которых не нашлось пары.

Определим данный АМП следующим образом:

Множество входных символов: **{ (,), ⊣}**.

Множество магазинных символов: **{ A, ▽}**

Множество состояний: **t**, где t является также и начальным состоянием автомата, раз оно единственное.

Переходы:

1. **(, A, S ⇒ ↓A, S, →**
2. **(, ▽, S ⇒ ↓A, S, →**
3. **), A, S ⇒ ↑, S, →**
4. **), ▽, S ⇒ Отвергнуть**
5. **⊣, A, S ⇒ Отвергнуть**
6. **⊣, ▽, S ⇒ Допустить**

В начальном состоянии магазин содержит только маркер дна (▽).

Из представленного описания видно, что поведение автомата имитирует ранее рассмотренный метод распознавания с использованием счетчика. Только вместо счетчика используются

"палочки" (раньше с помощью палочек учили считать в школе). Эти палочки могут записываться в стек и стираться из него, отражая разность между прочитанными открывающимися и закрывающимися скобками. Работу данного АМП можно рассмотреть на примере распознавания нескольких цепочек. Пусть, первая цепочка будет иметь следующий вид:

(( ) ( ))

Тогда осуществляемые автоматом переходы можно представить в виде следующей последовательности текущих состояний его устройств.

Номер шага	**Содержимое стека**	**Состояние автомата**	**Остаток входной цепочки**	**Номер применяемого правила**
1	▽	**[t]**	**(( ) ( ))** ⊣	**2**
2	▽**A**	**[t]**	**( ) ( ))** ⊣	**1**
3	▽ **A A**	**[t]**	**) ( ))** ⊣	**3**
4	▽ **A**	**[t]**	**( ))** ⊣	**1**
5	▽ **A A**	**[t]**	**))** ⊣	**3**
6	▽ **A**	**[t]**	**)** ⊣	**3**
7	▽	**[t]**	⊣	**Допустить**

В приведенном примере цепочка оказалась распознанной. Следующий пример раскрывает поведение автомата при распознавании цепочки, содержащей большее число правых круглых скобок чем левых:

( )))

Таблица переходов и состояний в этом случае будет выглядеть следующим образом:

Номер шага	Содержимое стека	Состояние автомата	Остаток входной цепочки	Номер применяемого правила
1	▽	**[t]**	**( ))) ⊣**	**2**
2	▽ **A**	**[t]**	**))) ⊣**	**3**
3	▽	**[t]**	**)) ⊣**	**Отвергнуть**

Аналогичные таблицы можно представить и для других вариантов входных цепочек.

Устройство управления данного автомата с магазинной памятью можно представить в виде следующей таблицы переходов.

Магазинные символы	**Входные символы**			
	(	**)**	**⊣**	
A	**↓A, →**	**↑, →**	**Отвергнуть**	
▽	**↓A, →**	**Отвергнуть**	**Допустить**	

Такая таблица является типичной формой представления одного внутреннего состояния для любого АМП. Если у автомата имеется несколько внутренних состояний, то для каждого из них строится такая таблица переходов. В большинстве реальных случаев АМП имеют только одно состояние.

Общая связь между грамматиками и автоматами с

магазинной памятью

Эта связь давно доказана и рассмотрена в специальной литературе для различных классов грамматик и методов разбора. В рамках данной работы нет особой необходимости повторять материал или его выдержки из толстых книг. Можно только сослаться на основные источники информации, позволяющие получить фундаментальную подготовку по формальным грамматикам и методам разбора [Ахо78, Льюис]. Однако, и в этих книгах рассматриваются только такие грамматики и автоматы, которые могут быть эффективно реализованы. Из всего разнообразия грамматик, используемых для построения распознавателей, нас будут интересовать только КС(1) грамматики, ориентированные на нисходящий левый разбор, при котором входная цепочка просматривается слева направо. Использование данных грамматик позволяет создавать достаточно мощные и понятные языки программирования.

Следует пояснить, почему дальнейшее изучение ограничивается только нисходящими методами разбора. Методы, используемые при нисходящем разборе, достаточно универсальны и разнообразны. Применение восходящего разбора позволяет использовать более мощные КС(1) грамматики, в том числе и грамматики с левой рекурсией, которые при нисходящем разборе использовать невозможно. Возникают проблемы преобразования таких грамматик в грамматики с правой рекурсией, ориентированные на нисходящий разбор. Такое преобразование не всегда очевидно. Однако, применение диаграмм Вирта для представления синтаксиса языков программирования позволяет легко заменить все левые рекурсии на итерации и использовать полученные правила для нисходящего разбора. Поэтому, на мой взгляд, при практической разработке трансляторов, не имеет особого смысла использовать восходящий разбор. Забегая вперед, отмечу, что не вижу особого смысла использовать и автоматы с магазинной памятью, так как можно использовать другую модель распознавателя, более близкую по смыслу к диаграммам Вирта. Эта иерархически порождаемые конечные автоматы (пока я считаю, что для построения трансляторов она придумана и введена мною).

Для того, чтобы рассмотреть зависимость, существующую между КС(1) грамматиками и нисходящими распознавателями с магазинной памятью, обратимся к материалу, изложенному в книге Льюиса, Розенкранца и Стринза [Льюис]. Нет смысла выдумывать то, что уже изложено в виде, доступном для понимания.

Связь между S-грамматикой и автоматом с магазинной памятью

Не все КС-грамматики пригодны для построения нисходящего детерминированного АМП, так как многие из них могут порождать одну и ту же терминальную цепочку различными левыми выводами. Это говорить об их неоднозначности и невозможности использования для детерминированного разбора. Однако, определены и изучены такие классы грамматик, которые поддерживают нисходящий детерминированный разбор. Наиболее простыми из них являются S-грамматики.

КС-грамматика называется **S-грамматикой** (а также раздельной, или простой) тогда и только тогда, когда выполняются два условия:

1. Правая часть каждого правила начинается терминалом $\mathbf{T_i}$.

2. Если два правила имеют совпадающие левые части, то правые части этих правил должны начинаться различными терминальными символами.

Пример

Рассмотрим две грамматики, которые задают один и тот же язык.

Грамматика $\mathbf{G_{7.2}(S)}$, Содержит правила P:

1. $S \rightarrow aT$
2. $S \rightarrow TbS$
3. $T \rightarrow bT$
4. $T \rightarrow ba$

Это не S-грамматика, так как правило 2 начинается с нетерминала, что нарушает условие 1, а правила 3, 4 имеют совпадающие левые части и начинаются с одинаковых терминальных символов.

S-грамматика $\mathbf{G_{7.3}(S)}$ для этого же языка может использовать для порождения цепочек следующие правила:

1. $S \rightarrow abR$
2. $S \rightarrow bRbS$
3. $R \rightarrow a$
4. $R \rightarrow bR$

Использование данной грамматики позволяет упростить разбор, используя для этого соответствующий АМП.

Обобщенный алгоритм построения нисходящего АМП для S - грамматики

Построение нисходящего автомата с магазинной памятью для заданной S-грамматики осуществляется по следующему обобщенному алгоритму.

1. Множество входных символов автомата - это множество терминальных символов грамматики, расширенное концевым маркером.
2. Множество магазинных символов состоит из маркера дна, нетерминальных символов грамматики и терминалов, встречающихся в правых частях правил, за исключением тех, которые занимают только крайнюю левую позицию.
3. В начальном состоянии магазин содержит маркера дна и начальный нетерминал.
4. Устройство управления АМП, имеет одно состояние и описывается управляющей таблицей, строки которой помечены магазинными символами, столбцы - входными символами, а элементы таблице описываются в ниже следующих пунктах.
5. Каждому правилу грамматики ставится в соответствие элемент (клетка) таблицы. Правило должно иметь вид:

$$\mathbf{A \rightarrow b\alpha},$$

где **A** - нетерминал, **b** - терминал, α - цепочка из терминалов и нетерминалов. Этому

▽ **Отвергнуть Отвергнуть Допустить**

правилу будет соответствовать элемент в строке **A** и столбце **b**:

Каким образом найдена представленная последовательность шагов построения АМП? Что-то получили с использованием дедукций, что-то - индукции. Остальное высосали из пальца. Если же присмотреться внимательнее к уже предложенным шагам, то можно понять, как они появились. В основе всего лежат особенности S-грамматики. Рассмотрим левый вывод терминальной цепочки **bbababa**. Он будет существовать следующим образом:

Заменить (α_i^r), Сдвиг

где α_i^r - цепочка α_i, записанная в обратном порядке (для того, чтобы удовлетворить соглашению, что верхний символ находится справа при принятом способе записи стека, когда его дно расположено слева).

$S \Rightarrow bRbS \Rightarrow bbRbS \Rightarrow bbaRbS \Rightarrow bbabaR \Rightarrow bbababa$

Процедура построения нисходящего дерева разбора, представленная на рис. 7.1, показывает, что на каждом шаге правило для дерева выбирается однозначно и увеличивает количество распознанных символов на один.

Заменить ε, Сдвиг

используется

Вытолкнуть, Сдвиг.

6. Если магазинным символом является терминал **b**, то элементом таблицы в строке **b** и столбце **b** будет

Вытолкнуть, Сдвиг

7. Элементом таблицы, который находится в строке маркера дна и столбце концевого маркера, является

Допустить

8. Элементы таблицы, не описанные ни в одном из пунктов 5, 6, 7 заполняются операцией

Отвергнуть

Используя данное описание, построим автомат для S- грамматики $\mathbf{G_{8.3}(S)}$:

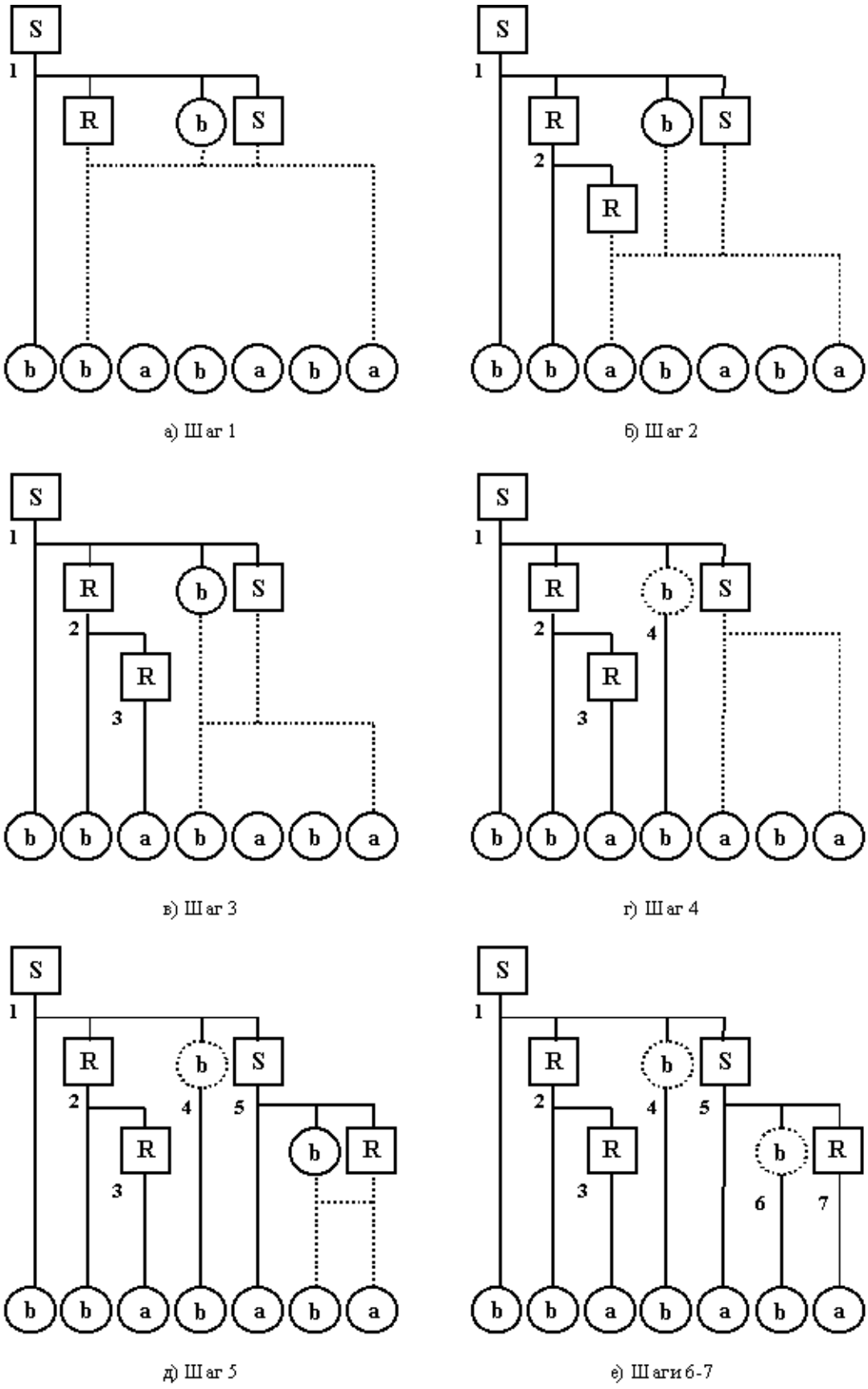

Рис. 7.1. Нисходящий разбор слева направо с использованием S-грамматики.

Этот выбор определяется грамматикой, которая четко устанавливает соответствие между