К.П. Голиков
«Системы программирования»

Тема: Синтаксический анализ.

Чтобы определить принадлежит ли заданная цепочка терминалов языку грамматики, синтаксический анализатор должен восстановить дерево разбора этой цепочки.

Дерево разбора играет ключевую роль на последующих этапах компиляции – этапах семантического анализа и генерации целевого кода.

Существует две стратегии разбора бесконтекстных языков - разбор сверху-вниз (нисходящий анализ) и разбор снизу-вверх(восходящий анализ).

Нисходящий анализ в основном применяют в тех случаях, когда анализатор строят вручную.

Для восходящих анализаторов разработано большое количество инструментальных средств автоматизированного конструирования.

Восходящий анализатор пытается восстановить дерево разбора, начиная с листьев(снизу), последовательно продвигаясь к корню(вверх). Этот процесс называется сворачиванием цепочки к стартовому символу грамматики.

На каждом шаге свертки цепочка просматривается слева направо и в ней выделяется правая часть некоторой продукции. Правая часть замещается нетерминалом левой части, и по завершении процесса мы получим цепочку, состоящую из одного только стартового символа.

Основой **правосентенциальной формы называется правая часть продукции, которая была применена на последнем шаге правого порождения этой сентенциальной формы.**

Понятие *основа* фиксирует не только правую часть продукции, но и ее местоположение в сентенциальной форме. Из определения, в частности следует, что справа от основы в сентенциальной форме располагаются только терминалы.

Именно основу ищет восходящий анализатор в цепочке и заменяет нетерминалом левой части продукции.

Последовательность шагов свертки в точности соответствует обращенному правому порождению цепочки.

Восходящий анализатор реализуется с помощью автомата с магазинной памятью, работающего по принципу перенос-свертка (ПС-анализатор).

На вход автомата подается цепочка терминалов β, заканчивающаяся маркером #.

Память представлена стеком, в который загружается цепочка из терминалов и нетерминалов.

Дно стека также отмечено маркером #.

В начале работы стек пуст.

СТЕК	ВХОД
#	β#

Анализатор переносит терминалы из входной цепочки в стек, пока в стеке не окажется основа. Затем он заменяет основу нетерминалом левой части продукции, т.е. производит свертку. Если в результате в стеке оказывается основа, производится еще одна свертка и т.д. Свертки выполняются до тех пор, пока это возможно, затем продолжается перенос терминалов. Анализатор заканчивает работу, придя в конфигурацию

СТЕК	ВХОД
#S	#

Рис.1

В стеке стартовый символ S, на входе пустая цепочка терминалов. В этом случае говорят, что анализатор *допускает* цепочку.

Существует две проблемы при разработке ПС-анализаторов. Первая – как обнаружить появление основы в стеке, вторая – как выбрать продукцию для свертки.
Имеется несколько схем конструирования ПС-анализаторов, отличающихся структурой управляющих таблиц и методикой их построения.

В инструментальных системах автоматизированного конструирования синтаксических анализаторов широко используются методы LR-анализа. В рамках этого практического курса вы изучите и на учебном тренажере освоите один из самых простых методов семейства LR – метод SLR-анализа.

Активным префиксом **правосентенциальной формы называется та ее часть, которая находится в стеке ПС-анализатора.**

Активный префикс накапливает информацию об уже разобранной части входной цепочки.

Метод SLR-анализа основан на фундаментальном утверждении о том, что активные префиксы образуют регулярный язык, т.е. язык детерминированного конечного автомата.

Доказательство этого утверждения включает алгоритм построения автомата для любой бесконтекстной грамматики. Не вдаваясь в детали этого алгоритма, я покажу, как выглядит диаграмма переходов автомата для уже знакомой вам грамматики G0. Грамматика порождает цепочки, составленные из терминалов ‘a’ и сбалансированного набора скобок.
S – стартовый символ грамматики. Вслед за правой частью записан уникальный номер продукции.

S -> L #1
L -> H E) #2
H -> (#3 | H E #4
E -> a #5 | L #6

На рис.2 показана диаграмма переходов автомата. При построении диаграммы применены те же соглашения, что и для диаграмм лексического анализа. Кружком обозначено обычное состояние, квадратом – заключительное.

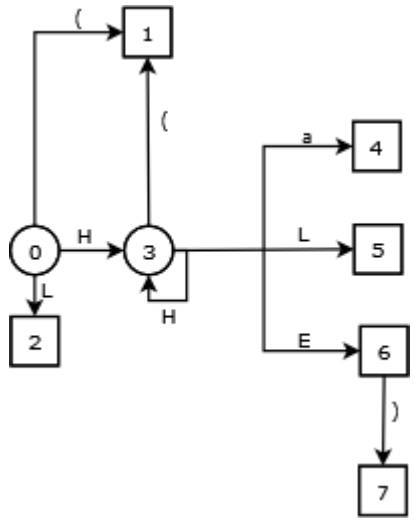

Рис.2

Алгоритм построения автомата гарантирует, что, переходя из начального в заключительное состояние, он допускает активный префикс, оканчивающийся основой сентенциальной формы.

Например, для цепочки ‘(a)’ обращенное правое порождение имеет вид: (a) <- Ha) <- HE) <- L <- S (подчеркнуты основы). Префикс ‘(‘ переводит автомат в состояние 1, префикс ‘Ha’ в состояние 3, префикс ‘HE)’ в состояние 6, префикс ‘L’ в состояние 2. Все перечисленные состояния заключительные.

Таким образом, автомат обнаруживает появление основы в стеке ПС-анализатора. Но для полного управления анализатором этого недостаточно. Необходимо добавить информацию о действии, которое анализатор должен выполнить, находясь в том или ином состоянии. Основных действий два – «перенос» или «свертка». Отобразим эту информацию непосредственно на диаграмме.

Действие «перенос» ассоциируется с каждой дугой диаграммы, помеченной терминалом грамматики, и никакого визуального представления не требует.

Действию «свертка» сопоставим дополнительный узел диаграммы, обозначенный ромбом. Внутри ромба записан уникальный номер продукции, применяемой для свертки.
Узел свертки соединим с узлом заключительного состояния дугой без стрелки и над дугой запишем набор терминалов, образующий *множество предпросмотра* для свертки. В методе SLR-анализа в

качестве множества предпросмотра используется множество FOLLOW(A), где A нетерминал левой части продукции.

FOLLOW(A) – это множество терминалов, которые могут следовать за A в какой-либо сентенциальной форме. Есть алгоритм, который по любой бесконтекстной грамматике строит множества FOLLOW(A) для всех ее нетерминалов.

В результате получим диаграмму управления SLR-анализатором, показанную на рис.3.

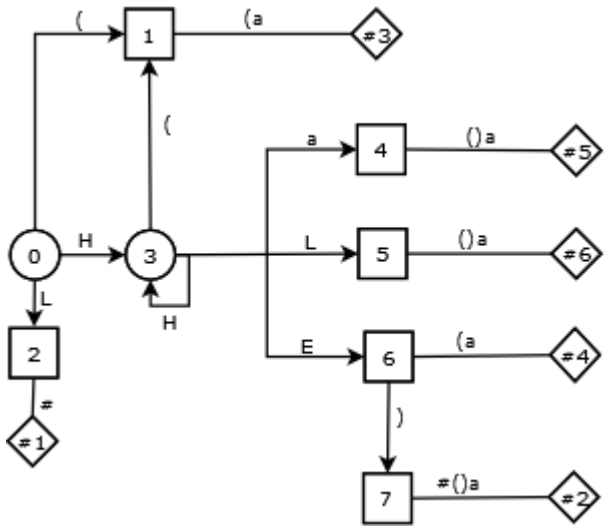

Рис.3

Анализатор начинает работу в состоянии 0 и переносит терминалы из входной цепочки в стек, переходя из одного состояния в другое в соответствии с диаграммой.

Кроме стека символов он заполняет еще один стек – стек состояний. Каждый раз, когда анализатор переходит в новое состояние, он помещает его номер на вершину стека состояний.

Перенос терминалов продолжается, пока анализатор не окажется в одном из заключительных состояний.

Обозначим через t первый терминал еще не проанализированной части входной цепочки.

Если t принадлежит множеству предпросмотра для свертки, ассоциированной с заключительным состоянием, то выполняется эта свертка.

Пусть A->β продукция для свертки и r – длина β.
Из стеков удаляются верхних r элементов, т.е. основа. На место основы в стек символов заносится нетерминал A. Если в результате анализатор придет в конфигурацию, показанную на рис.1, он останавливается и сообщает, что входная цепочка допустима.

Если S номер состояния, которое «всплывет» на вершину стека после удаления основы, то алгоритм построения диаграммы гарантирует, что из S есть переход в некоторое состояние SA, помеченный нетерминалом A.

SA помещается в стек состояний и анализатор продолжает сканировать входную цепочку из состояния SA, начиная с терминала t.

Если в каком-то состоянии анализатор не может выбрать действие для терминала t, он останавливается и сигнализирует об ошибке во входной цепочке.

Обратите внимание на состояние 6. В этом состоянии анализатор может выбрать одно из двух действий, либо перенос, либо свертку. Выбор однозначен, поскольку каждому действию соответствуют разные терминалы.

В общем случае диаграмма, построенная по грамматике, может содержать *конфликты* выбора действия.
Конфликт перенос-свертка возникает, когда одному и тому же терминалу соответствуют оба действия.
Конфликт свертка-свертка возникает, если в заключительном состоянии возможны две и более сверток, причем множества предпросмотра для сверток пересекаются.

Если в диаграмме нет конфликтов выбора действия, грамматика называется *SLR-грамматикой*.

Все грамматики, с которыми вы будете работать в ходе этого практикума, это SLR-грамматики.

В практической части лабораторной работы вы проведете серию экспериментов на учебном тренажере Pars. В тренажере на языке C++ реализована объектно-ориентированная модель SLR-анализатора.

Литература:

Ахо А.В., Сети Р., Ульман Д.Д.
Компиляторы: принципы, технологии и инструменты. - М.: Вильямс, 2003.