Глава 4. Восходящий синтаксический анализ

4.7. Синтаксический анализ

Работа синтаксического анализатора типа перенос-сверка совершенно не зависит от *LR*-метода, использованного для построения таблицы разбора, и от разбираемого языка (грамматика языка определяет содержимое таблицы, но не влияет на сам алгоритм анализа). Главное требование – *LR*-таблица разбора не должна содержать конфликтов. *LR*-анализатор считывает по одному символы входной строки слева направо и в процессе анализа переходит из одного состояния в другое. При этом используется два стека: стек символов (при практической реализации в нем нет необходимости) и стек состояний. Анализатор находится в состоянии, хранящемся в вершине стека состояний. Следующий шаг синтаксического анализатора зависит от элемента таблицы разбора, позиция которого определяется текущим состоянием (находится в вершине стека состояний) и входным символом. В качестве входного символа может быть текущий символ (терминал) входной строки или нетерминал из левой части продукции, для которой на предыдущем шаге выполнялась свертка.

Изначально в стеке состояний содержится исходное состояние I_0 анализатора. Процесс анализа продолжается до тех пор, пока не будет достигнуто успешное завершение (элемент *stop* в таблице разбора) или не обнаружится синтаксическая ошибка (пустая ячейка в таблице разбора, позиция которой может дать информацию о характере ошибки).

Рассмотрим работу анализатора для *LALR*(1)-грамматики, иллюстрирующей работу *LALR*(1)-метода, и соответствующей *LALR*(1)-таблицы разбора (см. табл. 4.11). Процесс разбора строки $dbadcdb\perp$, которая выводится в соответствии с правосторонней схемой

$$S\perp \Rightarrow AaBb\perp \Rightarrow AaCb\perp \Rightarrow AaCcdb\perp \Rightarrow Aadcdb\perp \Rightarrow dbadcdb\perp,$$

показан в табл. 4.14. Содержимое каждого стека представляется строкой, в которой самый правый символ находится в вершине стека, символ $\perp$ показывает дно стека. В стеке состояний указаны номера состояний.

LALR(1)-таблица разбора

Номер состояния	*S*	*A*	*B*	*C*	*a*	*b*	*c*	*d*	⊥
0	*stop*	*S*1		*S*2				*S*3	
1					*S*4				
2					*R*2		*S*5		
3					*R*6	*S*6	*R*6		
4			*S*7	*S*8				*S*9	
5								*S*10	
6					*R*3				
7						*S*11			
8						*R*4	*S*5		
9						*R*6	*R*6		
10					*R*5	*R*5	*R*5		
11									*R*1

$$S\perp \Rightarrow AaBb\perp \Rightarrow AaCb\perp \Rightarrow AaCcdb\perp \Rightarrow Aadcdb\perp \Rightarrow dbadcdb\perp,$$

Таблица 4.14. Процесс разбора строки *dbadcdb*⊥

Номер шага	Вводимая строка	Стек символов	Стек состояний	Выполняемые действия
1	*dbadcdb*⊥	⊥	⊥0	Перенос *S*3
2	*badcdb*⊥	⊥*d*	⊥0,3	Перенос *S*6
3	*adcdb*⊥	⊥*db*	⊥0,3,6	Сверка *R*3 для $A \to db$
4	*adcdb*⊥	⊥	⊥0	Входной символ *A*. Перенос *S*1
5	*adcdb*⊥	⊥*A*	⊥0,1	Перенос *S*4
6	*dcdb*⊥	⊥*Aa*	⊥0,1,4	Перенос *S*9
7	*cdb*⊥	⊥*Aad*	⊥0,1,4,9	Сверка *R*6 для $C \to d$
8	*cdb*⊥	⊥*Aa*	⊥0,1,4	Входной символ *C*. Перенос *S*8
9	*cdb*⊥	⊥*AaC*	⊥0,1,4,8	Перенос *S*5
10	*db*⊥	⊥*AaCc*	⊥0,1,4,8,5	Перенос *S*10
11	*b*⊥	⊥*AaCcd*	⊥0,1,4,8,5,10	Сверка *R*5 для $C \to Ccd$
12	*b*⊥	⊥*Aa*	⊥0,1,4	Входной символ *C*. Перенос *S*8
13	*b*⊥	⊥*AaC*	⊥0,1,4,8	Сверка *R*4 для $B \to C$
14	*b*⊥	⊥*Aa*	⊥0,1,4	Входной символ *B*. Перенос *S*7
15	*b*⊥	⊥*AaB*	⊥0,1,4,7	Перенос *S*11
16	⊥	⊥*AaBb*	⊥0,1,4,7,11	Сверка *R*1 для $S \to AaBb$
17	⊥	⊥	⊥0	Входной символ *S*. *stop*. Разбор успешно завершен

На шаге 1 входным является символ *d*, анализатор находится в состоянии 0. Поскольку в столбце *d* таблицы разбора для данного состояния содержится элемент *S*3, выполняется перенос: в стек символов помещается символ *d*, в стек состояний – состояние 3, и анализатор переходит в состояние 3. На шаге 2 входной символ – *b*, элемент таблицы – *S*6, следовательно, выполняется перенос: в стек символов заносится *b*, в стек состояний – 6, переход в состояние 6. На шаге 3 для состояния 6 и входного символа *a* в таблице указан элемент свертки *R*3, т. е. выполняется свертка по продукции $A \to db$. Поскольку в правой части этой продукции два символа, из стеков удаляются по два верхних элемента, т. е. из стека символов исключается основа. В результате в вершине стека состояний будет 0, т. е. анализатор переходит в состояние 0. Входным символом для следующего шага (шаг 4) будет нетерминал *A*. Последовательность выполнения действий свертки и переноса продолжается до тех пор, пока на шаге 17 элементом таблицы разбора не станет элемент *stop*. Это говорит о том, что разбор успешно завершен, входная строка принадлежит языку, т. е. анализатор принимает эту строку.

Пример обнаружения синтаксической ошибки при анализе строки $dbaab\perp$, не принадлежащей языку, показан в табл. 4.15.

Таблица 4.15. Процесс разбора строки *dbaab*⊥, не принадлежащей языку

Номер шага	Вводимая строка	Стек симво-лов	Стек состояний	Выполняемые действия
1	*dbaab*⊥	⊥	⊥0	Перенос *S*3
2	*baab*⊥	⊥*d*	⊥0,3	Перенос *S*6
3	*aab*⊥	⊥*db*	⊥0,3,6	Свертка *R*3 для $A \to db$
4	*aab*⊥	⊥	⊥0	Входной символ *A*. Перенос *S*1
5	*aab*⊥	⊥*A*	⊥0,1	Перенос *S*4
6	*ab*⊥	⊥*Aa*	⊥0,1,4	Синтаксическая ошибка

LALR(1)-таблица разбора

Номер состояния	*S*	*A*	*B*	*C*	*a*	*b*	*c*	*d*	⊥
0	*stop*	*S*1		*S*2				*S*3	
1					*S*4				
2					*R*2		*S*5		
3					*R*6	*S*6	*R*6		
4			*S*7	*S*8				*S*9	
5								*S*10	
6					*R*3				
7						*S*11			
8						*R*4	*S*5		
9						*R*6	*R*6		
10					*R*5	*R*5	*R*5		
11									*R*1

На шаге 6 в позиции таблицы разбора, соответствующей состоянию 4 и входному символу *a*, указан элемент ошибки (пустая ячейка), что говорит об обнаружении синтаксической ошибки. К этому моменту принята подстрока *dba*, свернутая в подстроку *Aa*. Характер этой ошибки можно определить, проанализировав строку таблицы разбора, соответствующую состоянию 4. В этой строке в столбцах *B, C* и *d* указаны элементы переноса, а во всех остальных – элементы ошибок. Это свидетельствует о том, что после принятой на настоящий момент подстроки *dba* (свернутой в процессе разбора в подстроку *Aa*) может следовать только подстрока, начинающаяся с *d*, либо подстрока, выводимая из нетерминалов *B* или *C*, т. е. с того же терминала *d*. Таким образом, для определения характера ошибки достаточно проанализировать только столбцы таблицы разбора, соответствующие терминалам, для состояния, в котором обнаружилась ошибка. Возможными очередными символами входной строки могут быть только терминалы, в столбцах которых не содержатся элементы ошибок.

4.8. Сравнение с *LL*-методом разбора

Основными достоинствами как *LL*-, так и *LR*-методов разбора являются их детерминированность и возможность обнаружения синтаксических ошибок, как правило, на самых ранних этапах анализа. Кроме того, оба метода позволяют включать в синтаксис действия для выполнения некоторых аспектов процесса трансляции.

Преимуществом *LR*-метода является то, что он применим к более широкому классу грамматик и языков и обычно не требует преобразований грамматик, которые практически всегда необходимы для *LL*-метода. Однако при наличии хорошего преобразователя грамматик сам факт необходимости преобразований грамматик в действительности не вызывает затруднений. В тех случаях, когда преобразователь грамматик не справляется с какой-либо конструкцией языка программирования, то часто эта конструкция не обладает и признаком *LR*(1). Поэтому для обоих методов возникает необходимость в ручном преобразовании грамматик. Это снижает значимость преимущества *LR*-метода перед *LL*-методом в части преобразования грамматик.

В отношении размеров таблиц разбора *LL*-метод существенно превосходит *LR*-метод. Однако использование методов оптимизации *LR*-таблиц разбора делает их размер примерно того же порядка, что и *LL*-таблицы разбора, но обычно это достигается за счет некоторого увеличения времени разбора.

Сравнение максимального, минимального и среднего времени разбора показывает, что *LL*-метод более эффективен, чем *LR*-метод.