Синтаксический анализ

Содержание

Деревья вывода

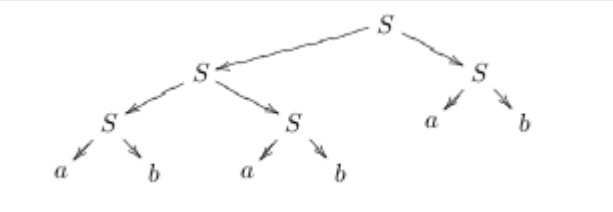
Рассмотрим КС-грамматику вида:

$$S \to SS, S \to ab, S \to aSb$$

В этой грамматике можно построить следуюший вывод:

$$S \to SS \to Sab \to SSab \to abSab \to ababab$$

Такому выводу можно поставить в соответствие некоторое дерево:



В общем случае выводам в КС-грамматике всегда можно сопоставить некоторое дерево, которое называется **деревом вывода** (**деревом разбора**). В вершинах такого дерева помещаются терминальные и нетерминальные символы грамматики. Корень дерева соответствует начальному нетерминальному символу. Каждому символу слова SS, на которое заменяется начальный символ на первом шаге вывода, ставится в соответствие вершина дерева, и к ней проводится дуга из корня. Далее подобные замены повторяются, пока не получим листья только из терминальных символов.

Деревья вывода

Вывод в контекстно-свободной грамматике называется **левосторонним или левым**, если на каждом шаге вывода заменяется самое левое из всех вхождений вспомогательных символов (то есть каждый шаг вывода имеет вид:

$$\alpha A \beta \Rightarrow \alpha \gamma \beta$$

где α состоит только из терминальных символов, A - первый нетерминал и имеется правило $A \to \beta$
Правосторонний вывод определяется аналогично.
Например, на предыдущем слайде вывод не является левосторонним.

Лемма

Для каждого слова, выводимого в КС-грамматике, существует левосторонний вывод.

Например, вот такой вывод по предыдущей грамматике является левосторонним:

$$S \Rightarrow SS \Rightarrow SSS \Rightarrow abSS \Rightarrow ababS \Rightarrow ababab$$

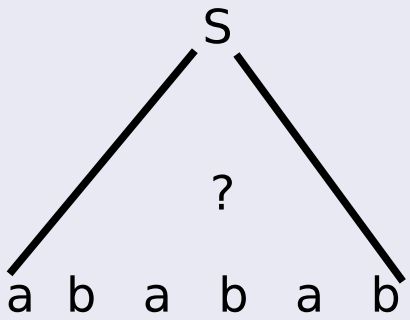
КС-грамматика называется **неоднозначной**, если существует слово, которое имеет два или более левосторонних вывода. В противном случае она называется **однозначной**.

Понятие синтаксического анализа

Задача синтаксического анализа состоит в том, чтобы для заданной КС-грамматики и выводимой в ней цепочки терминальных символов восстановить ее вывод из начального нетерминального символа.

$$S \Rightarrow ? \Rightarrow ababab$$

Что должно стоять вместо вопросительного знака?
Синтаксический анализ можно также понимать и как восстановление дерева вывода для заданной входной цепочки. Для этого дерева известны листья и корень. Требуется восстановить его середину.



Принципиально существуют две основные стратегии: восстановление дерева от корня к листьям - **нисходящий синтаксический анализ**; восстановление дерева от листьев к корню - **восходящий синтаксический анализ**.
Реализация каждой из них связана с понятием автомата с магазинной памятью.

Синтаксические анализаторы

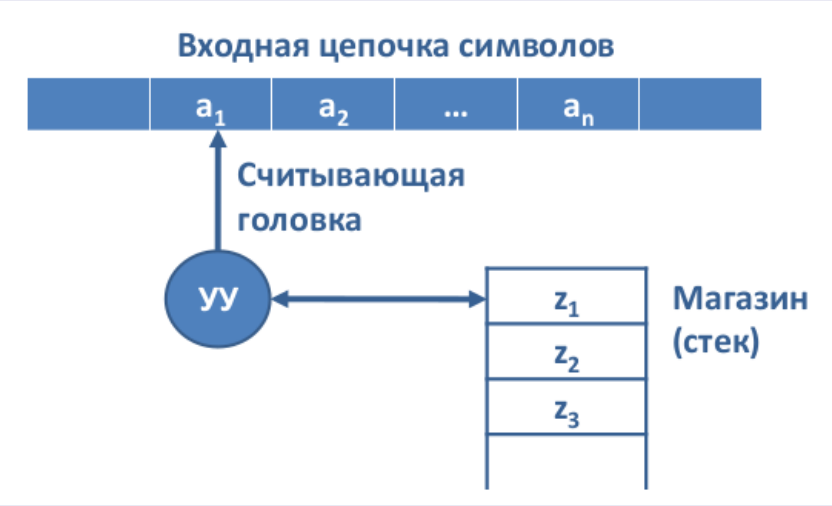
Синтаксический анализатор (синтаксический разбор) - это часть компилятора, которая отвечает за выявление и проверку синтаксических конструкций входного языка. В задачу синтаксическоrо анализа входит:

- поиск и выделение синтаксических конструкции в тексте исходной программы;
- установка типа и проверка правильности каждой синтаксической конструкции;
- представление синтаксических конструкций в виде, удобном для дальнейшей генерации текста результирующей программы.

Синтаксический анализатор - это основная часть компилятора на этапе анализа. Без выполнения синтаксическоrо разбора работа компилятора бессмысленна, в то время как лексический разбор, в принципе, является необязательной фазой. Все задачи по проверке синтаксиса входноrо языка могут быть решены на этапе синтаксическоrо разбора. Лексический анализатор только позволяет избавить сложный по структуре синтаксический анализатор от решения примитивных задач по выявлению и запоминанию лексем исходной программы. Выходом лексическоrо анализатора является таблица лексем. Эта таблица образует вход синтаксическоrо анализатора, который исследует только один компонент каждой лексемы ее тип. Остальная информация о лексемах используется на более поздних фазах компиляции при семантическом анализе.

Автоматы с магазинной памятью

Для построения дерева разбора используют так называемые автоматы с магазинной памятью, которые являются распознавателями для КС-языков. На рисунке представлен общий вид такого автомата.



Магазинный автомат — это, по существу, недетерминированный конечный автомат с возможными переходами без входных символов и одним дополнением — магазином, в котором хранится цепочка «магазинных символов». Присутствие магазина означает, что в отличие от конечного автомата магазинный автомат может помнить бесконечное количество информации. Однако в отличие от универсального компьютера, магазинный автомат имеет доступ к информации в магазине только с одного его конца в соответствии с принципом “последним пришел — первым ушел” (“last-in-first-out”).

Автамоты с магазинной памятью

Магазинный автомат (МП-автомат) может обозревать символ на вершине магазина и совершать переход на основе текущего состояния, входного символа и символа на вершине магазина. Он может также выполнить «спонтанный» переход, используя ϵ в качестве входного символа.

За один переход автомат совершает следующие действия.

1. Читает и пропускает входной символ, используемый при переходе. Если в качестве входа используется ϵ , то входные символы не пропускаются.
2. Переходит в новое состояние, которое может и не отличаться от предыдущего.
3. Заменяет символ на вершине магазина некоторой цепочкой. Цепочкой может быть ϵ, что соответствует снятию с вершины магазина. Это может быть тот же символ, который был ранее на вершине магазина, т.е. магазин не изменяется. Автомат может заменить магазинный символ, что равносильно изменению вершины без снятий и заталкиваний. Наконец, символ может быть заменен несколькими символами — это равносильно тому, что (возможно) изменяется символ на вершине, а затем туда помещаются один или несколько новых символов.

МП-автомат распознает входную цепочку, если после ее завершения автомат перейдет в одно из заключительных сосотояний и его магазин будет пустым.

МП-автомат называется детерминированным, если из каждой его конфигурации возможно не более одного перехода в следующую конфигурацию.

Формальное определение автомата с магазинной памятью

Определение

МП-автомат - это кортеж семи объектов: $P =< Q, \Sigma, M, \delta, q_o, Z_0, F >$

Q *- конечное множество состояний;*

Σ *- конечное множество входных символов;*

M *- конечный магазинный алфавит;*

δ *- функция переходов:* $\delta(q, \sigma, X) \to (p, \gamma)$*, где* q *- состояние из множества* Q*,* σ *- входной символ из алфавита* Σ*, либо пустая цепочка* ϵ*, которая не принаждежит* Σ*,* X *- магазинный символ из* M *на вершине стека,* p *- новое состояние из* Q*, в которое переходит МП-автомат,* γ *- цепочка магазинных символов, замещающая* X *на вершине стека магазина. Например, если* $\gamma = \epsilon$*, то магазинный символ просто снимается, если* $\gamma = X$*, то магазин не изменяется, если* $\gamma = YZ$*, то* X *заменяется на* Z*, и* Y *помещается в магазин.*

q_0 *- начальное состояние МП-автомата;*

Z_0 *- начальный магазинный символ (маркер дна);*

F *- множество заключительных состояний.*

Формальное определение автомата с магазинной памятью

Теорема

По каждой КС-грамматике $G = (T, N, S, R)$ можно построить недетерминированный МП-автомат, распознающий язык, порождаемый G.

Доказательство.

Определим МП-автомат следующим образом: множество состояний составляют три состояния: начальное q_0, рабочее q и заключительное f. Множество Σ МП-автомата совпадает с множеством терминальных символов грамматики, алфавит магазина составляют все нетерминальные символы, для каждого терминального символа $t \in T$ вводится символ $t_0 \in T_0$, тогда $M = N \cup T_0$. Также вводится функция $h(N_1) = N_1, h(t) = t_0, N_1 \in N$.

Функция переходов δ определяется следущим образом:

- $\delta(q_0, \epsilon, Z_0) = (q, SZ_0)$ - вначале в вершину стека помещаем начальный нетерминал грамматики G.
- $\forall t \in T : \delta(q, t, h(t)) = (q, \epsilon)$ - если в вершине стека символ $t_0 = h(t)$, то автомат переходит к следующему входному символу, выбрасывая из магазина верхний символ.
- $\forall A \in M : \forall A \to \alpha \in R : \delta(q, \epsilon, A) = (q, h(\alpha))$ - если в верхине стека стоит нетерминал A, то МП-автомат, не читая очередного входного символа, заменяет его в магазине одной из альтернатив этого нетерминала.
- $\delta(f, \epsilon, Z_0)$ - если в магазине больше нет символов, то переходим в заключительное состояние.

Можно понять, что один из путей такого недетерминированного МП-автомата смоделирует процесс порождения входной цепочки. □

Примеры построения МП-автомата

Пример 1

Построим МП-автомат по грамматике

$$S \to SS|(S)|\epsilon$$

Определим, как выше в теореме, q_0 - начальное состояние, q - рабочее состояние, f - заключительное состояние. Магазиный алфавит включает два символа $(_0,)_0$, соответствующих терминалам $(,)$. Тогда автомат задается следующим образом:

$$< \{q_0, q, f\}, \{(,)\}, \{S, (_0,)_0\}, q_0, Z_0, \delta, \{f\} >$$

Фунция переходов определяется следующим образом:
$\delta(q_0, \epsilon, Z_0) = (q, SZ_0)$ - в магазин записываем начальный нетерминал, автомат переходит в состояние q.
$\delta(q, (, (_0) = (q, \epsilon)$ - если терминал входной цепочки и вершина стека совпадают, выбрасываем символ из магащина и переходим к следующему входному символу;
$\delta(q,),)_0) = (q, \epsilon)$ - аналогично;
$\delta(q, \epsilon, S) = (q, SS)$ - в соответствии с правилами грамматики;
$\delta(q, \epsilon, S) = (q, \epsilon)$ - в соответствии с правилами грамматики;
$\delta(q, \epsilon, S) = (q, (_0S)_0)$ - в соответствии с правилами грамматики;
$\delta(q, \epsilon, Z_0) = (f, Z_0)$ - при пустом магазине входная цепочка распознана, если она кончилась.
Ясно, что пользоваться таким недетерминированным автоматом с магазинной памятью сложно для реального распознавания цепочек входного языка.

Примеры построения МП-автомата

Пример 2

Для грамматики $G(\{+,(,),a\},\{S,A\},\{S \to S+A|A, A \to (S)|a\},\{S\})$ построить МП-автомат.
Для реализации МП-автомата вида:

$$< Q, \Sigma, M, \delta, q_o, Z_0, F >$$

определим:
$Q = \{q\}, q_0 = q$ - одно единственное состояние МП-автомата;
$\Sigma = \{+,(,),a\}$ - множество входных символов совпадает с терминальными символами;
$M = \{S, A, +_0, (_0,)_0, a_0\}$ - множество магазинных символов;
$Z_0 = S$ - начальный магазинный символ;
$F = \{q\}$ - множество конечных состояний.
Далее определяем функцию переходов в соответствии с правилами грамматики:

$$\delta(q, \epsilon, S) = (q, S +_0 A)$$

$$\delta(q, \epsilon, S) = (q, A)$$

$$\delta(q, \epsilon, A) = (q, (_0 S)_0)$$

$$\delta(q, \epsilon, A) = (q, a_0)$$

$$\delta(q, t, t_0) = (q, \epsilon), t \in \Sigma$$

Рассмотрим работу данного МП-автомата при распознавании строки «(a)».

Примеры построения МП-автомата

Пример 2

Номер конфигурации	Текущее состояние	Входная строка	Содержимое магазина
1	q	(a)	S
2	q	(a)	A
3	q	(a)	$({}_0S)_0$
4	q	$a)$	$S)_0$
5	q	$a)$	$A)_0$
6	q	$a)$	$a_0)_0$
7	q	$)$	$)_0$
8	q	ϵ	ϵ

Как видно, работа МП-автомата реализует левосторонний вывод и нисходящий синтаксический разбор.

Работа автомата недетерминирована.

Примеры построения МП-автомата

Пример 3

Для грамматики предыдущего примера: $G(\{+,(,),a\},\{S,A\},\{S \to S+A|A, A \to (S)|a\},\{S\})$ построить МП-автомат, реализующий правосторонний разбор.

Данный автомат имеет единственное текущее состояние q и одно заключительное состояние r, в котором стек пуст. Стек содержит левую часть текущего правила. Первоначально в стек помещается специальный магазинный символ, маркер пустого стека $\#$. На каждом шаге автомат по правилу грамматики замещает нетерминалом строку верхних символов стека или дописывает в вершину входной символ.

Отличие данного автомата от предыдущего будет состоять в том, что он работает в другую сторону: для каждого правила $A \to \beta$ формируется функция переходов $\delta(q,\epsilon,\beta) = (q,A)$, предписывающая заменять правую часть правила в вершине стека нетерминалом из левой части, независимо от текущего символа входной строки.

Аналогично, для каждого терминала t необходимо сформировать магазинную функцию вида $\delta(q,t,\epsilon) = (q,t)$, которая помещает символ входной строки в вершину стека, если там нет правой части правила, и перемещает читающую головку.

Также следует предусмотреть магазинную функцию для перевода автомата в заключительное состояние $\delta(q,\epsilon,\#S) = (r,\epsilon)$.

Примеры построения МП-автомата

Пример 3

Для реализации МП-автомата вида:

$$< Q, \Sigma, M, \delta, q_o, Z_0, F >$$

определим:
$Q = \{q, r\}, q_0 = q$ - два состояния МП-автомата;
$\Sigma = \{+, (,), a\}$ - множество входных символов совпадает с терминальными символами;
$M = \{S, A, +_0, (_0,)_0, a_0\}$ - множество магазинных символов;
$Z_0 = \#$ - начальный магазинный символ;
$F = \{r\}$ - множество конечных состояний.
Далее определяем функцию переходов в соответствии с правилами грамматики:

$$\delta(q, \epsilon, S +_0 A) = (q, S)$$

$$\delta(q, \epsilon, A) = (q, S)$$

$$\delta(q, \epsilon, (_0 S)_0) = (q, A)$$

$$\delta(q, \epsilon, a_0) = (q, A)$$

$$\delta(q, t, \epsilon) = (q, t_0), t \in \Sigma$$

$$\delta(q, \epsilon, \#S) = (r, \epsilon)$$

Примеры построения МП-автомата

Пример 3

Номер конфигурации	Текущее состояние	Входная строка	Содержимое магазина
1	q	(a)	$\#$
2	q	$a)$	$\#(_0$
3	q	$)$	$\#(_0a_0$
4	q	$)$	$\#(_0A$
5	q	$)$	$\#(_0S$
6	q	ϵ	$\#(_0S)_0$
7	q	ϵ	$\#A$
8	q	ϵ	$\#S$
9	q	ϵ	ϵ

Как видно, работа МП-автомата реализует правосторонний вывод и восходящий синтаксический разбор.

Работа автомата детерминирована.

Варианты синтаксических анализаторов

Построение синтаксическоrо анализатора - это более творческий процесс, чем построение лексическоrо анализатора. Этот процесс не всегда может быть полностью формализован.
Имея грамматику входноrо языка, разработчик синтаксическоrо анализатора должен в первую очередь выполнить ряд формальных преобразований над этой грамматикой, облегчающих построение распознавателя. После этоrо он должен проверить, подпадает ли полученная грамматика под один из известных классов КС-языков, для которых существуют линейные распознаватели. Если такой класс найден, можно строить распознаватель (если найдено несколько классов, выбрать тот, для котороrо построение распознавателя проще, либо построенный распознаватель обладает лучшими характеристиками). Если же такой класс КС-языков найти не удалось, то разработчик должен попытаться выполнить над грамматикой некоторые преобразования, чтобы привести ее к одному из известных классов. Вот эти преобразования не могут быть описаны формально, и в каждом конкретном случае разработчик должен попытаться найти их сам.
Только в том случае, когда в результате всех этих действий не удалось найти соответствующий класс КС-языков, разработчик вынужден строить универсальный распознаватель. Характеристики такоrо распознавателя будут существенно хуже, чем у линейноrо распознавателя, в лучшем случае удается достичь квадратичной зависимости времени работы распознавателя от длины входной цепочки. Такие случаи бывают редко, поэтому все современные компиляторы построены на основе линейных распознавателей (иначе время их работы было бы недопустимо велико).
Нисходящий универсальный распознаватель использует алгоритм подбора альтернатив.
Восходящий универсальный распознаватель использует алгоритм «сдвиг-свертка».

Варианты синтаксических анализаторов

Поскольку моделируется недетерминированный МП-автомат, то на некотором шаге работы моделирующего алгоритма существует возможность возникновения нескольких возможных следующих состояний автомата. В таком случае есть два варианта реализации алгоритма.

В первом варианте на каждом шаге работы алгоритм должен запоминать все возможные следующие состояния МП-автомата, выбирать одно из них, переходить в это состояние и действовать так до тех пор, пока либо не будет достигнуто конечное состояние автомата, либо автомат не перейдет в такую конфигурацию, когда следующее состояние будет не определено. Если достигнуто одно из конечных состояний входная цепочка принята, работа алгоритма завершается. В противном случае алгоритм должен вернуть автомат на несколько шагов назад, когда еще был возможен выбор одного из набора следующих состояний, выбрать другой вариант и про моделировать поведение автомата с этим условием. Алгоритм завершается с ошибкой, когда все возможные варианты работы автомата перебраны и при этом не было достигнуто ни одного из возможных конечных состояний.

Во втором варианте алгоритм моделирования МП-автомата должен на каждом шаге работы при возникновении неоднозначности с несколькими возможными следующими состояниями автомата запускать новую свою копию для обработки каждого из этих состояний. Алгоритм завершается, если хотя бы одна из выполняющихся его копий достигнет одного из конечных состояний. При этом работа всех остальных копий алгоритма прекращается. Если ни одна из копий алгоритма не достигла конечного состояния МП-автомата, то алгоритм завершается с ошибкой.

Второй вариант реализации алгоритма связан с управлением параллельными процессами в вычислительных системах, поэтому сложен в реализации.

Рассмотрим реализацию этих алгоритмов.

Нисходящий распознаватель с возвратом на основе подбора альтернатив

Нисходящий распознаватель с возвратом на основе подбора альтернатив моделирует работу МП-автомата с одним состоянием q: $R(\{q\}, V, M, \delta, q, S, \{q\})$. Автомат распознает цепочки КС-языка, заданного КС-грамматикой $G(VT, VN, P, S)$. Входной алфавит автомата содержит терминальные символы грамматики: VT, а алфавит магазинных символов строится из терминальных и нетерминальных символов грамматики: $M = VT \cup VN$. Начальная конфигурация автомата определяется так: (q, α, S) автомат пребывает в своем единственном состоянии q, считывающая головка находится в начале входной цепочки символов $\alpha \in V^*$, в стеке лежит символ, соответствующий целевому символу грамматики S. Конечная конфигурация автомата определяется так: (q, ϵ, ϵ) автомат пребывает в своем единственном состоянии q, считывающая головка находится за концом входной цепочки символов, стек пуст.

Работу данного МП-автомата можно неформально описать следующим образом: если на верхушке стека автомата находится нетерминальный символ , то его можно заменить на цепочку символов α, если в грамматике языка есть правило $A \to \alpha$, не сдвигая при этом считывающую головку автомата (этот шаг работы называется подбор альтернативы); если же на верхушке стека находится терминальный символ , который совпадает с текущим символом входной цепочки, то этот символ можно выбросить из стека и передвинуть считывающую головку на одну позицию вправо (этот шаг работы называется выброс). Данный МП-автомат может быть недетерминированным, поскольку при подборе альтернативы в грамматике языка может оказаться более одного правила вида $A \to \alpha$, тогда функция $\delta(q, \epsilon, A)$ будет содержать более одного следующего состояния у МП-автомата будет несколько альтернатив.

Нисходящий распознаватель с возвратом на основе подбора альтернатив

Решение о том, выполнять ли на каждом шаге работы МП-автомата выброс или подбор альтернативы, принимается однозначно. Моделирующий алгоритм должен обеспечивать выбор одной из возможных альтернатив и хранение информации о том, какие альтернативы на каком шаге уже были выбраны, чтобы иметь возможность вернуться к этому шагу и подобрать другие альтернативы. Такой алгоритм разбора называется алгоритмом с подбором альтернатив.

Алгоритм использует также дополнительное состояние b (от back - назад), которое сигнализирует о выполнении возврата к уже прочитанной части входной цепочки. Для хранения уже выбранных альтернатив используется дополнительный стек L_2 , который может содержать следующую информацию: символы входного языка автомата; символы вида A_j - это означает, что среди всех возможных правил для символа была выбрана альтернатива с номером j.

Состояние алгоритма на каждом шаге определяется четырьмя параметрами: (Q, i, L_1, L_2), где q - текущее состояние автомата (q или b); i - положение считывающей головки во входной цепочке символов; L_1 - содержимое стека МП-автомата; L_2 - содержимое дополнительного стека.

Начальным состоянием алгоритма является состояние $(q, 1, S, \epsilon)$, где S - целевой символ грамматики. Алгоритм начинает свою работу с начального состояния и циклически выполняет шесть шагов до тех пор, пока не перейдет в конечное состояние или не обнаружит ошибку.

Нисходящий распознаватель с возвратом на основе подбора альтернатив

Алгоритм

1. Шаг «Разрастание». $(q, i, A\beta, \alpha) \to (q, i, \gamma_1\beta, \alpha A_1)$, если $A \to \gamma_1$ - это первая из всех возможных альтернатив для символа A.
2. Шаг «Успешное сравнение». $(q, i, a\beta, \alpha) \to (q, i+1, \beta, \alpha a)$, если a - это i-й символ во входной строке.
3. Шаг «Завершение». Если состояние соответствует $(q, n+1, \epsilon, \alpha)$, то разбор завершен, алгоритм заканчивает работу, иначе $(q, n+1, \epsilon, \alpha) \to (b, i, \epsilon, \alpha)$.
4. Шаг «Неуспешное сравнение». $(q, i, a\beta, \alpha) \to (b, i, a\beta, \alpha)$, если i-й символ входной строки не равен a.
5. Шаг «Возврат по входу». $(b, i, \beta, \alpha a) \to (b, i-1, a\beta, \alpha)$, если a - терминальный символ.
6. Шаг «Другая альтернатива». Исходное состояние $(b, i, \gamma_j\beta, \alpha A_j)$ действия:
   - перейти в состояние $(q, i, \gamma_{j+1}\beta, \alpha A_{j+1})$, если еще существует альтернатива $A \to \gamma_{j+1}$
   - сообщить об ошибке, если $A = S$ и не существует больше альтернатив для S.
   - иначе перейти в состояние $(q, i, A\beta, \alpha)$.

Цепочка вывода строится следующим образом: поместить в цепочку номер правила, соответствующий альтернативе $\to \gamma_j$, если в стеке содержится символ A_j (все терминальные символы, содержащиеся в стеке L_2 , игнорируются).

Нисходящий распознаватель с возвратом на основе подбора альтернатив

Пример

Пусть дана грамматика $G(\{+, -, /, *, a, b\}, \{S, R, T, F, E\}, P, S)$
P :

$$S \to T|TR$$

$$R \to +T| - T| + TR| - TR$$

$$T \to E|EF$$

$$F \to *E|/E| * EF|/EF$$

$$E \to (S)|a|b$$

Данная грамматика нелеворекурсивна. Построим вывод цепочки

$$a + (a * b)$$

Нисходящий распознаватель с возвратом на основе подбора альтернатив

Пример - решение

Итерация	Шаг алгоритма	Q	i	L_1	L_2
1	0	q	1	S	ϵ
2	1	q	1	T	S_1
3	1	q	1	E	S_1T_1
4	1	q	1	(S)	$S_1T_1E_1$
5	4	b	1	(S)	$S_1T_1E_1$
6	6	q	1	a	$S_1T_1E_2$
7	2	q	2	ϵ	$S_1T_1E_2a$
8	3	b	2	ϵ	$S_1T_1E_2a$
9	5	b	1	a	$S_1T_1E_2$
10	6	q	1	b	$S_1T_1E_3$
11	4	b	1	b	$S_1T_1E_3$
12	6	b	1	E	S_1T_1
13	6	q	1	EF	S_1T_2
14	1	q	1	$(S)F$	$S_1T_2E_1$
15	4	b	1	$(S)F$	$S_1T_2E_1$
16	6	q	1	aF	$S_1T_2E_2$
17	2	q	2	F	$S_1T_2E_2a$
18	1	q	2	$*E$	$S_1T_2E_2aF_1$

Нисходящий распознаватель с возвратом на основе подбора альтернатив

Пример - решение

Итерация	Шаг алгоритма	Q	i	L_1	L_2
19	4	b	2	$*E$	$S_1T_2E_2aF_1$
20	6	q	2	$/E$	$S_1T_2E_2aF_2$
21	4	b	2	$/E$	$S_1T_2E_2aF_2$
22	6	q	2	$*EF$	$S_1T_2E_2aF_3$
23	4	b	2	$*EF$	$S_1T_2E_2aF_3$
24	6	q	2	$/EF$	$S_1T_2E_2aF_4$
25	4	b	2	$/EF$	$S_1T_2E_2aF_4$
26	6	b	2	F	$S_1T_2E_2a$
27	5	b	1	aF	$S_1T_2E_2$
28	6	q	1	bF	$S_1T_2E_3$
29	4	b	1	bF	$S_1T_2E_3$
30	6	b	1	EF	S_1T_2
31	6	b	1	T	S_1
32	6	q	1	TR	S_2
33	1	q	1	ER	S_2T_1
34	1	q	1	$(S)R$	$S_2T_1E_1$
35	4	b	1	$(S)R$	$S_2T_1E_1$
36	6	q	1	aR	$S_2T_1E_2$

Нисходящий распознаватель с возвратом на основе подбора альтернатив

Пример - решение

Итерация	Шаг алгоритма	Q	i	L_1	L_2
37	2	q	2	R	$S_2T_1E_2a$
38	1	q	2	$+T$	$S_2T_1E_2aR1$
39	2	q	3	T	$S_2T_1E_2aR_1+$
40	1	q	3	E	$S_2T_1E_2aR_1+T_1$
41	1	q	3	(S)	$S_2T_1E_2aR_1+T_1E_1$
42	2	q	4	$S)$	$S_2T_1E_2aR_1+T_1E_1($
43	1	q	4	$T)$	$S_2T_1E_2aR_1+T_1E_1(S_1$
44	1	q	4	$E)$	$S_2T_1E_2aR_1+T_1E_1(S_1T_1$
45	1	q	4	$(S))$	$S_2T_1E_2aR_1+T_1E_1(S_1T_1E_1$
46	4	b	4	$(S))$	$S_2T_1E_2aR_1+T_1E_1(S_1T_1E_1$
47	6	q	4	$a)$	$S_2T_1E_2aR_1+T_1E_1(S_1T_1E_2$
48	2	q	5	$)$	$S_2T_1E_2aR_1+T_1E_1(S_1T_1E_2a$
49	4	b	5	$)$	$S_2T_1E_2aR_1+T_1E_1(S_1T_1E_2a$
50	5	b	4	$a)$	$S_2T_1E_2aR_1+T_1E_1(S_1T_1E_2$
51	6	q	4	$b)$	$S_2T_1E_2aR_1+T_1E_1(S_1T_1E_3$
52	4	b	4	$b)$	$S_2T_1E_2aR_1+T_1E_1(S_1T_1E_3$
53	6	b	4	$E)$	$S_2T_1E_2aR_1+T_1E_1(S_1T_1$
54	6	q	4	$EF)$	$S_2T_1E_2aR_1+T_1E_1(S_1T_2$

Нисходящий распознаватель с возвратом на основе подбора альтернатив

Пример - решение

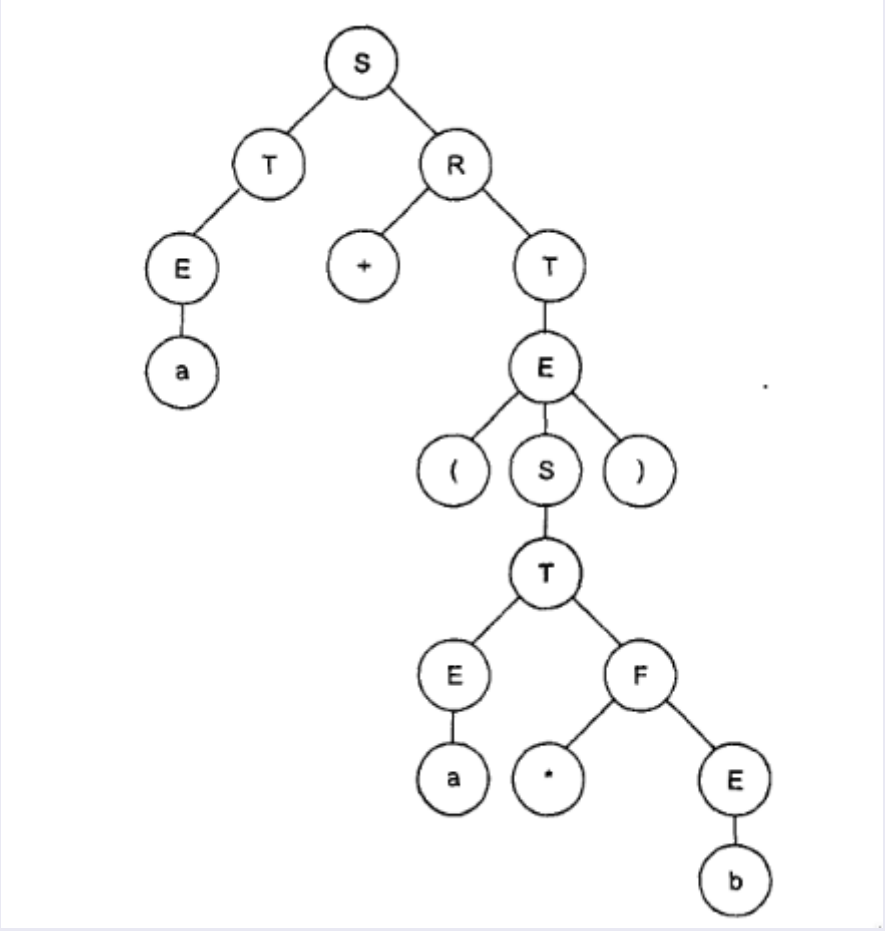
Итерация	Шаг алгоритма	Q	i	L_1	L_2
55	1	q	4	$(S)F)$	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_1$
56	4	b	4	$(S)F)$	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_1$
57	6	q	4	$aF)$	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_2$
58	2	q	5	$F)$	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_2a$
59	1	q	5	$*E)$	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_2aF_1$
60	2	q	6	$E)$	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_2aF_1*$
61	1	q	6	$(S))$	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_2aF_1 * E_1$
62	4	b	6	$(S))$	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_2aF_1 * E_1$
63	6	q	6	$a)$	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_2aF_1 * E_2$
64	4	b	6	$a)$	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_2aF_1 * E_2$
65	6	q	6	$b)$	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_2aF_1 * E_3$
66	2	q	7	$)$	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_2aF_1 * E_3b$
67	2	q	8	ϵ	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_2aF_1 * E_3b)$
68	3	q	9	ϵ	$S_2T_1E_2aR_1 + T_1E_1(S_1T_2E_2aF_1 * E_3b)$

На основе цепочки в стеке L_2 можно получить левосторонний вывод:

$$S \Rightarrow TR \Rightarrow ER \Rightarrow aR \Rightarrow a + T \Rightarrow a + E \Rightarrow a + (S) \Rightarrow a + (T) \Rightarrow$$

$$\Rightarrow a + (EF) \Rightarrow a + (aF) \Rightarrow a + (a * E) \Rightarrow a + (a * b)$$

Дерево вывода для примера



Распознаватель на основе алгоритма «сдвиг-свертка»

Восходящий распознаватель по алгоритму «сдвиг - свертка» строится на основе расширенного МП-автомата, осуществляя правосторонний вывод.
Начальная конфигурация МП-автомата определяется начальным состоянием q, считывающая головка находится в начале входной цепочки, стек пуст.
Конечная конфигурация определяется тем, что считывающая головка находится за концом входной цепочки символов, а в стеке лежит символ, соответствующий целевому символу грамматики S.
Неформально работу этого расширенного автомата можно описать так: если на верхушке стека находится цепочка символов γ, то ее можно заменить на нетерминальный символ , если в грамматике языка существует правило вида $A \to \gamma$, не сдвигая при этом считывающую головку автомата (этот шаг работы называется «свертка»). С другой стороны, если считывающая головка автомата обозревает некоторый символ входной цепочки , то его можно поместить в стек, сдвинув при этом головку на одну позицию вправо (этот шаг работы называется «сдвиг»).
Алгоритм, моделирующий работу такого расширенного МП-автомата, называется алгоритмом «сдвиг-свертка».

Распознаватель на основе алгоритма «сдвиг-свертка»

На каждом шаге работы автомата надо решать следующие вопросы:

1. что необходимо выполнять: сдвиг или свертку;
2. если выполнять свертку, то какую цепочку γ выбрать для поиска правил (цепочка γ должна встречаться в правой части правил грамматики);
3. какое правило выбрать для свертки, если окажется, что существует несколько правил вида $A \to \gamma$ (несколько правил с одинаковой правой частью).

Чтобы промоделировать работу этого расширенного МП-автомата, надо на каждом шаге запоминать все предпринятые действия, чтобы иметь возможность вернуться к уже сделанному шагу и выполнить эти же действия по другому. Этот процесс должен повторяться до тех пор, пока не будут перебраны все возможные варианты.

Реализация алгоритма «сдвиг-свертка»

Алгоритм работает с двумя стеками: L_1 - стек МП-автомата, L_2 - стек возвратов. Состояние алгоритма на каждом шаге описывается четырьмя параметрами: текущее состояние автомата q, b (состояние b обозначает необходимость возврата); i - положение считывающей головки во входной цепочке символов; L_1, L_2. Начальное состояние алгоритма $(q, 1, \epsilon, \epsilon)$. Алгоритм состоит из пяти шагов:

- Шаг 1. (Попытка свертки). $(q, i, \alpha\beta, \gamma) \to (q, i, \alpha A, j\gamma)$, если $A \to beta$ - это j-е правило из множества правил. Если свертка удалась, повторяем шаг 1, иначе переходим к шагу 2.
- Шаг 2. (Сдвиг). Если $i < n + 1$, то $(q, i, \alpha, \gamma) \to (q, i + 1, \alpha a_i, 0\gamma)$. Если $i = n + 1$, то перейти к шагу 3, иначе перейти к шагу 1.
- Шаг 3. (Завершение). Если состояние алгоритма соответствует $(q, n + 1, S, \gamma)$, то работа завершена, иначе перейти к шагу 4.
- Шаг 4. (Переход к возврату). $(q, n + 1, \alpha, \gamma) \to (b, n + 1, \alpha, \gamma)$.
- Шаг 5. (Возврат). Если исходное состояние $(b, i, \alpha A, j\gamma)$, то:
  - Если $A \to \beta$ - это j-е правило $(j > 0)$ и существует k-е правило $k > j$ вида $B \to \beta'$, такое, что $\alpha\beta = \alpha'\beta'$, то перейти в состояние $(q, i, \alpha' B, k\gamma)$ и вернуться к шагу 1.
  - Перейти в состояние $(b, n + 1, \alpha\beta, \gamma)$, если $i = n + 1, j > 0$ и $A \to \beta$ - это j-е правило и не существует других правил с номером $k > j$, таких, что их правая часть является правой подцепочкой из цепочки $\alpha\beta$, после этого вернуться к шагу 5.
  - Перейти в состояние $(q, i + 1, \alpha\beta a_i, 0\gamma)$, a_i - i-й терминальный символ, если $i \neq n + 1, j > 0 A \to \beta$ - это правило с номером j и не существует других правил с номером $k > j$, таких, что их правая часть является правой подцепочкой из цепочки $\alpha\beta$, после этого вернуться к шагу 1.

Реализация алгоритма «сдвиг-свертка»

(продолжение алгоритма)
Если исходное состояние $(b, i, \alpha a, 0\gamma)$, a - терминальный символ, то если $i > 1$, тогда перейти в следующее состояние $(b, i-1, \alpha, \gamma)$ и вернуться к шагу 5; иначе сигнализировать об ошибке и прекратить выпонение алгоритма.
После успешного завершения алгоритма цепочку вывода можно построить на основе стека L_2, удалив в нем все цифры 0.
Важно отметить, что применение данного алгоритма возможно при условии отсутствия в грамматике ϵ-правил, а также циклов. В противном случае прежде необходимо преобразовать грамматику к такому виду.
Рассмотрим пример использования данного алгоритма.
Пусть дана грамматика:

$$S \to S + T|S - T|T * E|T/E|(S)|a|b$$

$$T \to T * E|T/E|(S)|a|b$$

$$E \to (S)|a|b$$

Это грамматика для арифметических выражений без цепных правил. Рассмотрим разбор цепочки $a + (a * b)$ в этой грамматике на основе алгоритма сдвина-свертки.

Восходящий распознаватель с возвратом на основе алгоритма «сдвиг-свертка»

Пример - решение

Итерация	Шаг алгоритма	Q	i	L_1	L_2
1	0	q	1	ϵ	ϵ
2	2	q	2	a	0
3	1	q	2	S	$6,0$
4	2	q	3	$S+$	$0,6,0$
5	2	q	4	$S+($	$0,0,6,0$
6	2	q	5	$S+(a$	$0,0,0,6,0$
7	1	q	5	$S+(S$	$6,0,0,0,6,0$
8	2	q	6	$S+(S*$	$0,6,0,0,0,6,0$
9	2	q	7	$S+(S*b$	$0,0,6,0,0,0,6,0$
10	1	q	7	$S+(S*S$	$7,0,0,6,0,0,0,6,0$
11	2	q	8	$S+(S*S)$	$0,7,0,0,6,0,0,0,6,0$
12	4	b	8	$S+(S*S)$	$0,7,0,0,6,0,0,0,6,0$
13	5	b	7	$S+(S*S$	$7,0,0,6,0,0,0,6,0$
14	5	q	7	$S+(S*T$	$12,0,0,6,0,0,0,6,0$
15	2	q	8	$S+(S*T)$	$0,12,0,0,6,0,0,0,6,0$
16	4	b	8	$S+(S*T)$	$0,12,0,0,6,0,0,0,6,0$
17	5	b	7	$S+(S*T$	$12,0,0,6,0,0,0,6,0$

Восходящий распознаватель с возвратом на основе алгоритма «сдвиг-свертка»

Пример - решение

Итерация	Шаг алгоритма	Q	i	L_1	L_2
18	5	q	7	$S+(S*E$	$15,0,0,6,0,0,0,6,0$
19	2	q	8	$S+(S*E)$	$0,15,0,0,6,0,0,0,6,0$
20	4	b	8	$S+(S*E)$	$0,15,0,0,6,0,0,0,6,0$
21	5	b	7	$S+(S*E$	$15,0,0,6,0,0,0,6,0$
22	5	q	8	$S+(S*b)$	$0,0,0,6,0,0,0,6,0$
23	4	b	8	$S+(S*b)$	$0,0,0,6,0,0,0,6,0$
24	5	b	7	$S+(S*b$	$0,0,6,0,0,0,6,0$
25	5	b	6	$S+(S*$	$0,6,0,0,0,6,0$
26	5	b	5	$S+(S$	$6,0,0,0,6,0$
27	5	q	5	$S+(T$	$11,0,0,0,6,0$
28	2	q	6	$S+(T*$	$0,11,0,0,0,6,0$
29	2	q	7	$S+(T*b$	$0,0,11,0,0,0,6,0$
30	1	q	7	$S+(T*S$	$7,0,0,11,0,0,0,6,0$
31	2	q	8	$S+(T*S)$	$0,7,0,0,11,0,0,0,6,0$
32	4	b	8	$S+(T*S)$	$0,7,0,0,11,0,0,0,6,0$
33	5	b	7	$S+(T*S$	$7,0,0,11,0,0,0,6,0$
34	5	q	7	$S+(T*T$	$12,0,0,11,0,0,0,6,0$

Восходящий распознаватель с возвратом на основе алгоритма «сдвиг-свертка»

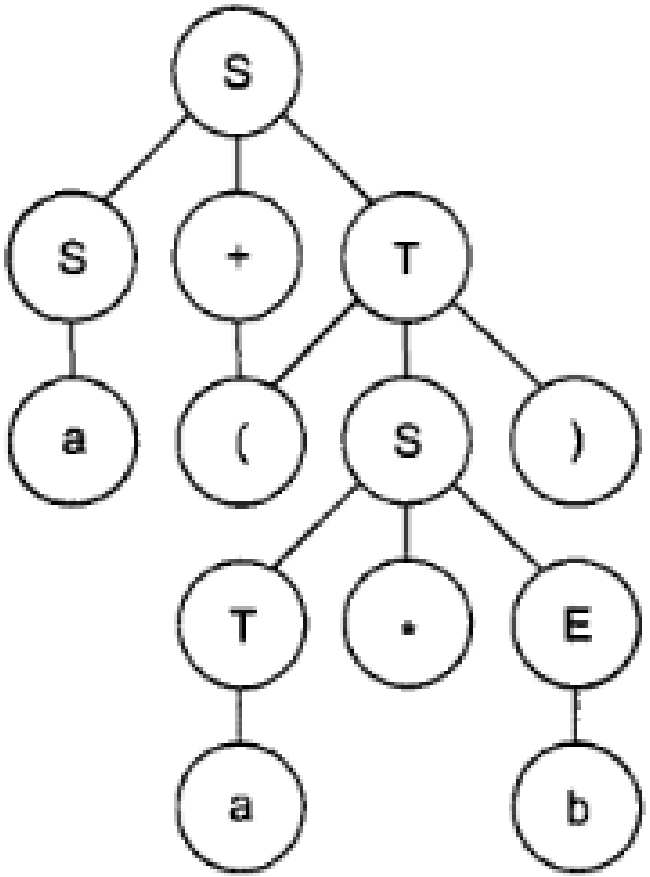
Пример - решение

Итерация	Шаг алгоритма	Q	i	L_1	L_2
35	2	q	8	$S+(T*T)$	$0,12,0,0,11,0,0,0,6,0$
36	4	b	8	$S+(T*T)$	$0,12,0,0,11,0,0,0,6,0$
37	5	b	7	$S+(T*T$	$12,0,0,11,0,0,0,6,0$
38	5	q	7	$S+(T*E$	$15,0,0,11,0,0,0,6,0$
39	1	q	7	$S+(S$	$3,15,0,0,11,0,0,0,6,0$
40	2	q	8	$S+(S)$	$0,3,15,0,0,11,0,0,0,6,0$
41	1	q	8	$S+S$	$5,0,3,15,0,0,11,0,0,0,6,0$
42	4	b	8	$S+S$	$5,0,3,15,0,0,11,0,0,0,6,0$
43	5	q	8	$S+T$	$10,0,3,15,0,0,11,0,0,0,6,0$
44	1	q	8	S	$1,10,0,3,15,0,0,11,0,0,0,6,0$

На основании полученной цепочки номеров $1,10,3,15,11,6$, получаем правосторонний вывод:

$$S \to S+T \to S+(S) \to S+(T*E) \to S+(T*b) \to S+(a*b) \to a+(a*b)$$

Дерево вывода для примера



Задание 1

В грамматике правильных скобочных выражений: $S \to (S)S|()|()S$ построить вывод цепочки

$$()(())$$

с помощью нисходящего и восходящего алгоритмов с возвратами.