Теоретическая информатика, осень 2020 г. Лекция 7. Синтаксический анализ за кубическое время, построение дерева разбора. Параллельный разбор за время $(\log n)^2$.[*]

Александр Охотин

22 октября 2020 г.

Содержание

1 Синтаксический анализ за кубическое время

1.1 Алгоритм Кокка–Касами–Янгера

Синтаксический анализ: для фиксированной грамматики $G = (\Sigma, N, R, S)$ и для данной входной строки $w \in \Sigma^*$, проверить, принадлежит ли w языку $L(G)$ — то есть, синтаксически правильна ли она согласно грамматике G. Если принадлежит, то построить дерево разбора.

Рис. 1: Джон Кокк (1925–2002), Тадао Касами (1930–2007), Дэниел Янгер.

[*]Краткое содержание лекций, прочитанных студентам 2-го курса факультета МКН СПбГУ в осеннем семестре 2020–2021 учебного года. Страница курса: `http://users.math-cs.spbu.ru/~okhotin/teaching/tcs_fl_2020/`.

Теорема 1. *Для всякой грамматики $G = (\Sigma, N, R, S)$ есть алгоритм синтаксического анализа, работающий за время $O(n^3)$, где n — длина входной строки.*

Грамматика $G = (\Sigma, N, R, S)$ в н.в.Хомского, входная строка $w = a_1 \ldots a_n$. Алгоритм основан на методе динамического программирования. Цель: определить для каждой подстроки $a_{i+1} \ldots a_j$ и для каждого нетерминального символа $A \in N$, имеет ли эта подстрока свойство A. Эти сведения записываются в *таблицу разбора* $T \in (2^N)^{n \times n}$, в которой всякий элемент $T_{i,j}$, где $0 \leqslant i < j \leqslant n$, содержит множество нетерминальных символов, задающих подстроку между позициями $i+1$ и j.

$$T_{i,j} = \{\, A \in N \mid a_{i+1} \ldots a_j \in L_G(A) \,\}$$

Элементы таблицы заполняются последовательно, от меньших подстрок к большим. Сперва вычисляются свойства всех подстрок длины 1. Чтобы односимвольная подстрока a_i обладала свойством A, необходимо, чтобы в грамматике было правило $A \to a_i$.

$$T_{i-1,i} = \{\, A \mid A \to a_i \in R \,\}$$

Подстрока $a_{i+1} \ldots a_j$ длины 2 и более может обладать свойством A, если она определяется некоторым правилом $A \to BC$. Тогда подстрока разбивается на две непустых подстроки, $a_{i+1} \ldots a_k$ и $a_{k+1} \ldots a_j$, причём первая должна обладать свойством B, а вторая — свойством C. Точка разбиения k может быть любым числом от $i+1$ до $j-1$. Так как обе эти подстроки *короче*, чем вся подстрока, то все сведения об этих более коротких строках к этому моменту уже вычислены и занесены в ячейки таблицы $T_{i,k}$ и $T_{k,j}$. Отсюда можно вычислить множество всех нетерминальных символов для подстроки $a_{i+1} \ldots a_j$.

$$T_{i,j} = \{\, A \mid \exists A \to BC \in R,\ \exists k \in \{i+1, \ldots, j-1\} :\ B \in T_{i,k} \land C \in T_{k,j} \,\}$$

Алгоритм 1 Алгоритм Кокка–Касами–Янгера.

Грамматика $G = (\Sigma, N, R, S)$ в н.в.Хомского, входная строка $w = a_1 \ldots a_n$, где $n \geqslant 1$ и $a_i \in \Sigma$. Для всех $0 \leqslant i < j \leqslant n$, пусть $T_{i,j}$ — переменная, принимающая значение подмножества N.

for $i = 1$ to n **do**
    $T_{i-1,i} = \{\, A \mid A \to a_i \in R \,\}$
for $\ell = 2$ to n **do**
    for $i = 0$ to $n - \ell$ **do**
        пусть $j = i + \ell$
        $T_{i,j} = \varnothing$
        for all $A \to BC \in R$ **do**
            for $k = i+1$ to $j-1$ **do**
                if $B \in T_{i,k} \land C \in T_{k,j}$ **then**
                    $T_{i,j} = T_{i,j} \cup \{A\}$
принять тогда и только тогда, когда $S \in T_{0,n}$

Пример 1. *Следующая грамматика в н.в.Хомского задаёт язык Дика без пустой строки.*

$$\begin{aligned}
&S \to SS \mid AC \\
&A \to a \\
&B \to b \\
&C \to SB \mid b
\end{aligned}$$

Таблица разбора для строки $w = abaabb$ такова.

	0	1	2	3	4	5	6
0		$\{A\}$	$\{S\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{S\}$
1			$\{B, C\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
2				$\{A\}$	$\varnothing$	$\varnothing$	$\{S\}$
3					$\{A\}$	$\{S\}$	$\{C\}$
4						$\{B, C\}$	$\varnothing$
5							$\{B, C\}$
6							

Поскольку $S \in T_{0,6}$, строка принадлежит языку.

1.2 Построение дерева разбора

Пусть для грамматики $G = (\Sigma, N, R, S)$ в н.в.Хомского и для входной строки $w = a_1 \dots a_n$ построена таблица разбора.

$$T_{i,j} = \{\, A \in N \mid a_{i+1} \dots a_j \in L_G(A) \,\}$$

Пусть оказалось, что $S \in T_{0,n}$, и, стало быть, $w \in L(G)$. Тогда, используя таблицу, алгоритм 2 строит дерево разбора.

Алгоритм 2 Построение дерева разбора по таблице

Пусть $G = (\Sigma,\ N,\ R,\ S)$ — грамматика в н.в.Хомского, пусть $w = a_1 \dots a_n$, где $n \geqslant 1$ и $a_i \in \Sigma$ — входная строка, и пусть множества $T_{i,j} = \{\, A \mid a_{i+1} \dots a_j \in L_G(A) \,\}$ известны для всех $0 \leqslant i < j \leqslant n$.
Дано: $A \in N$, позиции ℓ и m, где $0 \leqslant \ell < m \leqslant n$, причём верно $A \in T_{\ell,m}$.
Построить: дерево разбора подстроки $a_{\ell+1} \dots a_m$ по A.

процедура $parse(A, \ell, m)$, предусловие: $A \in T_{\ell,m}$

if $m - \ell = 1$ **then**
    return дерево с корнем A, к которому подсоединён лист a_m
else
    for all $A \to BC \in R$ **do**
        for $k = \ell + 1$ to $m - 1$ **do**
            if $B \in T_{\ell,k}$ и $C \in T_{k,m}$ **then**
                Создать вершину τ, помеченную A
                Добавить потомка $parse(B, \ell, k)$ к τ
                Добавить потомка $parse(C, k, m)$ к τ
                return τ

Время работы пропорционально $n \cdot t$, где t — число вершин в дереве разбора, поскольку

Рис. 2: Лесли Валиант (род. 1949).

при построении каждой вершины выполняется цикл в строке 5. Так как $t = \Theta(n)$, алгоритм работает за время $\Theta(n^2)$, что меньше, чем время построения таблицы.

1.3 Более быстрый алгоритм

Известен *алгоритм Валианта* [1975] — алгоритм синтаксического анализа, работающий за время $O(n^\omega)$, где n^ω — время умножения матриц размера $n \times n$. Алгоритм основан на методе «разделяй и властвуй» и вычисляет ту же самую таблицу $T_{i,j}$, что и в алгоритме Кокка–Касами–Янгера.

2 Параллельный разбор за время $(\log n)^2$

2.1 Высота дерева разбора

Высота дерева разбора — то есть, длина самого длинного пути в нём — это глубина логических зависимостей в определении строки. Своего рода мера сложности грамматики. Связана с объёмом памяти, требуемым для синтаксического анализа, а также со временем работы параллельных алгоритмов анализа.

Наименьшая возможная высота — логарифмическая от длины строки. Пример: грамматика $S \to SS \mid aSb \mid \varepsilon$ и хорошо подобранная последовательность скобок, такая как $aaababbaababbb$.

Но чаще высота дерева будет линейной (более чем линейной она быть не может).

Пример 2. *Грамматика для языка* $\{ a^n b^n \mid n \geqslant 0 \}$.

$$S \to aSb \mid \varepsilon$$

Дерево разбора строки $w = a^8 b^8$ *дано на рис. 3.*

Упражнение 1. *Доказать, что всякая грамматика, задающая язык* $\{ a^n b^n \mid n \geqslant 0 \}$, *определяет деревья разбора линейной высоты.*

План: найти промежуточную вершину; отдельно вывести (а) это поддерево и (б) всё кроме этого поддерева; объединить их.

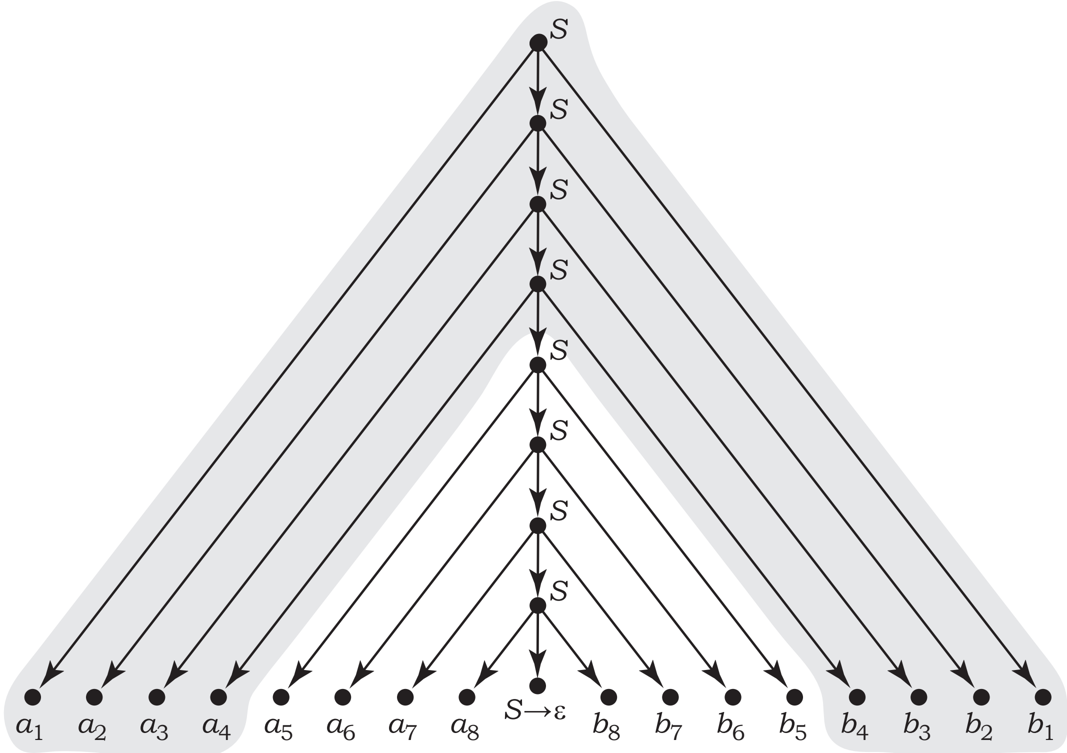

Рис. 3: Дерево разбора строки $a^8 b^8$ по грамматике из примера 2: символы помечены своими номерами, $a_1 \ldots a_8 b_8 \ldots b_1$; серая область — поддерево с дыркой $\frac{S}{S}(a_1 \ldots a_4 : b_4 \ldots b_1)$.

Лемма 1. *Пусть $G = (\Sigma, N, R, S)$ — грамматика в н.в.Хомского. Тогда всякое дерево разбора с n листьями содержит вершину, в чьём поддереве более чем $\frac{1}{3}n$ листьев, и самое большее $\frac{2}{3}n$.*

Доказательство. Путь строится сверху, каждый раз выбирается большее поддерево. Поскольку ветвление двоичное, искомая вершина рано или поздно найдётся. ☐

2.2 Как обеспечить малую высоту зависимостей?

Идея: чтобы доказать утверждение φ, независимо доказать некоторое утверждение ξ и *условное утверждение* вида «если ξ, то φ», и затем вывести из них φ. Если утверждение ξ выбрано хорошо, глубина доказательства ополовинится.

В определении грамматик через логический вывод все утверждения имеют вид $A(u)$, где A — нетерминальный символ (т.е., свойство подстрок), а u — подстрока. Условные утверждения «если ξ, то φ» тогда принимают вид «если $D(x)$, то $A(uxv)$», где $A, D \in N$, $u, v \in \Sigma^*$ а x — переменная, обозначающая некоторую подстроку. Точнее сказать, x означает *дырку*, оставленную в строке со свойством A для более короткой строки со свойством D. Обозначение: $\frac{A}{D}(u : v)$.

Утверждение $\frac{A}{D}(u : v)$, где $A, D \in N$ и $u, v \in \Sigma^*$, означает, что есть дерево разбора с корнем $A \in N$ и с дыркой в виде D без потомков, содержащее $|u| + |v|$ листьев, образующих строку u слева от D и строку v справа от D.

Пример 3. *Для грамматики в примере 2, все условные утверждения имеют вид* $\frac{S}{S}(u,v)$*, и используются следующие правила вывода.*

$$\frac{}{\frac{S}{S}(a:b)} \qquad \text{(создание дырки)}$$

$$\frac{\frac{S}{S}(u:v),\, S(w)}{S(uwv)} \qquad \text{(затыкание дырки)}$$

$$\frac{\frac{S}{S}(u:v),\, \frac{S}{S}(x:y)}{\frac{S}{S}(ux:yv)} \qquad \text{(соединение условных утверждений)}$$

На рис. 4 показано доказательство малой высоты для утверждения $S(aaaaaaaabbbbbbbb)$*. Последний шаг доказательства таков.*

$$\frac{S}{S}(a^4:b^4), S(a^4b^4) \vdash S(a^8b^8)$$

На рис. 3 этот шаг показан в виде соединения дерева с дыркой $\frac{S}{S}(a^4:b^4)$ *и стандартного дерева* $S(a^4b^4)$*.*

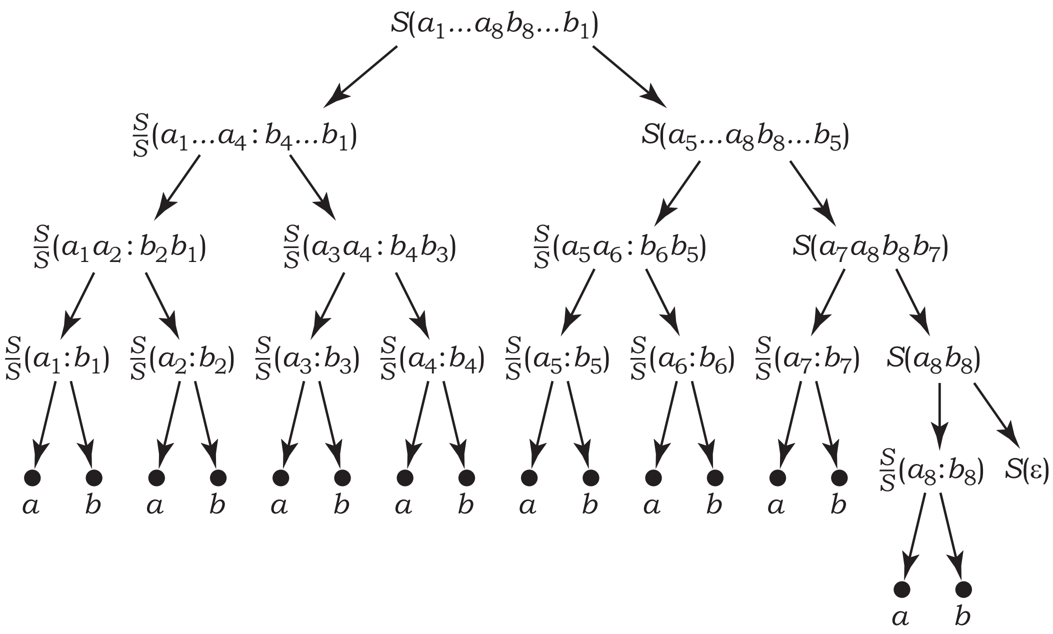

Рис. 4: Доказательство $S(a^8b^8)$ малой высоты по грамматике из примера 2; символы помечены номерами: $a_1 \dots a_8 b_8 \dots b_1$.

В общем случае, пусть грамматика — в н.в.Хомского. Тогда используется система вывода

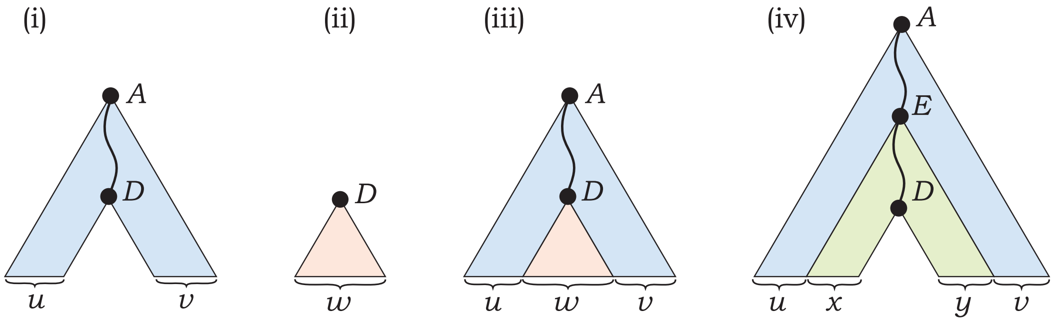

Рис. 5: (i) Дерево с дыркой, соответствующее условному утверждению $\frac{A}{D}(u:v)$; (ii) Полное дерево, соответствующее обычному утверждению $D(w)$; (iii) Полное дерево разбора строки uwv из D, полученное соединением этих двух утверждений; (iv) Соединение двух условных утверждений.

со следующими правилами.

$$\frac{}{A(a)} \qquad \text{(одиночный символ, если } A \to a \in R\text{)}$$

$$\frac{B(u)}{\frac{A}{C}(u:\varepsilon)} \qquad \text{(создание дырки справа, если } A \to BC \in R\text{)}$$

$$\frac{C(v)}{\frac{A}{B}(\varepsilon:v)} \qquad \text{(создание дырки слева, если } A \to BC \in R\text{)}$$

$$\frac{\frac{A}{D}(u:v),\, D(w)}{A(uwv)} \qquad \text{(затыкание дырки)}$$

$$\frac{\frac{A}{E}(u:v),\, \frac{E}{D}(x:y)}{\frac{A}{D}(ux:yv)} \qquad \text{(соединение условных утверждений)}$$

Лемма 2. *Всё, что доказывается в расширенной системе, доказывается и в стандартной, с использованием лишь утверждений вида $A(w)$.*

Действительно, это же обычные деревья разбора, просто строящиеся по кускам! Строгое доказательство — индукцией по длине доказательства в расширенной системе.

Лемма 3. *В расширенной системе всякое утверждение $\frac{A}{D}(u:v)$ или $A(w)$ имеет доказательство высоты не более чем $4\log_{3/2} n$, где n — число листьев, т.е., $n = |uv|$ или $n = |w|$.*

Доказательство. Индукция по n.

Для дерева разбора $A(w)$ по лемме 1 находится промежуточная вершина, и записывается вывод по правилу затыкания дырки, из двух посылок содержащих не более $\frac{2}{3}n$ листьев каждая — и по предположению индукции у них есть доказательства логарифмической высоты.

Для условного утверждения $\frac{A}{D}(u:v)$ промежуточная вершина в дереве с дыркой находится точно так же, однако, в зависимости от её положения в дереве, нужно рассмотреть несколько случаев. Если поддерево промежуточной вершины содержит дырку — то есть,

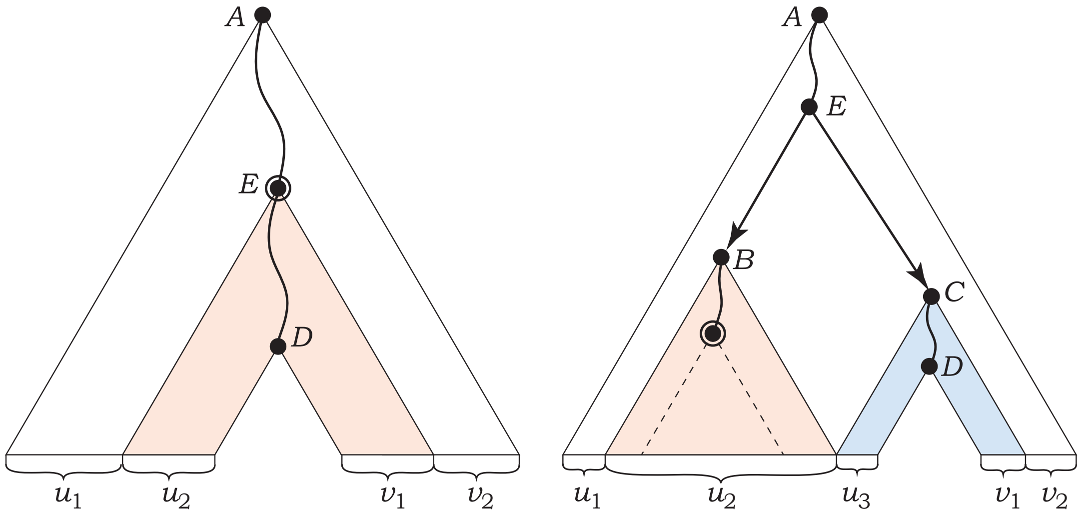

Рис. 6: Два случая разбиения доказательства условного утверждения, в зависимости от положения промежуточной вершины.

включает в себя некоторый суффикс u и некоторый префикс v — то пусть $u = u_1u_2$ и $v = v_1v_2$, и пусть промежуточная вершина помечена нетерминальным символом $E \in N$. Тогда $\frac{A}{D}(u:v)$ выводится из $\frac{A}{E}(u_1:v_2)$ и $\frac{E}{D}(u_2:v_1)$ по правилу соединения условных утверждений.

Пусть поддерево промежуточной вершины содержит только листья из u. Тогда рассматривается наибольшее поддерево, включающее в себя данное поддерево и содержащее только листья из u. Пусть $u = u_1u_2u_3$, где u_2 — листья этого большего поддерева, пусть $B \in N$ — нетерминальный символ в корне поддерева, пусть $E \in N$ — нетерминальный символ в его родителе, и пусть $C \in N$ — второй потомок E (используется правило $E \to BC$). Поскольку в поддереве C есть листья из v, пусть $v = v_1v_2$, где v_1 — листья в поддереве C. Тогда $\frac{A}{D}(u:v)$ выводится так.

$$\dfrac{\frac{A}{E}(u_1:v_2) \qquad \dfrac{\dfrac{B(u_2)}{\frac{E}{C}(u_2:\varepsilon)} \qquad \frac{C}{D}(u_3:v_1)}{\frac{E}{D}(u_2u_3:v_1)}}{\frac{A}{D}(u:v)}$$

Поскольку u_2 содержит не менее чем $\frac{1}{3}n$ листьев, два других условных утверждения содержат не более чем $\frac{2}{3}n$ листьев каждое, и потому, по предположению индукции, оба имеют доказательства небольшой высоты. Для дерева без дырки $B(u_2)$ уже установлено существование доказательства небольшой высоты, в котором дерево u_2 из B на первом шаге разбивается на поддерево с дыркой и обычное поддерево с примерно равным числом листьев. Стало быть, искомое поддерево с n листьями выводится из поддеревьев не более чем с $\frac{2}{3}n$ листьями в доказательстве высоты не более 4 — отсюда константа 4 в формулировке леммы. $\square$

2.3 Разбор с использованием памяти $(\log n)^2$

Теорема 2. *Алгоритм 3 определяет принадлежность строки длины n языку $L(G)$ за время $n^{O(\log n)}$, используя $O((\log n)^2)$ битов памяти.*

Алгоритм 3 Алгоритм Льюиса–Стирнса–Хартманиса

Пусть $G = (\Sigma, N, R, S)$ — грамматика в н.в.Хомского. Пусть $w = a_1 \dots a_n$, где $n \geqslant 1$ и $a_i \in \Sigma$, — входная строка.

- Процедура $A(i, j; d)$, для $A \in N$, определяет, есть ли доказательство $A(a_{i+1} \dots a_j)$ высоты не более чем d в расширенной системе.
- Процедура $\frac{A}{D}(i, k, \ell, j; d)$, для $A, D \in N$, определяет, есть ли доказательство $\frac{A}{D}(a_{i+1} \dots a_k : a_{\ell+1} \dots a_j)$ высоты не более чем d в расширенной системе.

Затем достаточно вызвать $S(0, n; 4 \log_{3/2} n)$.

процедура $A(i, j; d)$

if $d = 0$ **then**
  return false
if $i + 1 = j \wedge A \to a_j \in R$ **then** /* одиночный символ */
  return true
for all k, ℓ: $i \leqslant k \leqslant \ell \leqslant j$ **do**
  for all $D \in N$ **do**
    if $\frac{A}{D}(i, k, \ell, j; d-1) \wedge D(k, \ell; d-1)$ **then** /* затыкание дырки */
      return true
return false

процедура $\frac{A}{D}(i, k, \ell, j; d)$

if $d = 0$ **then**
  return false
if $\ell = j$ **then**
  for all $A \to BD \in R$ **do**
    if $B(i, k; d-1)$ **then** /* создание дырки справа */
      return true
if $i = k$ **then**
  for all $A \to DC \in R$ **do**
    if $C(\ell, j; d-1)$ **then** /* создание дырки слева */
      return true
for all s, t: $i \leqslant s \leqslant k$, $\ell \leqslant t \leqslant j$, $(s, t) \neq (i, j)$, $(s, t) \neq (k, \ell)$ **do**
  for all $E \in N$ **do**
    if $\frac{A}{E}(i, s, t, j; d-1) \wedge \frac{E}{D}(s, k, \ell, t; d-1)$ **then** /* соединение */
      return true
return false

 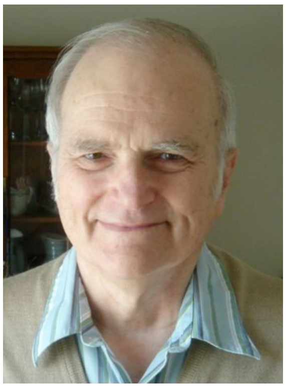 

Рис. 7: Филипп Льюис (род. 1931), Ричард Стинрс (род. 1936), Юрис Хартманис (род. 1928).

Доказательство. Глубина рекурсии — не более чем $4\log_{3/2} n$, поскольку она явна задана параметром. На каждом уровне в стеке размещено $O(\log n)$ битов — отсюда верхняя оценка объёма используемой памяти $O((\log n)^2)$. На каждом уровне делается не более чем n^2 рекурсивных вызовов — и отсюда верхняя оценка времени работы $n^{2\cdot 4\log_{3/2} n}$. □

2.4 Параллельный разбор за время $(\log n)^2$

Алгоритм Брента–Гольдшлягера–Риттера.

 

Рис. 8: Ричард Брент (род. 1946), Войцех Риттер (род. 1948).

Теорема 3 (Руззо [1980], Брент и Гольдшлягер [1984]; Риттер [1985])**.** *Для всякой грамматики $G = (\Sigma, N, R, S)$ в н.в.Хомского и для всякой длины строк $n \geqslant 1$, существует булева схема глубины $O(\log^2 n)$, имеющая $|\Sigma| \cdot n$ входов, через которые вводится строка $w = a_1 \ldots a_n$, $O(n^6 \log n)$ промежуточных булевых элементов, а также один выходной элемент, сообщающий принадлежность w языку $L(G)$.*

Доказательство. Схема содержит следующие элементы.

- Для всех i, j, где $0 \leqslant i < j \leqslant n$, есть элемент $x_{A,i,j}$, в котором вычисляется значение $A(a_{i+1} \ldots a_j)$, то есть, принадлежит ли строка $a_{i+1} \ldots a_j$ языку $L_G(A)$.

- Элемент $y_{A,i,j,D,k,\ell}$, где $A, D \in N$, $0 \leqslant i \leqslant k < \ell \leqslant j \leqslant n$ и $(k - i) + (j - \ell) \geqslant 0$. Этот элемент определяет, существует ли дерево разбора $a_{i+1} \ldots a_j$ из A, с дыркой вместо поддерева $a_{k+1} \dots a_\ell$ из D, так что в нём вычисляется значение 1 тогда и только тогда, когда верно условное утверждение $\frac{A}{D}(a_{i+1} \dots a_k : a_{\ell+1} \dots a_j)$.

Всего таких элементов $\Theta(n^4)$, и каждый из них соответствует запуску одной из процедур алгоритма 3 с некоторыми значениями аргументов. В схеме значение этого элемента вычисляется по тем же формулам, что и в соответствующих процедурах. $\square$

3 Неразрешимые задачи для грамматик

Кодирование неразрешимых задач в грамматиках: метод *историй вычисления машины Тьюринга*, открытый Хартманисом [1967].

История вычисления — все конфигурации, записанные одна за другой. Если машина останавливается, это конечная строка. Множество всех историй принимающих вычислений: $\mathrm{VALC}(M)$, где M — машина Тьюринга.

Пусть $M = (\Sigma, \Gamma, Q, q_0, \delta, q_{acc})$ — детерминированная МТ, где $\Gamma \supset \Sigma$ — её рабочий алфавит, содержащий символ пробела $\llcorner\!\lrcorner \in \Gamma \setminus \Sigma$, и Q — множество состояний, не пересекающееся с Γ. Пусть M работает на односторонней бесконечной ленте и никогда не пытается заехать левее самого левого входного символа. Начальное состояние — $q_0 \in Q$. Машина принимает, переходя в состояние q_{acc}, и отвергает, зацикливаясь.

Пусть $\Omega = \Gamma \cup Q \cup \{\#, \$\}$ — алфавит, используемый для представления историй вычисления. Когда M, находясь в состоянии $q \in Q$, видит символ $a \in \Gamma$, и на ленте слева от головки лежит строка $u \in \Gamma^*$, а справа — строка $v \in \Gamma^*$ (не считая пробелов, которые ещё не посещались), то такая конфигурация обозначается строкой $uqav \in \Gamma^* Q \Gamma^+$. Для всякой входной строки $w \in \Sigma^*$, конфигурация МТ после i шагов вычисления обозначается так.

$$C_i = C_i(M, w) = uqav$$

Если M останавливается на w после n шагов, то история её вычисления имеет следующий вид.

$$C_M(w) = C_0 \# C_1 \# C_2 \# \dots \# C_{n-1} \# C_n \$ C_n^R \# C_{n-1}^R \# \dots \# C_2^R \# C_1^R \# C_0^R$$

Наконец, определяется язык историй вычисления машины Тьюринга M.

$$\mathrm{VALC}(M) = \{\, C_M(w) \mid w \in L(M) \,\}$$

Лемма 4. *Для всякой машины Тьюринга M существуют и могут быть эффективно построены такие грамматики G_1 и G_2, что $L(G_1) \cap L(G_2) = \mathrm{VALC}(M)$. Кроме того, существуют грамматики G_1' и G_2', задающие дополнения этих языков: $L(G_i') = \overline{L(G_i)}$ для $i \in \{1, 2\}$ — и потому есть и грамматика, задающая язык $\overline{\mathrm{VALC}(M)}$.*

Доказательство. Общий вид историй вычисления и способ их проверки с помощью пересечения двух языков изображены на рис. 9. Каждая i-я конфигурация встречается в истории вычисления дважды: в своём первоначальном виде C_i в левой части строки, и в перевёрнутом виде C_i^R в правой части. Грамматика G_1 сравнивает каждую i-ю конфигурацию C_i с перевёрнутой следующей конфигурацией C_{i+1}^R, проверяя правильность одного шага машины Тьюринга. Кроме того, G_1 проверяет, что конфигурация C_n — принимающая, а C_0^R — это обращение начальной конфигурации на какой-то строке.

Грамматика G_2 всего лишь задаёт палиндромы со знаком доллара посередине.

Построение грамматики G_1 требует рассмотреть разные виды переходов машины Тьюринга, и потому приведённая ниже грамматика относительно велика; но ничего особенно сложного в ней нет.

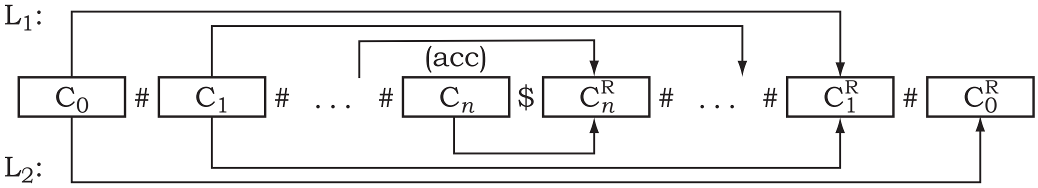

Рис. 9: Представление $\mathrm{VALC}(M)$ в виде $L(G_1) \cap L(G_2)$: общая структура сравнений, задаваемых двумя грамматиками.

Грамматика G_1 сперва задаёт вид конфигурации C_0^R — это может быть любая начальная конфигурация машины Тьюринга, записанная в конце строки в обратном порядке.

$$\begin{aligned} S_1 &\to Baq_0 && (a \in \Sigma) \\ B &\to Ba && (a \in \Sigma) \\ B &\to A\# \end{aligned}$$

Случай пустой строки ($w = \varepsilon$) рассматривается отдельно.

$$S_1 \to A\#\llcorner q_0$$

Далее, нетерминальный символ A сравнивает каждую конфигурацию C_i слева с перевёрнутой следующей конфигурацией C_{i+1}^R справа, чтобы убедиться, что это действительно последовательные конфигурации машины M.

Конфигурация C_i имеет общий вид $u\alpha v$, где α — несколько символов вокруг головки, затрагиваемые переходом, а $u, v \in \Gamma^*$ — все остальные символы на ленте, остающиеся неизменными на этом шаге. Тогда следующая конфигурация имеет вид $C_{i+1} = u\beta v$, или $C_{i+1}^R = v^R\beta^R u^R$ в перевёрнутом виде. Правила грамматики для A сперва сопоставляют друг другу символы начального куска ленты u, с которых начинается C_i и которыми заканчивается C_{i+1}^R. Это делается следующими правилами.

$$A \to cAc \qquad (c \in \Gamma)$$

Далее наступает очередь перехода машины Тьюринга на данном шаге. Пусть $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$ — функция переходов M, где $\delta(q, a)$ определяет действие M в состоянии $q \in Q$ при виде символа $a \in \Gamma$: машина переходит в указанное новое состояние, заменяет a на указанный новый символ и перемещает головку в указанном направлении.

Пусть переход в i-конфигурации перемещает головку налево, то есть, $C_i = ubqav$ и $\delta(q, a) = (q', a', -1)$. Тогда вслед за конфигурацией $C_i = ubqav$ идёт конфигурация $C_{i+1} = uq'ba'v$, или $C_{i+1}^R = v^R a' b q' u^R$ в перевёрнутом виде. Соответственно, дойдя до подстроки bqa слева, грамматика требует, чтобы справа была подстрока $a'bq'$, как показано на рис. 10(левом). Это делается правилами следующего вида.

$$A \to bqaAa'bq' \qquad (\delta(q, a) = (q', a', -1),\ b \in \Gamma)$$

Если же машина перемещает головку направо и $\delta(q, a) = (q', a', +1)$, то конфигурация вида $C_i = uqacv$ сменяется конфигурацией $C_{i+1} = ua'q'cv$, записанной в виде $C_{i+1}^R = v^R c q' a' u^R$. Тогда грамматика сопоставляет символам aq слева символы $a'q'$ справа. Этот случай показан на рис. 10(правом).

$$A \to qaAq'a' \qquad (\delta(q, a) = (q', a', +1))$$

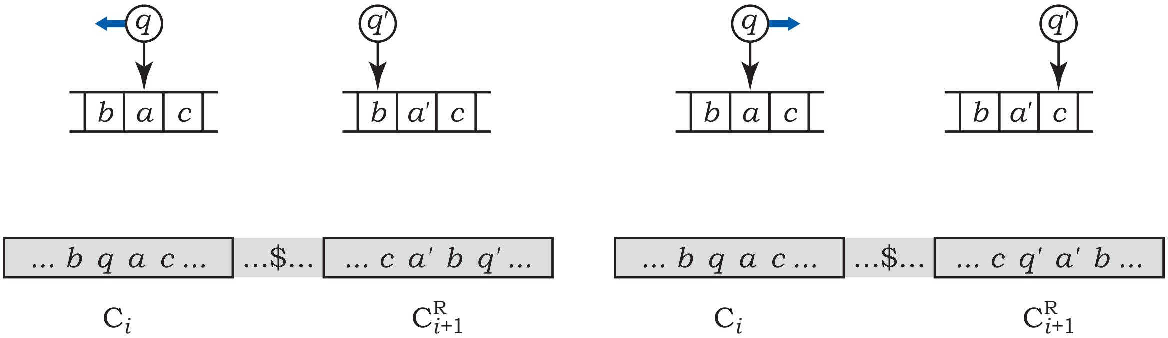

Рис. 10: Представление $\mathrm{VALC}(M) = \{\, C_M(w) \mid w \in L(M)\,\}$ в виде $L(G_1) \cap L(G_2)$: как первая грамматика обеспечивает, что для всякой конфигурации C_i, на другой стороне находится обращение следующей конфигурации C_{i+1}.

Остальные символы из C_i сравниваются с соответствующими символами из C_{i+1}^R с помощью правила $A \to cAc$. Когда все эти символы пройдены, грамматика переходит или к сравнению следующей пары конфигураций, или к проверке, что осталась одна принимающая конфигурация.

$$\begin{aligned} A &\to \#A\# \\ A &\to \#C\$ \end{aligned}$$

Также нужно рассмотреть особый случай перехода от одной конфигурации к другой, когда машина Тьюринга перемещается направо от самого правого символа ленты. В этом случае $C_i = uq$, и отсутствующий символ под q считается пробелом.

$$\begin{aligned} A &\to bq\#A\#a'bq' && (\delta(q, \textvisiblespace) = (q', a', -1),\ b \in \Gamma) \\ A &\to bq\#C\$a'bq' && (\delta(q, \textvisiblespace) = (q', a', -1),\ b \in \Gamma) \\ A &\to q\#A\#q'a' && (\delta(q, \textvisiblespace) = (q', a', +1)) \\ A &\to q\#C\$q'a' && (\delta(q, \textvisiblespace) = (q', a', +1)) \end{aligned}$$

Наконец, грамматика G_1 должна задать условие, что последняя конфигурация C_n^R — принимающая. Это делается нетерминальным символом C.

$$\begin{aligned} C &\to cC && (c \in \Gamma) \\ C &\to q_{\mathrm{acc}}C \\ C &\to \varepsilon \end{aligned}$$

Вторая грамматика G_2 просто проверяет, что два экземпляра каждой конфигурации соответствуют друг другу — то есть, одна является обращением другой. Для этого достаточно проверить, что строка — палиндром.

$$\begin{aligned} S_2 &\to sS_2s && (s \in \Omega) \\ S_2 &\to \$ \end{aligned}$$

Грамматики G_1' и G_2', задающие дополнения этих двух языков, строятся по одному и тому же принципу, причём для G_2' (не-палиндромов) технически существенно проще.

В верхней части дерева разбора проверяются все те же условия, что в грамматике для палиндромов, при этом задаются префикс u и суффикс v — и так *до первой ошибки*, когда оказывается, что ни одна строка вида uxv исходной грамматикой не задаётся. Тогда, какие бы дополнительные символы x ни лежали внутри, строка должна задаваться новой грамматикой.

$$\begin{aligned} S_2 &\to sS_2s && (s \in \Omega) \\ S_2 &\to sXt && (s, t \in \Omega,\ s \neq t) \\ X &\to sX && (s \in \Omega) \\ X &\to \varepsilon \end{aligned}$$

Грамматика G'_1, задающая дополнение языка $L(G_1)$, строится так же, только нужно рассмотреть много видов ошибок: чтобы одна конфигурация не следовала за другой, определение перехода машины Тьюринга должно где-то нарушаться.

Построив грамматики $G'_1 = (\Sigma, N_1, R_1, S_1)$ и $G'_2 = (\Sigma, N_2, R_2, S_2)$, можно добавить к ним правила $S \to S_1 \mid S_2$ и получить грамматику для дополнения языка VALC(M). $\square$

Проверка пустоты языка, распознаваемого данной машиной Тьюринга M — это неразрешимая задача. Поскольку язык VALC(M) пуст тогда и только тогда, когда пуст $L(M)$, задача проверки VALC(M) на пустоту также неразрешима.

Теорема 4. *Следующие задачи неразрешимы:*

1. *пустота пересечения для двух данных грамматик;*
2. *определяет ли данная грамматика множество всех строк;*

Доказательство. Пусть есть алгоритм, определяющий пустоту пересечения двух данных грамматик. Тогда существует следующий алгоритм, проверяющий, пуст ли язык, распознаваемый данной машиной Тьюринга M. Алгоритм сперва построит грамматики G_1 и G_2 как в лемме 4, и затем проверит, пусто ли пересечение $L(G_1) \cap L(G_2) = \mathrm{VALC}(M)$. Если пересечение пусто, он примет, а если непусто, то отвергнет. Поскольку VALC(M) пуст тогда и только тогда, когда $L(M)$ пуст, алгоритм работает правильно. Но такого алгоритма существовать не может, поскольку пустота МТ неразрешима по теореме Райса.

Если по данной машине M построить грамматику для языка $\overline{\mathrm{VALC}(M)}$, то окажется, что она задаёт язык Σ^* тогда и только тогда, когда язык VALC(M) пуст — что неразрешимо. $\square$

Список литературы

[1984] R. P. Brent, L. M. Goldschlager, “A parallel algorithm for context-free parsing”, *Australian Computer Science Communications*, 6:7 (1984), 7.1–7.10.

[1965] T. Kasami, “An efficient recognition and syntax-analysis algorithm for context-free languages”, Report AF CRL-65-758, Air Force Cambridge Research Laboratory, USA, 1965.

[1965] P. M. Lewis II, R. E. Stearns, J. Hartmanis, “Memory bounds for recognition of context-free and context-sensitive languages”, *IEEE Conference Record on Switching Circuit Theory and Logical Design*, 1965, 191–202.

[1980] W. L. Ruzzo, “Tree-size bounded alternation”, *Journal of Computer and System Sciences*, 21:2 (1980), 218–235.

[1985] W. Rytter, "On the recognition of context-free languages", *Fundamentals of Computation Theory* (FCT 1985, Cottbus, Germany), LNCS 208, 315–322.

[1975] L. G. Valiant, "General context-free recognition in less than cubic time", *Journal of Computer and System Sciences*, 10:2 (1975), 308–314.

[1967] D. H. Younger, "Recognition and parsing of context-free languages in time n^3", *Information and Control*, 10 (1967), 189–208.

[1967] J. Hartmanis, "Context-free languages and Turing machine computations", *Proceedings of Symposia in Applied Mathematics*, Vol. 19, AMS, 1967, 42–51.