Теоретическая информатика, осень 2020 г. Лекция 8. Понятие о магазинных автоматах. Логика FO(LFP) как обобщение грамматик, её равносильность полиномиальному времени*

Александр Охотин

22 октября 2020 г.

Содержание

1 Понятие о магазинных автоматах

Недетерминированные магазинные автоматы (nondeterministic pushdown automata, NPDA), введённые Хомским и Шютценберже [1963] — это NFA, дополнительно оснащённые стеком. Кроме обычной смены состояний и чтения символов слева направо, на каждом шаге работы NPDA может извлечь верхний символ стека и записать символы в стек сверху.

Сама по себе эта модель не очень интересная. На заре теоретической информатики было доказано, что NPDA равносильны грамматикам — и с тех пор повелось воспроизводить этот результат из учебника в учебник. Это любопытный факт, хотя особенного толку от этой эквивалентности нет: строить NPDA неудобно, а если взяться доказывать результаты о грамматиках через NPDA, доказательства становятся неподъёмными.

Подлинный интерес представляет частный случай этой модели — *детерминированные магазинные автоматы (DPDA)*, впервые изученные Гинзбургом и Грейбах [1966]. Эта модель — теоретическая основа для синтаксических анализаторов, работающих за линейное время. Детерминированные магазинные автоматы слабее недетерминированных, и вокруг них разработана большая и интересная теория. Однако эта теория слишком велика, чтобы войти в базовый курс теоретической информатики.

Определение 1 (Хомский и Шютценберже [1963])**.** Недетерминированный магазинный автомат *(NPDA) — это семёрка* $\mathcal{A} = (\Sigma, Q, \Gamma, q_0, \perp, \delta, F)$*, состоящая из следующих компонентов:*

*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/`.

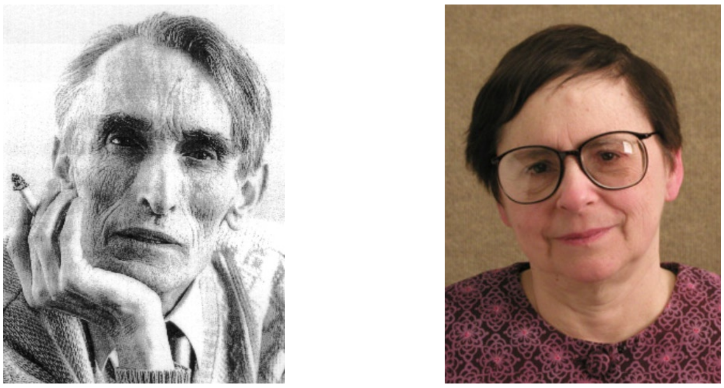

Рис. 1: Марсель-Поль Шютценберже (1920–1996), Шейла Грейбах (род. 1939).

- *конечный алфавит* Σ,
- *конечное множество состояний* Q,
- *конечный магазинный алфавит* Γ,
- *начальное состояние* $q_0 \in Q$,
- *символ дна магазина* $\perp \in \Gamma$,
- *функция переходов* $\delta \colon Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$, *где* $\delta(q, a, s) \ni (q', \gamma)$ *означает, что автомат, будучи в состоянии* $q \in \Sigma$, *читая входной символ* $a \in \Sigma \cup \{\varepsilon\}$ *и извлекая из стека символ* $s \in \Gamma$, *может перейти в состояние* q' *и записать в стек последовательность символов* γ,
- *множество принимающих состояний* $F \subseteq Q$.

Конфигурации автомата — тройки (q, w, x), *где* $q \in Q$ *— состояние,* $w \in \Sigma^*$ *— непрочитанная часть входной строки,* $\gamma \in \Gamma^*$ *— содержимое стека. Вводится отношение перехода на множестве этих троек:* $(q, uw, \gamma_0 s) \vdash (q', w, \gamma_0\gamma)$, *где* $\delta(q, u, s) \ni (q', \gamma)$. *Автомат принимает строку, если он прочитывает все входные символы и переходит в одно из принимающих состояний; содержимое стека при этом не имеет значения.*

$$L(\mathcal{A}) = \{\, w \in \Sigma^* \mid (q_0, w, \perp) \vdash^* (q_{acc}, \varepsilon, \gamma) \text{ для некоторых } q_{acc} \in F \text{ и } \gamma \in \perp\Gamma^* \,\}.$$

Теорема 1. *Пусть* $L \subseteq \Sigma^*$ *— язык. Тогда* L *задаётся грамматикой тогда и только тогда, когда* L *распознаётся NPDA.*

Набросок доказательства. $\Rightarrow$ Пусть L задаётся грамматикой $G = (\Sigma, N, R, S)$. Строится магазинный автомат, который будет моделировать построение дерева разбора снизу вверх, от листьев к корню. Стековый алфавит $\Gamma = \Sigma \cup N$ позволит автомату хранить в стеке последовательность меток вершин — корней поддеревьев.

Множество состояний автомата: $Q = \{\, q_\beta \mid A \to \alpha\beta \in R \,\} \cup \{q_S\} \cup \{r_{acc}\}$. Начальное состояние: q_ε. Переходы. Проталкивание входного символа:

$$\delta(q_\varepsilon, a, s) = (q_\varepsilon, sa)$$

Чтение верхних символов стека в состояние.

$$\delta(q_\beta, \varepsilon, X) = (q_{X\beta}, \varepsilon) \qquad (X \in \Sigma \cup N)$$

Свёртка тела правила, прочитанного в стеке, в нетерминальный символ.

$$\delta(q_\alpha, \varepsilon, s) = (q_\varepsilon, sA), \qquad (A \to \alpha \in R)$$

Наконец, чтобы принять, автомат проверяет, что в стеке лежит S.

$$\delta(q_S, \varepsilon, \perp) = (r_{acc}, \perp)$$

Инвариант: конкатенация стека, нижнего индекса состояния и непрочитанной части входной строки — это строка, из которой можно перезаписью строк получить первоначальную входную строку. В начале работы стек пуст, входная строка вся не прочитана — инвариант выполнен. Далее все переходы сохраняют инвариант.

$\Leftarrow$ Пусть L задаётся магазинным автоматом. Сперва автомат переделывается так, чтобы он перед принятием опустошал весь стек, включая символ дна стека, извлекаемый на последнем шаге; пусть $\mathcal{A} = (\Sigma, Q, \Gamma, q_0, \perp, \delta, F)$ — получившийся автомат. Строится грамматика со следующим множеством нетерминальных символов.

$$N = \{\, A^s_{p,q} \mid p, q \in Q,\ s \in \Gamma \,\} \cup \{S\}$$

Цель построения: всякий нетерминальный символ $A^s_{p,q}$ будет задавать множество всех таких строк $w \in \Sigma^*$, что автомат, начав вычисление в состоянии p, имея в стеке s, может прочитать w, не заглядывая в стек ниже символа s, а на последнем шаге извлечёт нижний символ стека (s или тот символ, которых будет записан на его месте) и перейдёт в состояние q. Если $s = \perp$, то нижний символ на последнем шаге не извлекается.

Правило для нетерминального символа $A^s_{p,q}$ начинается с применения перехода автомата. Пусть автомат в состоянии p извлечёт из стека верхний символ s, прочитает входной символ $a \in \Sigma \cup \{\varepsilon\}$ и запишет в стек строку стековых символов $s_1 \ldots s_\ell$ (s_1 — сверху). В успешном вычислении автомата все эти стековые символы будут рано или поздно извлечены из стека; пусть u_i — строка, читая которую, автомат извлечёт стековый символ s_i. Тогда грамматика должна задавать конкатенацию $au_1 \ldots u_\ell$. Это делается следующим правилом.

$$A^s_{p,q} \to aA^{s_1}_{r_0,r_1} \ldots A^{s_\ell}_{r_{\ell-1},r_\ell} \qquad (\delta(p, a, s) \ni (r_0, s_1 \ldots s_\ell),\ r_1, \ldots, r_{\ell-1} \in Q,\ r_\ell = q)$$

Каждая строка, принимаемая автоматом в состоянии $q \in F$, задаётся нетерминальным символом $A^\perp_{q_0,q}$; чтобы задать все такие строки грамматикой, нужны следующие правила.

$$S \to A^\perp_{q_0,q} \qquad (q \in F)$$

$\square$

2 Логика FO(LFP)

Для каждого понятия бывает интересно изучить более общее понятие, частным случаем которого оно является. Это даёт лучшую перспективу на исходное понятие, а также позволяет найти общее у, казалось бы, совсем разных вещей.

Грамматики — это частный случай чего? В старых учебниках обычно рассказывают про «иерархию Хомского» — но это устаревшая модель, в современной науке не нужная, и изучать её стоит разве что в курсе по истории науки.

Логика, частным случаем которой являются формальные грамматики: FO(LFP). Это достаточно простая логика, предназначенная для описания свойств строк; для философских рассуждений она непригодна. Изначально предложена в качестве языка запросов к базам данных, теоретически изучена в работах Иммермана [1986] и Варди [1982].

Название FO(LFP) означает *логику предикатов первого порядка* (first-order, FO) с семантикой *наименьшей неподвижной точки* (least fixed point, LFP). «Логика первого порядка» — это логика предикатов, допускающая использование кванторов (существует x; для любого x), в которой ведутся рассуждения о некоторых элементарных объектах (в данных случаях, о позициях в строке), но не о *множествах* таких объектов и не о *функциях* из объектов в объекты (это уже были бы объекты «второго порядка»).

«Неподвижная точка» — так называется решение уравнения вида $x = f(x)$. Формальное определение этой логики по сути использует именно такое уравнение.

2.1 Обобщение формальных грамматик

Раундс [1988]: представление грамматики как логической формулы, описывающей, какими свойствами должна обладать строка, чтобы быть грамматически правильной. Всякий нетерминальный символ A становится двуместным предикатом $A(x, y)$, где переменные x и y принимают значения *позиций в определяемой строке*. Позиции в строке $w = a_1 \dots a_n$ пронумерованы от 0 до n. Пара позиций (i, j) обозначает подстроку $a_{i+1} \dots a_j$.

Пример 1. *Грамматика для языка Дика.*

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

В логическом представлении, предикат $S(x, y)$ задаётся следующим рекуррентным соотношением: подстрока от позиции x до позиции y принадлежит языку Дика тогда и только тогда, когда выполняется одно из следующих трёх условий.

1. *Подстрока от x до y — это конкатенация двух строк из языка Дика, одна из которых простирается от позиции x до некоторой позиции z, а другая — от z до y.*

$$(\exists z)(S(x, z) \wedge S(z, y))$$

2. *В позиции $x + 1$ находится символ a, в позиции y находится b, а подстрока между позициями $x + 1$ и $y - 1$ принадлежит языку Дика.*

$$a(x + 1) \wedge S(x + 1, y - 1) \wedge b(y)$$

3. *Это пустая подстрока, то есть x и y совпадают.*

$$x = y$$

Тогда $S(x, y)$ можно определить как дизъюнкцию этих трёх условий.

Принадлежность строки $w \in \{a, b\}^*$ языку Дика тогда описывается высказыванием $S(0, |w|)$, которое может быть истинным или ложным.

Какие выразительные средства используются в этом описании? Во-первых, элементарная арифметика с номерами позиций. Всякий номер позиции в формуле задаётся элементарным выражением — *термом*.

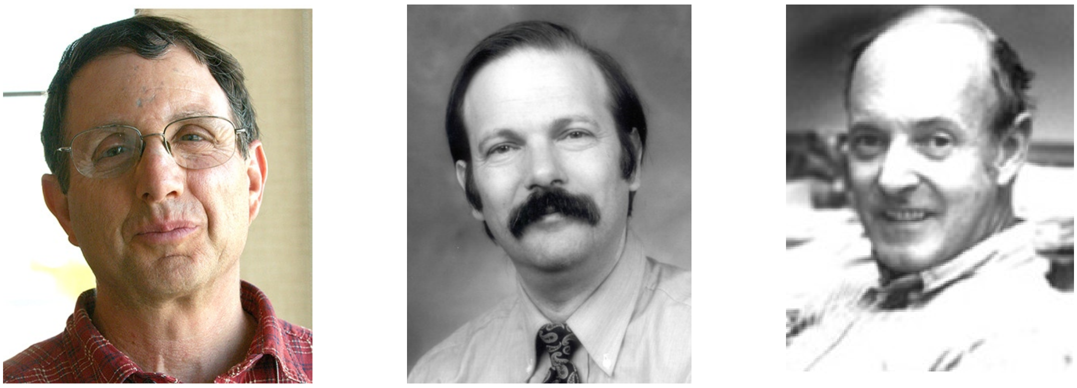

Рис. 2: Нейл Иммерман (род. 1953), Моше Варди (род. 1954), Вильям Раундс (род. 1942).

Определение 2. *Для данного множества переменных, термы определяются так:*

- *всякая переменная — терм;*
- *термы-константы начало и конец — обозначают начальную и конечную позицию в строке;*
- *если t — терм, то $t+1$ и $t-1$ — тоже термы.*

У всякого предиката — конечное число аргументов (позиций), и для всякого набора позиций он принимает значение «истина» или «ложь». Предикаты, обозначающие символы входной строки: $a(x)$, где $a \in \Sigma$, означает, что символ в позиции x — это a. Предикаты для сравнения номеров позиций: $x < y$ and $x = y$. Наконец, есть определяемые предикаты — такие как $S(x, y)$ в примере. Формулы строятся из предикатов с помощью связок — конъюнкции и дизъюнкции — и кванторов по номерам позиций.

Определение 3. *Пусть Σ — алфавит, пусть N — конечное множество предикатных символов, где всякий $A \in N$ имеет конечную размерность, обозначаемую через* $\dim A$.

- *Если $A \in N$, $\dim A = k$ и $t_1, \ldots, t_k$ — термы, то $A(t_1, \ldots, t_k)$ — формула;*
- *если $a \in \Sigma$ — символ и t — терм, то $\varphi = a(t)$ — формула;*
- *если t и t' — термы, то $t < t'$ и $t = t'$ — формулы;*
- *если φ и ψ — формулы, то $\varphi \vee \psi$ и $\varphi \wedge \psi$ — тоже формулы;*
- *если φ — формула, а x — свободная переменная в φ, то $(\exists x)\varphi$ и $(\forall x)\varphi$ — тоже формулы.*

Сокращённые обозначения: $t > t'$ значит $t' < t$; $t \neq t'$ значит $t < t' \vee t > t'$, $t \leqslant t'$ значит $t < t' \vee t = t'$, $t \geqslant t'$ значит $t' < t \vee t = t'$.

Рекурсивное определение: всякий предикат $A(x_1, \ldots, x_n)$ определяется формулой $\varphi_A(x_1, \ldots, x_n)$.

Определение 4 (Раундс [1988]). *FO(LFP)-определение языка — это пятёрка* $G = (\Sigma, N, \dim, \langle\varphi_A\rangle_{A \in N}, \sigma)$, *где*

- Σ — *алфавит;*
- N — *конечное множество предикатных символов;*
- $\dim \colon N \to \mathbb{N}$ — *функция, задающее число аргументов;*
- *всякий предикат* $A \in N$ *определяется формулой* φ_A *с* $\dim A$ *свободных переменных;*
- *формула* σ, *без свободных переменных, задаёт условие принадлежности строки языку.*

Запись: $A(x_1, \ldots, x_{\dim A}) = \varphi_A(x_1, \ldots, x_{\dim A})$, *отдельно задаётся* σ.

Формальная запись примера 1.

Пример 2. *FO(LFP)-определение* $G = (\Sigma, \{S\}, \dim, \langle\varphi_S\rangle, \sigma)$, *где* $\dim S = 2$, *и предикат* S *задаётся следующей формулой.*

$$S(x,y) = \underbrace{\big[(\exists z)(S(x,z) \wedge S(z,y))\big] \vee (a(x+1) \wedge S(x+1, y-1) \wedge b(y)) \vee x = y}_{\varphi_S}$$

Далее, $\sigma = S(\underline{\text{начало}}, \underline{\text{конец}})$.

3 Равносильность полиномиального времени и FO(LFP)

Класс сложности P (полиномиальное время) — задачи, решаемые машиной Тьюринга за полиномиальное время. Язык L лежит в классе P, если существует многочлен $p(n)$ и МТ, останавливающаяся на всякой строке w за время $p(|w|)$ и распознающая принадлежность языку L.

Теорема 2 (Иммерман [1986]; Варди [1982]). *Для всякой машины Тьюринга M с входным алфавитом Σ, распознающей язык $L \subseteq \Sigma^*$ за время $O(n^k)$, существует и может быть эффективно построено FO(LFP)-определение $G = (\Sigma, N, \dim, \langle\varphi_A\rangle_{A \in N}, \sigma)$ языка $L(M)$, в котором наибольшая размерность предиката — $2k$, и глубина вложения кванторов — тоже $2k$.*

На самом деле, можно обойтись без кванторов — ценой не очень большого усложнения построения. Но для доказательства утверждения о равносильности FO(LFP) и P такая оптимизация не требуется.

Доказательство. Машина Тьюринга $\mathcal{M} = (\Sigma, \Gamma, Q, q_0, \delta, q_{acc}, q_{rej})$: входной алфавит Σ; рабочий алфавит Γ, где $\Sigma \subset \Gamma$; множество состояний Q; начальное q_0; функция переходов $\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$; принимающее q_{acc}; отвергающее q_{rej}. Также предполагается, что машина Тьюринга никогда не заходит левее клетки, где находился первый символ входной строки.

Пусть на входной строке длины n машина работает не более чем за $(n+1)^k - 1$ шагов. Стало быть, она также использует не более чем $(n+1)^k - 1$ ячеек памяти (дальше она не успеет заехать).

Тогда номер шага и положение головки можно кодировать *k-значными числами в системе счисления по основанию $n+1$* — иными словами, *наборами* $(x_1, \ldots, x_k)$ *позиций во входной строке*. Всякий такой набор соответствует числу $\sum_{i=1}^{k} x_i \cdot (n+1)^{i-1}$, и всякое число от 0 до $(n+1)^k - 1$ представляется таким набором.

Цель: определить предикаты следующих двух видов. Во-первых, для всякого состояния $q \in Q$, предикат $A_q(x_1, \ldots, x_k, y_1, \ldots, y_k)$, означает, что во время $(x_1, \ldots, x_k)$ машина была в состоянии q, а её головка — в позиции $(y_1, \ldots, y_k)$. Во-вторых, предикат $C_a(x_1, \ldots, x_k, y_1, \ldots, y_k)$, где $a \in \Gamma$, значит, что в момент времени $(x_1, \ldots, x_k)$ в клетке $(y_1, \ldots, y_k)$ находился символ a.

Сперва задаётся служебный предикат $\mathrm{inc}(x_1, \ldots, x_k, x'_1, \ldots, x'_k)$, проверяющий, что наборы позиций $\boldsymbol{x} = (x_1, \ldots, x_k)$ и $\boldsymbol{x'} = (x'_1, \ldots, x'_k)$ кодируют два числа, отличающиеся на единицу. Необходимо проверить правильность прибавления единицы в системе счисления по основанию $n+1$; для этого нужно убедиться, что числа имеют следующий вид, для некоторого $i \in \{1, \ldots k\}$.

$$\begin{array}{cccccccccc}
 & x_k & x_{k-1} & \ldots & x_{i+1} & x_i & n & \ldots & n \\
+ & & & & & & & & 1 \\
\hline
 & x_k & x_{k-1} & \ldots & x_{i+1} & x_i+1 & 0 & \ldots & 0
\end{array}$$

Проверяющий это предикат задаётся так.

$$\begin{aligned}
\mathrm{inc}(x_1, \ldots, x_k, x'_1, \ldots, x'_k) = \bigvee_{i=1}^{k} \big(& (x_1 = \underline{\text{конец}} \wedge x'_1 = \underline{\text{начало}}) \wedge \\
& \ldots \\
& \wedge (x_{i-1} = \underline{\text{конец}} \wedge x'_{i-1} = \underline{\text{начало}}) \wedge \\
& \wedge\ x_i + 1 = x'_i\ \wedge \\
& \wedge\ x_{i+1} = x'_{i+1}\ \wedge \\
& \ldots \\
& \wedge\ x_k = x'_k \big)
\end{aligned}$$

Определение предиката $A_q(x_1, \ldots, x_k, y_1, \ldots, y_k)$ — дизъюнкция трёх условий. Если на шаге с номером $\boldsymbol{x} = (x_1, \ldots, x_k)$ машина находится в состоянии q, в позиции $\boldsymbol{y} = (y_1, \ldots, y_k)$, то она прибыла туда слева или справа, или же это начальная конфигурация.

$$\begin{aligned}
A_q(\boldsymbol{x}, \boldsymbol{y}) = & \bigvee_{\substack{q' \in Q,\, a, a' \in \Gamma: \\ \delta(q', a') = (q, a, +1)}} (\exists \boldsymbol{x'})(\exists \boldsymbol{y'}) \Big(\underbrace{\mathrm{inc}(\boldsymbol{x'}, \boldsymbol{x})}_{\substack{\boldsymbol{x'} \text{ на шаг} \\ \text{раньше } \boldsymbol{x}}} \wedge \underbrace{\mathrm{inc}(\boldsymbol{y'}, \boldsymbol{y})}_{\substack{\boldsymbol{y'} \text{ на 1 поз.} \\ \text{слева от } \boldsymbol{y}}} \wedge \underbrace{A_{q'}(\boldsymbol{x'}, \boldsymbol{y'})}_{\substack{\text{МТ была слева} \\ \text{в сост. } q'}} \wedge \underbrace{C_{a'}(\boldsymbol{x'}, \boldsymbol{y'})}_{\substack{\text{слева был} \\ \text{символ } a'}} \Big) \vee \\
& \vee \bigvee_{\substack{q' \in Q,\, a, a' \in \Gamma: \\ \delta(q', a') = (q, a, -1)}} (\exists \boldsymbol{x'})(\exists \boldsymbol{y'}) \Big(\underbrace{\mathrm{inc}(\boldsymbol{x'}, \boldsymbol{x})}_{\substack{\boldsymbol{x'} \text{ на шаг} \\ \text{раньше } \boldsymbol{x}}} \wedge \underbrace{\mathrm{inc}(\boldsymbol{y}, \boldsymbol{y'})}_{\substack{\boldsymbol{y'} \text{ на 1 поз.} \\ \text{справа от } \boldsymbol{y}}} \wedge \underbrace{A_{q'}(\boldsymbol{x'}, \boldsymbol{y'})}_{\substack{\text{МТ была справа} \\ \text{в сост. } q'}} \wedge \underbrace{C_{a'}(\boldsymbol{x'}, \boldsymbol{y'})}_{\substack{\text{справа был} \\ \text{символ } a'}} \Big) \vee \\
& \vee \underbrace{(\boldsymbol{x} = \boldsymbol{0} \wedge \boldsymbol{y} = \boldsymbol{0})}_{\text{только если } q = q_0}
\end{aligned}$$

Сокращённая запись $\boldsymbol{x} = \boldsymbol{0}$ означает формулу вида $\bigvee_{i=1}^{k} (x_i = \underline{\text{начало}})$.

Предикат $C_a(x_1, \ldots, x_k, y_1, \ldots, y_k)$ определяется следующим условием. Во время $\boldsymbol{x} = (x_1, \ldots, x_k)$ в позиции $\boldsymbol{y} = (y_1, \ldots, y_k)$ символ a стоит в одном из следующих случаев: или он только что был туда записан; или он уже был там на прошлом шаге и не был на прошлом шаге перезаписан, поскольку головка была в каком-то другом месте; или это начальная конфигурация, и в ней на этом месте положено быть именно этому символу.

$$\begin{aligned}
C_a(\boldsymbol{x},\boldsymbol{y}) =&(\exists \boldsymbol{x'})\Big(\underbrace{\mathrm{inc}(\boldsymbol{x'},\boldsymbol{x})}_{\substack{\boldsymbol{x'}\text{ на шаг}\\ \text{раньше } \boldsymbol{x}}} \wedge \bigvee_{\substack{\widetilde{q}\in Q,\ \widetilde{a}\in\Gamma:\\ \delta(\widetilde{q},\widetilde{a})=(q,a,d)\\ \text{для каких-то}\\ q\in Q,\ d\in\{-1,+1\}}} \big(\underbrace{A_{\widetilde{q}}(\boldsymbol{x'},\boldsymbol{y})}_{\substack{\text{МТ была здесь}\\ \text{в сост. } \widetilde{q}}} \wedge \underbrace{C_{\widetilde{a}}(\boldsymbol{x'},\boldsymbol{y})}_{\substack{\text{здесь был}\\ \text{символ } \widetilde{a}}}\big)\Big) \vee \\
&\vee (\exists \boldsymbol{x'})(\exists \boldsymbol{y'})\Big(\underbrace{\mathrm{inc}(\boldsymbol{x'},\boldsymbol{x})}_{\substack{\boldsymbol{x'}\text{ на шаг}\\ \text{раньше } \boldsymbol{x}}} \wedge \underbrace{C_a(\boldsymbol{x'},\boldsymbol{y})}_{\substack{\text{здесь был}\\ \text{символ } a}} \wedge \underbrace{\boldsymbol{y'} \neq \boldsymbol{y}}_{\boldsymbol{y'}\text{ не здесь}} \wedge \underbrace{\bigvee_{q\in Q} A_q(\boldsymbol{x'},\boldsymbol{y'})}_{\text{МТ была не здесь}} \Big) \vee \\
&\vee \underbrace{\Big(\boldsymbol{x}=\mathbf{0} \wedge y_1 < \underline{\text{конец}} \wedge \bigwedge_{i=2}^{k} y_i = \underline{\text{начало}} \wedge a'(y_1+1)\Big)}_{\text{если } a'\in\Sigma} \vee \underbrace{\Big(\boldsymbol{x}=\mathbf{0} \wedge y_1 = \underline{\text{конец}} \vee \bigwedge_{i=2}^{k} y_i > \underline{\text{начало}}\Big)}_{\text{если } a' = \text{␣}}
\end{aligned}$$

Сокращённая запись $\boldsymbol{y'} \neq \boldsymbol{y}$ означает формулу вида $\bigvee_{i=1}^{k}(y_i' < y_i \vee y_i' > y_i)$.

Главная формула: на некотором шаге вычисления $\boldsymbol{x}$, машина находится в принимающем состоянии.

$$\sigma = (\exists \boldsymbol{x})(\exists \boldsymbol{y}) A_{q_{acc}}(\boldsymbol{x},\boldsymbol{y})$$

□

Следствие 1. *Язык определяется в логике FO(LFP) тогда и только тогда, когда он распознаётся машиной Тьюринга за полиномиальное время.*

Список литературы

[1963] N. Chomsky, M. P. Schützenberger, “The algebraic theory of context-free languages”, in: Braffort, Hirschberg (Eds.), *Computer Programming and Formal Systems*, North-Holland, 1963, 118–161.

[1966] S. Ginsburg, S. A. Greibach, “Deterministic context-free languages”, *Information and Control*, 9:6 (1966), 620–648.

[1986] N. Immerman, “Relational queries computable in polynomial time”, *Information and Control*, 68:1–3 (1986), 86–104.

[1988] W. C. Rounds, “LFP: A logic for linguistic descriptions and an analysis of its complexity”, *Computational Linguistics*, 14:4 (1988), 1–9.

[1982] M. Y. Vardi, “The complexity of relational query languages”, *STOC 1982*, 137–146.