

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

Александр Охотин

23 декабря 2020 г.

## Содержание

1	Понятие о магазинных автоматах	1
2	Логика $FO(LFP)$	4
3	Равносильность полиномиального времени и $FO(LFP)$	9

## 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 переписывать этот результат из учебника в учебник. Это любопытный факт, хотя никаких важных следствий из него извлечь не удаётся: строить NPDA неудобно, а если взяться доказывать результаты о грамматиках в терминах NPDA, доказательства становятся неподъёмными.<sup>1</sup>

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 по объёму, чтобы войти в базовый курс теоретической информатики.

---

\*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/](http://users.math-cs.spbu.ru/~okhotin/teaching/tcs_fl_2020/).

<sup>1</sup>Зато их можно пропускать, с оговоркой, что «можно доказать» — что и проделано во многих учебниках. Такова практическая выгода от перехода к более запутанной модели!

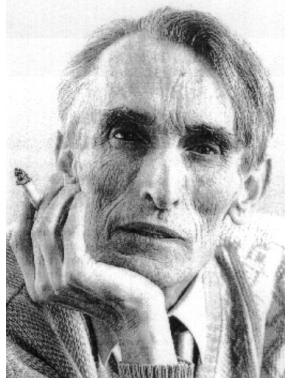


Рис. 1: Марсель-Поль Шютценберже (1920–1996), Шейла Грейбах (род. 1939).

**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:

- конечный алфавит  $\Sigma$ ,
- конечное множество состояний  $Q$ ,
- конечный магазинный алфавит  $\Gamma$ ,
- начальное состояние  $q_0 \in Q$ ,
- символ дна магазина  $\perp \in \Gamma$ ,
- функция переходов  $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$ , где  $\delta(q, a, s) \ni (q', \gamma)$  означает, что автомат, будучи в состоянии  $q \in Q$ , 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  $\gamma$ ,
- множество принимающих состояний  $F \subseteq Q$ .

Конфигурации автомата — тройки  $(q, w, x)$ , где  $q \in Q$  — состояние,  $w \in \Sigma^*$  — непрочитанная часть входной строки,  $x \in \Gamma^*$  — 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.

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

**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:  $Q = \{q_\beta \mid A \rightarrow \alpha\beta \in R\} \cup \{q_S\} \cup \{r_{acc}\}$ . Состояние  $q_\beta$  означает, что сверху стека были символы  $\beta$ , которые автомат взял оттуда взаймы в свой буфер конечного размера. Так автомат может извлечь из стека правую часть некоторого правила, чтобы потом вернуть долг, записав в стек нетерминальный символ, определяемый этим правилом.

Начальное состояние —  $q_\varepsilon$ : ничего из стека не извлечено, да и сам стек в это время пусь. Есть четыре типа переходов. Во-первых, автомат может протолкнуть очередной входной символ в стек.

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

Верхний символ стека можно прочесть в буфер.

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

Тело правила, обнаруженного сверху стека, можно свернуть в нетерминальный символ.

$$\delta(q_\alpha, \varepsilon, s) = (q_\varepsilon, sA), \quad (A \rightarrow \alpha \in R)$$

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

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

Автомат поддерживает следующий инвариант: в каждый момент времени, 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  $w$  вся не прочитана — инвариант выполнен. Далее все переходы сохраняют инвариант — поэтому, если автомат когда-нибудь примет, то для предпоследней его конфигурации  $(q_S, \varepsilon, \perp)$  верно, что  $S$  можно перезаписать в  $w$ .

⊕ 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.

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

Цель построения: всякий нетерминальный символ  $A_{p,q}^s$  будет задавать множество всех таких строк  $w \in \Sigma^*$ , что автомат, начав вычисление в состоянии  $p$ , имея в стеке  $s$ , может прочесть  $w$ , 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  $A_{p,q}^s$  начинается с применения перехода автомата. Пусть автомат в состоянии  $p$  извлечёт из стека верхний символ  $s$ , прочтёт входной символ  $a \in \Sigma \cup \{\varepsilon\}$  и запишет в стек строку стековых символов  $s_1 \dots s_\ell$  ( $s_1$  — 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  $au_1 \dots u_\ell$ . Это делается следующим правилом.

$$A_{p,q}^s \rightarrow aA_{r_0,r_1}^{s_1} \dots A_{r_{\ell-1},r_\ell}^{s_\ell} \quad (\delta(p, a, s) \ni (r_0, s_1 \dots s_\ell), r_1, \dots, r_{\ell-1} \in Q, r_\ell = q)$$

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

$$S \rightarrow A_{q_0,q}^\perp \quad (q \in F)$$

□

## 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.

$$S \rightarrow SS \mid aSb \mid \varepsilon$$

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.

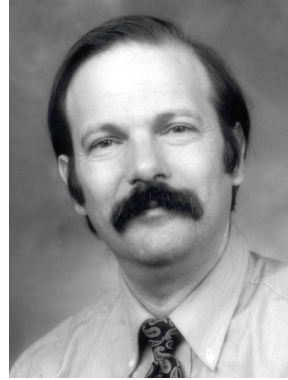


Рис. 2: Нейл Иммерман (род. 1953), Моше Варди (род. 1954), Вильям Раундс (род. 1942).

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

**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  $\Sigma$  — 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  $t_1, \dots, t_k$  — термы, то  $A(t_1, \dots, t_k)$  — 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  $\varphi$  и  $\psi$  — формулы, то  $\varphi \vee \psi$  и  $\varphi \wedge \psi$  — тоже формулы;
- если  $\varphi$  — формула, а  $x$  — свободная переменная в  $\varphi$ , то  $(\exists x)\varphi$  и  $(\forall x)\varphi$  — тоже формулы.

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

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

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

- $\Sigma$  — алфавит;
- $N$  — конечное множество предикатных символов;
- $\dim: N \rightarrow \mathbb{N}$  — функция, задающее число аргументов;
- всякий предикат  $A \in N$  определяется формулой  $\varphi_A$  с  $\dim A$  свободных переменных;
- формула  $\sigma$ , без свободных переменных, задаёт условие принадлежности строки языку.

Запись:  $A(x_1, \dots, x_{\dim A}) = \varphi_A(x_1, \dots, x_{\dim A})$ , отдельно задаётся  $\sigma$ .

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.

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

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

## 2.2 Строгое определение

Сперва определяется значение терма на данной строке и данном наборе номеров позиций. У термов, как они были определены, бывает 0, 1 или 2 аргумента; ради единообразия обозначений определение даётся для  $n$  аргументов.

**Определение 2(Е).** Пусть  $w \in \Sigma^*$  — строка,  $t(x_1, \dots, x_n)$  — терм, и  $x_1 = i_1, \dots, x_n = i_n$ , где  $i_1, \dots, i_n \in \{0, 1, \dots, |w|\}$ , — значения переменных. Тогда значение терма  $t(i_1, \dots, i_n)$  определяется так:

- если  $t = x_j$ , то  $t(i_1, \dots, i_n) = i_j$ ;
- если  $t = \underline{\text{начало}}$ , то  $t(i_1, \dots, i_n) = 0$ , и если  $t = \underline{\text{конец}}$ , то  $t(i_1, \dots, i_n) = |w|$ ;
- для всякого терма  $t$ , если  $t(i_1, \dots, i_n) > 0$ , то  $(t-1)(i_1, \dots, i_n) = t(i_1, \dots, i_n) - 1$  — иначе значение  $t-1$  не определено; аналогично,  $(t+1)(i_1, \dots, i_n) = t(i_1, \dots, i_n) + 1$ , коль скоро результат не превосходит  $|w|$ .

Истинность формулы.

**Определение 3(Е).** Пусть  $\Sigma$  — алфавит, пусть  $N$  — конечное множество предикатных символов. Для строки  $w \in \Sigma^*$ , задаётся множество элементарных утверждений о ней.

$$\widehat{I}_w = \{ A(i_1, \dots, i_k) \mid A \in N, k = \dim A, i_1, \dots, i_k \in \{0, \dots, |w|\} \}$$

Всякое подмножество  $I \subseteq \widehat{I}_w$  утверждений, предполагаемых истинными, называется **интерпретацией**. Тогда, для всякой формулы  $\varphi(x_1, \dots, x_n)$  с  $n$  свободными аргументами, и для всех значений  $i_1, \dots, i_n \in \{0, \dots, |w|\}$ , утверждение  $\varphi(i_1, \dots, i_n)$  называется истинным на строке  $w \in \Sigma^*$  в интерпретации  $I$  (обозначение:  $I \models_w \varphi(i_1, \dots, i_n)$ ), если формула удовлетворяет следующему индуктивному определению.

- Если  $\varphi(x_1, \dots, x_n) = A(t_1, \dots, t_k)$ , то  $I \models_w \varphi(i_1, \dots, i_n)$ , если  $A(t_1(i_1, \dots, i_n), \dots, t_k(i_1, \dots, i_n)) \in I$ .
- Если  $\varphi(x_1, \dots, x_n) = a(t)$ , то  $I \models_w \varphi(i_1, \dots, i_n)$ , если символ строки  $w$  в позиции  $t(i_1, \dots, i_n)$  — это  $a$ .
- Если  $\varphi(x_1, \dots, x_n) = (t < t')$ , то  $I \models_w \varphi(i_1, \dots, i_n)$ , если число  $t(i_1, \dots, i_n)$  меньше, чем  $t'(i_1, \dots, i_n)$ .
- Если  $\varphi(x_1, \dots, x_n) = (t = t')$ , то  $I \models_w \varphi(i_1, \dots, i_n)$ , если числа  $t(i_1, \dots, i_n)$  и  $t'(i_1, \dots, i_n)$  равны.
- Если  $\varphi(x_1, \dots, x_n) = \psi \vee \xi$ , то  $I \models_w \varphi(i_1, \dots, i_n)$ , если  $I \models_w \psi(i_1, \dots, i_n)$  или  $I \models_w \xi(i_1, \dots, i_n)$ .
- Если  $\varphi(x_1, \dots, x_n) = \psi \wedge \xi$ , то  $I \models_w \varphi(i_1, \dots, i_n)$ , если  $I \models_w \psi(i_1, \dots, i_n)$  и  $I \models_w \xi(i_1, \dots, i_n)$ .
- Если  $\varphi(x_1, \dots, x_n) = (\exists x)\psi(x_1, \dots, x_n, x)$ , то  $I \models_w \varphi(i_1, \dots, i_n)$ , если  $I \models_w \psi(i_1, \dots, i_n, i)$  для некоторого числа  $i \in \{0, \dots, |w|\}$ ,
- Если  $\varphi(x_1, \dots, x_n) = (\forall x)\psi(x_1, \dots, x_n, x)$ , то  $I \models_w \varphi(i_1, \dots, i_n)$ , если  $I \models_w \psi(i_1, \dots, i_n, i)$  для всякого числа  $i \in \{0, \dots, |w|\}$ .

**Определение 4(Е).** Пусть  $G = (\Sigma, N, \dim, \langle \varphi_A \rangle_{A \in N}, \sigma)$  —  $FO(LFP)$ -определение. Пусть  $w \in \Sigma^*$  — строка. Интерпретация  $I \subseteq \widehat{I}_w$  — это **модель**, если  $A(i_1, \dots, i_{\dim A}) \in I$  тогда и только тогда, когда  $I \models_w \varphi_A(i_1, \dots, i_{\dim A})$ , для всех  $A \in N$  и для всех номеров позиций. Утверждение  $\psi(x_1, \dots, x_k)$  называется **истинным** на  $w$  (обозначение:  $\models_w \psi(i_1, \dots, i_k)$ ), если  $I \models_w \psi(i_1, \dots, i_k)$  для всякой модели  $I$ . Задаваемый язык:  $L(G) = \{ w \in \Sigma^* \mid \models_w \sigma \}$ .

Всегда есть наименьшая модель.

**Лемма 1.**  $G = (\Sigma, N, \dim, \langle \varphi_A \rangle_{A \in N}, \sigma)$ , строка  $w \in \Sigma^*$ . Определяется оператор  $\Phi: 2^{\widehat{I}_w} \rightarrow 2^{\widehat{I}_w}$ , такой, что  $\Phi(I)$  — это множество всех элементарных утверждений, которые будут выводиться из формул  $\varphi_A$  в предположении, что все утверждения из  $I$  истинны.

$$\Phi(I) = \{ A(i_1, \dots, i_{\dim A}) \mid I \models_w \varphi_A(i_1, \dots, i_{\dim A}) \}.$$

Оператор  $\Phi$  монотонен, т.е., из  $I \subseteq I'$  следует  $\Phi(I) \subseteq \Phi(I')$ . Применяя этот оператор итеративно, начиная с пустого множества, получается возрастающая последовательность подмножеств  $\widehat{I_w}$ , сходящая к следующему пределу.

$$I_w = \bigcup_{\ell \geq 0} \Phi^\ell(\emptyset)$$

Тогда  $I_w$  — модель, и, более того, наименьшая модель, то есть, всякая модель  $I$  содержит  $I_w$ .

**Пример 3** (см. пример 2).  $FO(LFP)$ -определение для языка Дика.

$$S(x, y) = [(\exists z)(S(x, z) \wedge S(z, y))] \vee (a(x+1) \wedge S(x+1, y-1) \wedge b(y)) \vee x = y$$

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

Для строки  $w = \text{abaabb}$  строится последовательность интерпретаций.

$$\begin{aligned} I_w^{(0)} &= \emptyset \\ I_w^{(1)} &= \{S(0, 0), \dots, S(6, 6)\} \\ I_w^{(2)} &= \{S(0, 0), \dots, S(6, 6), S(0, 2), S(3, 5)\} \\ I_w^{(3)} &= \{S(0, 0), \dots, S(6, 6), S(0, 2), S(3, 5), S(2, 6)\} \\ I_w^{(4)} &= \{S(0, 0), \dots, S(6, 6), S(0, 2), S(3, 5), S(2, 6), S(0, 6)\} \\ I_w^{(5)} &= I_w^{(4)} = I_w \end{aligned}$$

Поскольку  $S(0, 6) \in I_w$ , формула  $\sigma$  истинна на наименьшей модели, и  $w \in L(G)$ .

**Упражнение 1.** Построить  $FO(LFP)$ -определения для языков  $\{a^n b^n c^n \mid n \geq 0\}$ ,  $\{a^n b^{in} \mid i, n \geq 0\}$ ,  $\{w w \mid w \in \{a, b\}^*\}$ ,  $\{a^{2^n} \mid w \in \{a, b\}^*\}$ .

## 2.3 Алгоритм распознавания

Вывести все верные утверждения о данной строке, построив наименьшую модель.

**Теорема 2.**  $G = (\Sigma, N, \dim, \langle \varphi_A \rangle_{A \in N}, \sigma)$ ,  $k = \max_{A \in N} \dim A$ ,  $m$  — наибольшая глубина вложения кванторов в  $\varphi_A$  и  $\sigma$ . Тогда существует алгоритм, который, по данной строке  $w \in \Sigma^*$ , проверяет принадлежность  $w$  языку  $L(G)$  за время  $O(n^{2k+m})$ , используя память  $O(n^k)$ .

*Доказательство.* Всего  $O(n^k)$  утверждений, и в худшем случае на каждом шаге получится вывести лишь одно из них. На каждом шаге, попробовать вывести каждое недоказанное из этих  $O(n^k)$  утверждений. Для всякого квантора надо проверить  $n$  возможных значений связанной переменной, поэтому всякая попытка доказать одно из утверждений требует  $O(n^m)$  шагов.  $\square$



Для грамматик, размерность всякого предиката —  $k = 2$ , кванторы друг в друга не вкладываются ( $m = 1$ ), и отсюда выходит время работы  $O(n^5)$ . На самом-то деле можно за время  $O(n^3)$  и даже за  $O(n^\omega)$ .

А насколько сложные языки можно задать логикой FO(LFP)? Чтобы ответить на этот вопрос, нужно начать со строгого определения сложности вычислений.

### 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 машина Тьюринга, останавливающаяся на всякой строке  $w$  за время  $p(|w|)$  и распознающая принадлежность языку  $L$ .

**Теорема 3** (Иммерман [1986]; Варди [1982]). *Для всякой машины Тьюринга  $M$  с входным алфавитом  $\Sigma$ , 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  $M = (\Sigma, \Gamma, Q, q_0, \delta, q_{acc}, q_{rej})$ : входной алфавит  $\Sigma$ ; рабочий алфавит  $\Gamma$ , где  $\Sigma \subset \Gamma$ ; множество состояний  $Q$ ; начальное  $q_0$ ; функция переходов  $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{-1, +1\}$ ; 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  $(x_1, \dots, x_k)$  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  $A_q(x_1, \dots, x_k, y_1, \dots, y_k)$ , означает, что во время  $(x_1, \dots, x_k)$  машина была в состоянии  $q$ , а её головка — в позиции  $(y_1, \dots, y_k)$ . Во-вторых, предикат  $C_a(x_1, \dots, x_k, y_1, \dots, y_k)$ , где  $a \in \Gamma$ , значит, что в момент времени  $(x_1, \dots, x_k)$  в клетке  $(y_1, \dots, y_k)$  находился символ  $a$ .

Сперва задаётся служебный предикат  $\text{inc}(x_1, \dots, x_k, x'_1, \dots, x'_k)$ , проверяющий, что наборы позиций  $\mathbf{x} = (x_1, \dots, x_k)$  и  $\mathbf{x}' = (x'_1, \dots, x'_k)$  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  $i \in \{1, \dots, k\}$ .

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

Проверяющий это предикат задаётся так.

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

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

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

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

Предикат  $C_a(x_1, \dots, x_k, y_1, \dots, y_k)$  определяется следующим условием. Во время  $\mathbf{x} = (x_1, \dots, x_k)$  в позиции  $\mathbf{y} = (y_1, \dots, y_k)$  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.

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

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

Главная формула выражает то условие, что на некотором шаге вычисления  $\mathbf{x}$ , машина находится в принимающем состоянии.

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

□

**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.