УДК 510.52 + 58

И.В.Латкин

Восточно-Казахстанский государственный технический университет им. Д.С.Серикбаева, Усть-Каменогорск (E-mail: lativan@yandex.ru)

Неравенство младших классов полиномиально-ограниченной иерархии языков

Изучены младшие, т.е. первые, самые маленькие, классы полиномиально-ограниченной иерархии языков. На основании ранее доказанной автором возможности моделирования вычислений экспоненциальной длины формулами булевых алгебр сделан вывод о различии, по меньшей мере, двух самых маленьких классов этой иерархии. Вопрос о совпадении других классов остаётся открытым.

Ключевые слова: машины Тьюринга, полиномиально-ограниченная иерархия языков, моделирование вычислений, определимость.

Принимается соглашение о выборе *базовой модели вычислений*: подразумевается, что все вычисления выполняются только на *детерминированных* машинах Тьюринга (далее — просто *машинах*), а сложность алгоритма, что-то решающего (распознающего или допускающего), — это временная сложность работы соответствующей машины.

1 О работе машин. Многие детали доказательства основной теоремы в значительной мере используют подробности конкретного способа записи программ одноленточных машин и их работы. Покажем, что применение этой специфики не выводит за рамки полиномиальной вычислимости. Кроме того, любое из описанных ниже ограничений устранимо за счёт усложнения доказательства основной теоремы.

При использовании только одной ленты время работы машины возрастает, но остаётся полиномиальным [1].

Предполагается, что лента бесконечна только вправо, и машина начинает и заканчивает работу в положении, когда головка обозревает крайнюю левую ячейку, в которой ничего не записано. Оценим возрастание числа шагов T при этих ограничениях. Допустим, что в процессе вычислений на ленте, бесконечной в обе стороны, была сделана запись левее начальной клетки, и для дальнейшего хода алгоритма положение этой записи (справа или слева) имеет значение. В этом случае запишем её всё равно в правой части, заключив в кавычки, состоящие из кода специального символа. Сама эта процедура добавляет относительно мало шагов — порядка общего числа символов, которые имеются в это время на ленте. Но каждое обращение к этой вспомогательной записи в правой части может, в свою очередь, приводить примерно к такому же возрастанию количества шагов по сравнению с её положением в левой. Тем не менее, таких обращений — не больше, чем общее число шагов в первоначальном варианте алгоритма, т.е. имеет верхнюю границу $O(T)$. Возврат головки в исходное положение исполняется за время порядка длины, окончательной записи на ленте.

Машинные команды у нас — однотактные, т.е. имеют вид $q_j\alpha \to q_r\beta$, а не двухтактные, вида $q_j\alpha \to q_r\beta\gamma$, где $\alpha \in A$, $\beta,\gamma \in A \cup \{R, L\}$ и A — рабочий алфавит. Если даже считать исполнение двухтактной команды за один шаг, то разница во времени работы получается линейная. Машина начинает работу во внутреннем состоянии q_1, и если она распознаёт вход, то приходит либо в допускающее состояние q_0, либо в отвергающее — q_2.

Предполагается также, что используемые машины не попадают в ситуацию, когда она остановилась с не определённым ответом. А именно, они не пытаются выйти за левый край ленты и не содержат висячих внутренних состояний q_j, для которых $j \neq 0,2$ и есть команды с концом $\ldots \to q_j\beta$, но нет команд с началом $q_j\alpha \to$ хотя бы для одного $\alpha \in A$. Попытку выйти за левый край ленты можно блокировать введением дополнительного символа $\#$, которым помечается левый край ленты, и заменой команд вида $q_r\# \to q_kL$ командами $q_r\# \to q_r\#$. Висячие внутренние состояния ликвидируются дописыванием команд вида $q_j\alpha \to q_j\alpha$.

2 О кодах и алфавитах. Рабочий алфавит $E^{\Lambda}=\{1;0(\Lambda)\}$ у применяемых нами машин имеет всего два символа 0 и 1, где 0 означает также и пустой символ. Но обычно он состоит, по крайней мере, из двух основных символов и символа пустой клетки Λ (часто неявно). Поэтому для корректности ссылок нам нужно уметь взаимно однозначно переписывать коды объектов в наборы слов из E^{Λ}, т.е. нужно научиться писать *1-коды* объектов естественного языка. Вначале пометим левый край ленты 011. Затем перекодируем входное слово конечного алфавита A: k -му символу алфавита сопоставим 01^{k+2}, где 1^t означает t единиц, записанных подряд. Соответствующим образом перепишем исходную программу A в программу P_1, которая работает в E^{Λ} с кодами символов из $A\cup\{\Lambda\}$ так же, как A с самими символами (Λ заменяем при этом 01).

Понятно, что если сделать всё достаточно аккуратно, то: 1) программа P_1 получается из A за полиномиальное от длины $|P|$ время; 2) когда время работы машины A полиномиальное (от длины входа), то и у P_1 оно такое же; 3) машина A распознаёт (допускает) какой-то вход тогда и только тогда, когда машина P_1 распознаёт (допускает) его код.

Кодирование машинных программ и формул сигнатуры булевых алгебр назовём *кодированием, по Куку и Карпу,* если оно осуществляется по записи формул и программ в естественном языке за полиномиальное время от их длины, и его однозначное декодирование осуществимо также за полиномиальное время.

Описанное выше кодирование наборами слов алфавита E^{Λ} очевидно является Кук-Карп кодированием, так как между словами кодирующего набора не может быть более одной пустой ячейки, что позволяет быстро сделать однозначное декодирование.

Для формул зафиксируем следующий алфавит («естественный язык»): а) сигнатурные символы $\cup,\cap,C,0,1$ и знак равенства $\approx$; б) латинские буквы f,q,x,y,z,u,v,w,g,h,d — для указания видов предметных переменных; в) арабские цифры и запятая — для записи индексов; г) логические связки $\wedge,\vee,\neg,\to$; д) знаки кванторов $\forall,\exists$; е) вспомогательные символы (,).

Приоритет связок и операций или его отсутствие не имеют значения, так как разница в длине формул в этих случаях получается линейной.

Машинные программы с рабочим алфавитом $E^{\Lambda}=\{1;0\}$ будем записывать символами $q,R,L,\to$ и арабскими цифрами. Если i — набор слов алфавита E^{Λ} (*1*-код), который кодирует программу некоторой машины, то эту программу обозначим как P_i.

Если x и y – *1*-коды каких-то объектов, то через $\langle x,y\rangle$ обозначается набор слов алфавита E^{Λ}, получающийся из последовательности x, y заменой 0 на 10, 1 на 11 и запятой — 01, соответственно.

3 Некоторые напоминания и основная теорема.

3.1 В [2; 204] даётся одно из эквивалентных определений полиномиально-ограниченной иерархии языков. Вначале полагается: $\Sigma_0^P=\Pi_0^P=P$ — класс языков, распознаваемых за полиномиальное время. Затем определяется, что язык L принадлежит классу Σ_{k+1}^P тогда и только тогда, когда существуют такие многочлен $p(n)$ и вычисляемый за полиномиальное время предикат $R(x,y_1,...,y_{k+1})$, что

$$x\in L\Leftrightarrow\exists y_1\forall y_2\exists y_3...Qy_{k+1}\ \&_{j=1}^{k+1}[|y_j|\le p(|x|)\ \&\ R(x,y_1,...,y_{k+1})], \qquad (1)$$

где Q — это $\exists$, если k чётное, и — $\forall$, если k нечётное. Здесь и всюду далее $|x|$ — длина слова x. При определении класса Π_{k+1}^P кванторы $\exists$ и $\forall$ меняются местами. В [2; 208] отмечается следующее свойство этой иерархии: если для некоторого $k\ge1$ верно $\Sigma_k^P=\Pi_k^P$, то для всех $j\ge k$ имеет место $\Sigma_j^P=\Pi_j^P$; и если $\Sigma_0^P=\Sigma_1^P$, то $\Sigma_j^P=\Pi_j^P=\Sigma_0^P$ для каждого j.

3.2 В [3; 277] доказывается, что *язык* $E=\{\langle i,X\rangle\mid$ *— машина с кодом программы* i *допускает цепочку* X *не более чем за* $\exp(2,|X|)$ *шагов}* *не лежит в классе P.*

Для всякого множества B обозначим как $\Sigma_k^{P,B}$ — класс языков, определяемых условиями вида (1), в которых предикат $R(x,y_1,...,y_k)$ вычислим за полиномиальное время машинами с оракулом B. Верен релятивизованный аналог теоремы 6.1.2 из [3; 277]:

Лемма. *Для всякого оракула* B *язык* $AK^B = \{\langle i, X\rangle \mid$ *машина с кодом программы* i *и оракулом* B *допускает 1-код* X *менее чем за* $\exp(2,|X|)$ *шагов}* *не лежит в классе* $P^B = \Sigma_0^{P,B}$.

Доказательство. Предположим, что этот язык принадлежит классу P^B. Тогда *язык* $L^B = \{i \mid$ *машина с кодом программы* i *и оракулом* B *не допускает 1-код* i *менее чем за* $\exp(2,|i|)$ *шагов}* также лежит в этом классе. Это означает, что имеется некоторая машина P_l^B, которая распознаёт этот язык за время, ограниченное подходящим многочленом $f(|i|)$. Каков будет её ответ, когда она встретится со своим *l*-кодом *l*?

Если ответ отрицательный, то это означает, по определению языка L^B, что машина P_l^B допускает этот *l*-код. Противоречие.

Положительный ответ означает, что машина P_l^B отвергает этот вход или работает на нём не менее $\exp(2,|l|)$ шагов. Тогда при достаточно длинном *l*-коде l имеем $f(l) < \exp(2,|l|)$ (код машины можно сделать сколько угодно большим, например, вводя неисполняемые команды). Полученное неравенство противоречит предположению об ограниченности вычислений значением $f(|l|)$, и завершает доказательство.

3.3 Идея доказательства основной теоремы почерпнута из [4].

Теорема [5; 75]. *По всякому входу* X *на ленте машины и любому коду машинной программы* i *можно эффективно построить формулу* $\theta(X,i)$ *сигнатуры* $\langle \cup, \cap, C, 0, 1\rangle$, *обладающую свойствами*:

а) формула $\theta(X,i)$ *строится за время* $g(|X|)+|i|)$ *для некоторого многочлена* g, *фиксированного для всех* X *и* i;

б) формула $\theta(X,i)$ *истинна на булевой алгебре* B *из двух элементов тогда и только тогда, когда машина с кодом программы* i *допускает вход* X, *сделав менее* $\exp(2,|X|)$ *шагов.*

Подробное доказательство теоремы дано в [6; 226–237].

Следствие. *Класс* P *— собственный подкласс классов* Σ_1^P, Π_1^P, Σ_2^P *и* Π_2^P.

Доказательство. Из построения формулы $\theta(X,i)$, приведённой в [5; 77–75] и [6; 227–231] и которое мы кратко повторим в разделах 4–5, хорошо видно, что эта формула имеет вид $\forall\exists$-формулы (п. 5.4). Таким образом, теорема утверждает, что языки E и AK полиномиально трансформируются к языку Π_2^0-формул, истинных на булевой алгебре B, а тем самым эти языки полиномиально редуцируются к языку C_2 Кук-Карп кодов таких формул. Как известно, язык C_2 полон в классе Π_2^P относительно этой сводимости [2; 207]. Следовательно, языки E и AK также входят в класс Π_2^P. Однако из [3; 277] и леммы следует, что эти языки не лежат в классе P. Отсюда и отмеченного выше свойства полиномиально-ограниченной иерархии и вытекает следствие.

3.4 Обсудим возможности релятивизации. Известно, что существует такой оракул A, для которого $P^A = \Sigma_1^{P,A}$ [2; 230 и 7; 433]; и по этой причине вся полиномиально-ограниченная иерархия для этого оракула сжимается до одного класса P^A.

Лемма прямо утверждает, что некоторые стадии доказательства следствия — релятивизуемые. Однако этап построения формулы $\theta(X,i)$ таковым не является. Действительно, в классической теории вычислимости для построения подобной формулы арифметики Пеано, которая описывает работу машины с не рекурсивным оракулом A, в арифметику необходимо добавить предикат, выделяющий A [8]. А какой неформульный предикат можно добавить в булеву алгебру из двух элементов, оба из которых и без того уже выделены константными символами? Имеются и другие детали построения формулы $\theta(X,i)$, мешающие написать её для машин с оракулом. Это описано в [5; 77].

4 Построение формулы $\theta(X,i)$. **Обозначения и смысл основных переменных**

4.1 Для лучшего восприятия договоримся о следующих сокращениях при записи формул:

1) операцию $\cap$ будем записывать и в виде умножения: $x \cap y = x \cdot y = xy$; 2) считаем, что $\cap$ и $\wedge$ (иногда записываемая как $\&$) связывают сильнее, чем $\cup$ и $\vee, \to$; 3) наряду с сигнатурными операциями используется ещё сложение по модулю два: $x \oplus y = x \cdot Cy \cup Cx \cdot y$; 4) $x < y = x \approx 0 \wedge y \approx 1$; 5) $x^\alpha = Cx \cdot C\alpha \cup x \cdot \alpha$, т.е. $x^0 = Cx$ и $x^1 = x$; 6) кроме обычных скобок в длинных формулах приме-

няются также квадратные и фигурные; 7) $(\alpha_0,\ldots,\alpha_m)<(\beta_0,\ldots,\beta_m)$ — сравнение наборов в лексикографическом порядке: $\alpha_0<\beta_0\vee\{\alpha_0=\beta_0\wedge(\alpha_1<\beta_1\vee[\alpha_1=\beta_1\wedge(\alpha_2<\beta_2\vee\{\alpha_2=\beta_2\wedge(\alpha_3<\beta_3\vee\ldots)\})])\}$.

При фиксированной длине m упорядоченного набора $(x_0,\ldots,x_{m-1})$ запись $\bar{x}$ может означать и сам набор, и терм вида $x_0\cdot\ldots\cdot x_{m-1}$, это всегда будет понятно из контекста. Естественно, что $\bar{x}^{\bar{\alpha}}=x_1^{\alpha_1}\cdot\ldots\cdot x_{m-1}^{\alpha_{m-1}}$, а $C\bar{x}=Cx_0\cdot\ldots\cdot Cx_{m-1}$. Наборы переменных с двумя индексами будут встречаться только в таком виде, когда первый индекс фиксирован, например, $(u_{k,0},\ldots,u_{k,m-1})$, его обозначим как $\bar{u}_k$. Термы вида $\bar{x}^{\bar{\alpha}}$ удобны тем, что формула $\bar{x}^{\bar{\alpha}}\approx 1$ эквивалентна системе равенств: $x_0,\ldots,x_{m-1}\approx\alpha_{m-1}$.

Если $(\bar{\gamma})_2=(\gamma_1,\ldots,\gamma_m)_2$ — двоичная запись натурального числа t, то $t+1=(\bar{\gamma})_2+1=$ $=(\gamma_1\oplus\gamma_2\cdot\ldots\cdot\gamma_{m-1}\cdot\gamma_m,\ldots,\gamma_{m-2}\oplus\gamma_{m-1}\cdot\gamma_m,\gamma_{m-1}\oplus\gamma_m,\gamma_m\oplus 1)_2$; при этом число $t-1$ выражается в виде $(\bar{\gamma})_2-1=(\gamma_1\oplus C\gamma_2\cdot\ldots\cdot C\gamma_{m-1}\cdot C\gamma_m,\ldots,\gamma_{m-2}\oplus C\gamma_{m-1}\cdot C\gamma_m,\gamma_{m-1}\oplus C\gamma_m,\gamma_m\oplus 1)_2$. Двоичную запись натурального числа t обозначим как $(t)_2$, а $\bar{\alpha}<\bar{\beta}$ равносильно $(\bar{\alpha})_2<(\bar{\beta})_2$.

4.2 Для моделирования работы машины Тьюринга на входе X за первые $\exp(2,|X|)-1$ достаточно уметь описывать её действия на полосе шириной в $T=\exp(2,m)$ клеток, где $m=|X|$. Поэтому протокол её работы (последовательность мгновенных описаний или конфигураций) можно представлять себе в виде нескольких прозрачных листов, на которых расчерчены таблицы из T строк и столбцов. Строка с номером t одного из листов отображает содержимое ленты после t шагов работы. Номер n внутреннего состояния, в котором находится машина, и символ α, обозреваемый в этот момент головкой, образуют маркер (α,n) листа, на котором записана строка, отражающая данную конфигурацию на ленте. Точнее, каждой команде $q_n\alpha\to\ldots$ соответствуют две страницы разного *цвета* с маркером (α,n), в которых могут быть сделаны какие-то записи на некоторых строках того же цвета, что и у номера строки. Для некоторых команд записи делаются на обеих страницах. Номер клетки, осматриваемой головкой на этом шаге, показывается значением специального счётчика.

В каждой строке этой таблицы-параллелепипеда, представляющей протокол действий машины, единицы могут встретиться только на одном листе, на других листах в этой строке хотелось бы поставить нули. Но это влечёт технические затруднения, поэтому там ставить ничего не будем.

4.3 Чётные и нечётные шаги выполнения программы описываются по-разному, это выглядит как записи разного *цвета*, и для этого применяются переменные двух цветов: f_0 и f_1. Эта «разноцветность» помогает «выбрать» нужный шаг (см. [5; 78–81] и [6; 232–235]).

Цвет записи согласован с чётностью номера $t=(\bar{\gamma})_2$ строки (шага, на котором возникла данная конфигурация): номер строки кодируется для чётных шагов посредством $\bar{y}_0^{\bar{\gamma}}=y_{0,0}^{\gamma_0}\ldots y_{0,m-1}^{\gamma_{m-1}}$, а термы вида $\bar{y}_1^{\bar{\gamma}}=y_{1,0}^{\gamma_0}\ldots y_{1,m-1}^{\gamma_{m-1}}$ применяются для описания нечётных шагов.

Однако номера ячеек (номера столбцов) на всех строках кодируются посредством термов из переменных одного цвета — $x_0,\ldots,x_{m-1}$ вне зависимости от чётности номера описываемой строки: терм $\bar{x}^{\bar{\alpha}}=x_0^{\alpha_0}\ldots x_{m-1}^{\alpha_{m-1}}$ задаёт номер $(\bar{\alpha})_2$ нужной ячейки в рассматриваемой строке.

Но маркер листа и указатель положения головки — опять разноцветные в соответствии с цветом записи. Таким образом, каждой команде $q_n\alpha\to\ldots$ машины соответствуют две страницы разного цвета с маркером (α,n). Там могут быть сделаны какие-то записи в некоторых строках, общее количество записанных строк равно числу выполнения данной команды. Первоначальная конфигурация на ленте записывается на странице с маркером $(0,1)$. Если $q_0,q_1,\ldots,q_U$ — все внутренние состояния машины, то достаточно $4\exp(2,r)$ листов, где $r=\min\{S\mid\exp(2,S)\ge U+1\}$.

Поэтому номер $n=(\bar{\delta})_2$ внутреннего состояния q_n и обозреваемый головкой символ α вместе с номером $(\bar{\beta})_2$ той клетки, где он стоит (маркер страницы вместе с указателем положения головки),

представляется как $d_0^{\alpha}\ \bar{q}_0^{\bar{\delta}}\ \bar{z}_0^{\bar{\beta}} = d_0^{\alpha}\ q_{0,0}^{\delta_0}...q_{0,r-1}^{\delta_{r-1}}\ z_{0,0}^{\beta_0}...z_{0,m-1}^{\beta_{m-1}}$ для чётных шагов, а на нечётных строках используется другой цвет — первые индексы переменных равны *1*.

В этом пункте введены обозначения вида $\bar{x}$, $\bar{z}_k$, $\bar{y}_j$ и $\bar{q}_s$ для наборов разной длины, но это не приведёт к путанице, так как у наборов из «иксов», «игреков» и «зетов» длина всегда будет — m, а у набора из «кю» — r. Наборы констант или других переменных могут быть тоже разной длины, но такой набор будет всегда однозначно привязан к одному из этих.

5 Окончание построения формулы $\theta(X,i)$. Непосредственно перед её написанием к программе дописываются команды холостого хода: $q_n\alpha \to q_n\alpha$, где $n \in \{0,2\}$, выполняя которые, машина работает, не меняя конфигурацию на ленте.

Фрагменты формулы $\theta(X,i)$ строятся с учётом соглашений разделов 1–2 и со свободными переменными $f_i, q_{e,r}, x_{j,k}, y_{s,k}, z_{t,n}, d_e$ и $v_{n,r}$, а в конце построения по ним берётся квантор всеобщности.

5.1 Описание строк начинается с *таймера* (тоже двухцветного), т.е. конъюнктивного члена вида

$$\pi_e(\bar{\gamma} \to \alpha, \bar{\delta}, \bar{\xi}) \ = \ \bar{y}_e^{\bar{\gamma}} \approx 1 \to d_e^{\alpha}\ \bar{z}_e^{\bar{\delta}}\ \bar{p}_e^{\bar{\xi}} \approx 1,$$

где $e = 0,1$ обозначает чётность шага. Эти «часы» показывают не столько «время» (номер шага), сколько то, что после шага $t = (\bar{\gamma})_2$ осматривается ячейка с номером $(\bar{\xi})_2$ и пора исполнить команду с началом $q_n\alpha \to ...$, где $n = (\bar{\delta})_2$.

Отдельная ячейка описывается *квазиуравнением* (или *клаузой*):

$$\psi_k(\bar{\mu}, \bar{\eta} \to \varepsilon) \ = \ \bar{x}^{\bar{\mu}}\ \bar{y}_k^{\bar{\eta}} \approx 1 \to \ f_k \approx \varepsilon,$$

где $k = 0$ на чётных шагах и $k = 1$ — на нечётных. Их смысл прост: после шага $(\bar{\eta})_2$ в ячейке с номером $(\bar{\mu})_2$ стоит ε. Квазиуравнение равносильно подходящей элементарной дизъюнкции, и потому ложно только на единственном наборе значений входящих в его запись переменных. эти формулы можно называть и *хорновыми дизъюнктами*.

5.2 Описание начальной конфигурации на ленте производится для нулевой, т.е. чётной, строки:

$$\chi(0) = (C\bar{y}_0 \approx 1 \to Cd_0 \cdot Cq_{0,0} \cdot ... \, Cq_{0,r-2} \cdot q_{0,r-1} \cdot C\bar{z}_0 \approx 1) \wedge \chi_1(0) \wedge \chi_2(0), \text{ где } \chi_1(0) =$$

$$= \underset{\alpha}{\&} (\bar{x}^{\bar{\alpha}} \cdot C\bar{y}_0 \approx 1 \to f_0 \approx 1) \wedge \underset{\bar{\beta}}{\&} (\bar{x}_0^{\bar{\beta}} \cdot C\bar{y}_0 \approx 1 \to f_0 \approx 0) = \underset{\alpha}{\&}\ \psi_0(\bar{\alpha}, \bar{0} \to 1) \wedge \underset{\beta}{\&}\ \psi_0(\bar{\beta}, \bar{0} \to 0).$$

Фрагмент $\chi_1(0)$ кодирует состояние первых $m+1$ клеток строки номер 0, т.е. собственно саму входную цепочку x на ленте. Здесь первая конъюнкция $\underset{\alpha}{\&}(\bar{x}_0^{\bar{\alpha}}...)$ берётся по всем тем наборам $\bar{\alpha}$, что в ячейке с номером $(\bar{\alpha})_2 \le m$ стоит *1*, а другая $\underset{\bar{\beta}}{\&}(\bar{x}_0^{\bar{\beta}}...)$ — по всем таким наборам $\bar{\beta}$, что в ячейке с номером $(\bar{\beta})_2 < m$ стоит 0.

А $\pi_0(\bar{0} \to 0,(1)_2,\bar{0}) = C\bar{y}_0 \approx 1 \to Cd_0 \cdot Cq_{0,0} \cdot \ldots Cq_{0,r-2} \cdot q_{0,r-1} \cdot \cdot C\bar{z}_0 \approx 1$ означает: «таймер цвета 0 показывает готовность машины во время 0 исполнить команду с началом $q_1 0 \to \ldots$ (страница с маркером $(0,1)$), а её головка — в крайнем левом положении». Осталось записать, что в строке 0 стоят «пустышки», начиная с ячейки с номером $m+1 = (\bar{\gamma})_2 + 1$:

$$\chi_2(0) \ = \ \forall \bar{u}\,(\bar{u} < \bar{\gamma} \to (\bar{x}_0^{\bar{u}}\ C\bar{y}_0 \approx 1 \to f_0 \approx 0)) = \forall \bar{u}\,(\bar{u} < \bar{\gamma} \to \psi_0(\bar{u}, \bar{0} \to 0)).$$

5.3 Каждой команде $M(k) = q_n\alpha \to q_j\beta$ с номером k (при $\alpha \in \{0,1\}$, $\beta \in \{R,L,0,1\}$, в том числе и командам холостого хода, сопоставим две формулы, описывающие её работу после чётных (при $e = 0$) или нечётных (при $e = 1$) шагов:

$$\varphi_e(k) \ = \ \forall \bar{u}_k \{\pi_e((\bar{v})_2 + e \to \alpha, (n)_2, \bar{u}_k) \to \forall h_k\,(\varphi_{\text{поиск,e}}(k,\beta) \to [\varphi_{\text{запись,e}}(k,\beta) \wedge$$

$$\wedge \pi_{Ce}((\bar{v})_2 + (e+1)_2 \to h_k, (j)_2, \bar{u}_k(\beta))]\}.$$

Для краткости обозначений для индекса e употребляется сигнатурная операция взятия дополнения. Первая посылка формулы $\varphi_e(k)$ — таймер $\pi_e((\bar{v})_2 + e \to \alpha, (n)_2, \bar{u}_k)$ цвета e «выясняет» приме-

нимость команды $M(k)$ к конфигурации, возникшей перед шагом $(t+e)+1=(\bar{v})_2+(e+1)_2$: действительно ли машина находится во внутреннем состоянии q_n, а её головка видит символ α, если она наведена на клетку с номером $(\bar{u}_k)_2$. После этого формула $\varphi_{\text{поиск,е}}(k,\beta)$ «вычисляет» тот символ h_k, который будет осматриваться головкой на шаге $(t+e)+1$. При этом номер $\left(\bar{u}_k(\beta)\right)_2$ той ячейки, которую будет обозревать головка после выполнения команды $M(k)$, определяется по метасимволу $\beta\in\{R,L,0,1\}$ и набору $\bar{u}_k$ следующим образом:

$$\bar{u}_k(0)=\bar{u}_k(1)=\bar{u}_k,\ \left(\bar{u}_k(R)\right)_2=\left(\bar{u}_k\right)_2+1 \text{ и } \left(\bar{u}_k(L)\right)_2=\left(\bar{u}_k\right)_2-1.$$

При $\beta\in\{R,L\}$ формула $\varphi_{\text{поиск,е}}(k,\beta)$ «считывает» тот символ h_k, который «записан» в соседней (справа или слева в зависимости от β) с обозреваемой ячейкой после шага $t+e$:

$$\varphi_{\text{поиск,е}}(k,\beta)=\left(\bar{x}^{\bar{u}_k(\beta)}\cdot\bar{y}_e^{\bar{v}+e}\approx 1\to f_e\approx h_k\right)=\psi_e\left(\bar{u}_k(\beta),\bar{v}+e\to h_k\right).$$

Формула $\varphi_{\text{запись,е}}(k,\beta)$ при этих β «копирует» строку с номером $t+e$ с (α,n)-ой страницы на (h_k,j)-ую и помещает её там, в строке с номером $(t+e)+1$, но при этом запись делается другим цветом:

$$\varphi_{\text{запись,е}}(k,\beta)=\forall g_k\ \forall\bar{w}_k\ ((\bar{x}^{\bar{w}_k}\bar{y}_e^{\bar{v}+e}\approx 1\to f_e\approx g_k)\ \to\ (\bar{x}^{\bar{w}_k}\bar{y}_{Ce}^{(\bar{v})_2+(e+1)_2}\approx 1\to f_{Ce}\approx g_k))=$$
$$=\forall g_k\ \forall\bar{w}_k\ (\psi_e(\bar{w}_k,\bar{v}+e\to g_k)\to\psi_{Ce}(\bar{w}_k,(\bar{v})_2+(1+e)_2\to g_k)).$$

При $\beta\in\{0,1\}$ искать ничего не нужно: $\varphi_{\text{поиск,е}}(k,\beta)=h_k\approx\beta$. Формула $\varphi_{\text{запись,е}}(k,\beta)$ при этом «ставит» символ β, цвета Ce, в ячейку $(\bar{u}_k)_2$ строки с номером $(t+e)+1$ на (h_k,j)-ой странице. Остальные клетки этой строки точно так же, как и раньше с изменением цветности, «копируются» со строки $t+e$ на (α,n)-ой странице:

$$\varphi_{\text{запись,е}}(k,\beta)\ =\ \psi_{Ce}\left(\bar{u}_k,\left(\bar{v}\right)_2+(e+1)_2\to\beta\right)\wedge\forall\bar{w}_k\ \forall g_k\,(\neg\bar{w}_k\approx\bar{u}_k\to$$
$$\left[\psi_e\left(\bar{w}_k,\bar{v}+e\to g_k\right)\to\psi_{Ce}\left(\bar{w}_k,(\bar{v})_2+(e+1)_2\to g_k\right)\right]).$$

Таймер $\pi_{Ce}((\bar{v})_2+(e+1)_2\to h_k,(j)_2,\bar{u}_k(\beta))$ цвета Ce, стоящий в заключении формулы $\varphi_e(k)$, «переводит стрелки на время следующего шага, а весь механизм подготавливает к его выполнению».

5.4 Утверждение о том, что машина допускает вход X менее чем за $\exp(2,m)$ шагов запишем формулой $\chi(\omega)$. Она означает, что машина пришла в допускающее внутреннее состояние q_0, её головка осматривает при этом крайнюю левую ячейку (с номером *0*), в которой записан *0*, и это случилось на шаге, меньшем $\exp(2,m)$. Ввиду наличия команд холостого хода (начало раздела 5), это осуществляется «проверкой» таймера на листе с маркером $(0,\bar{0})$ только в последней строке с нечётным номером $(\bar{1})_2$: $\chi(\omega)=\pi_1(\bar{1}\to 0,\bar{0},\bar{0})$.

Вся формула $\theta(X,i)$ имеет вид (здесь N — число команд машины с кодом i, вместе с добавленными командами холостого хода):

$$\theta(X,i)\ =\ \forall\bar{x},\ \bar{y}_0,\ \bar{y}_1,\ \bar{z}_0,\ \bar{z}_1,\ \bar{q}_0,\ \bar{q}_1\ \forall d_0,d_1,f_0,f_1\,[\,\chi(0)\,\&\,R(i)\to\chi(\omega)\,],$$

где

$$R(i)=\exists\bar{v}\,\{\,v_{m-1}\approx 0\wedge\left\lfloor\bar{v}\right\rfloor\approx\left\lfloor\bar{y}_1\right\rfloor\wedge\ \&_{0\le k\le N-1}\varphi_0(k)\wedge[\,(\bar{v})_2+1<\bar{1}\to\ \&_{0\le k\le N-1}\varphi_1(k)\,]\}.$$

Здесь $\left\lfloor\bar{v}\right\rfloor=(v_0,\ldots,\ v_{m-2})$ — усечение набора $\bar{v}$. Таким образом, условие $v_{m-1}\approx 0\wedge\left\lfloor\bar{v}\right\rfloor\approx\left\lfloor\bar{y}_1\right\rfloor$ однозначно задаёт чётное число $(\bar{v})_2$, не большее $(\bar{y}_1)_2$. Обратим внимание на то, что формулы $\varphi_1(k)$ предполагаются записанными как в п. 5.3, т.е. их клаузы и таймеры цвета *1* в посылках содержат $\bar{y}_1^{(\bar{v})_2+1}$, а квазиуравнения и таймеры цвета — $0-\bar{y}_0^{(\bar{v})_2+2}$.

По построению формулы $\chi(0)$, $\varphi_e(k)$ универсальные, так как в их записи кванторы всеобщности встречаются в заключениях импликаций, перед одним из конъюнктивных членов и перед самими

формулами $\varphi_e(k)$. Формула $\chi(\omega)$ бескванторная. При вынесении кванторов $\exists\bar{v}$ и всеобщности из посылки формулы $\theta(X,i)$ — формулы $\chi(0)\,\&\,R(i)$ они поменяются на кванторы всеобщности и существования соответственно и в результате формула $\theta(X,i)$ приобретёт вид $\forall\exists$-формулы.

References

1 *Aho A., Hopcroft J., Ullman J.* The design and analysis of computer algorithms. — Moscow: Mir, 1976. — P. 414.

2 *Garey M.R., Johnson D.S.* Computers and Intractability task: A Guide to the Theory of NP-completeness. — New York: Freeman, 1979. — P. 416.

3 *Lewis H.R., Papadimitriou C.H.* Elements of the theory of computation. — New Jersey 07458: Prentice Hall Inc., Upper Saddle River, 1998. — P. 356.

4 *Rabin M.O.* Solvable theories // Handbook of a mathematical logic. Ed. J.Barwise. — Amsterdam: North-Holland Publ. Company, 1977. — P. 104.

5 *Latkin I.V.* The computational expressiveness of theory of the two-elements Boolean algebra // Computability and Models Proc. of the Intern. Sc. Conf. — Ust-Kamenogorsk: EKSTU, 2010. — P. 73–82 (in Russian).

6 *Latkin I.V.* Recognition complexity of theories and their computational expressivity // Algebra and Logic. — 2012. — Vol. 51. — № 2. — P. 216–238.

7 *Baker Th., Gill J., Solovay R.* Relativizations of the P=? NP question // SIAM J.Comput. — 1975. — Vol. 4 — P. 433.

8 *Rogers H.Jr.* Theory of recursive functions and effective computability. — New York: McGraw-Hill Book Company, 1967. — P. 400–406.

И.В.Латкин

Тілдердің полиномды-шектелген иерархиясының төменгі кластарының теңсіздігі

Ең төменгі, яғни, тілдердің полиномды-шектелген иерархия класының ең алғашқысы, өте кішілері қарастырылды. Автордың бұрынғы жүргізілген дәлелдеулерінің нәтижесінде бульдік алгебраның формулаларының экспоненциалды ұзындықтарын есептеуді үлгілеу мүмкіндігін пайдаланып, осы иерархияның тым болмағанда ең кіші екі класы үшін айырмашылық туралы қорытынды жасалды. Басқа кластардың беттесетіндігі туралы сұрақ ашық қалды.

I.V.Latkin

The inequality of the low classes of polynomial-bonded hierarchy of languages

We study the low, i.e., the first, smallest, classes of the polynomial-bounded hierarchy of languages. The author has formerly constructed the simulating by formulae of Boolean algebras for the computations which have the exponential length. It follows from this that the hierarchy contains the two essentially different classes; these classes are the smallest in the hierarchy. The question about the coincidences of other classes of this hierarchy remains open for the time being.