

О синтаксическом определении класса языков, распознаваемых недетерминированными машинами Тьюринга на логарифмической памяти

Д.А. Носов
ООО «Яндекс», Москва
dmitrytyr@gmail.com

Аннотация. В работе М. А. Тайцлина и А. П. Столбоушкина было введено понятие недетерминированной программы, и определен класс языков, распознаваемых недетерминированными программами. Доказана теорема, свидетельствующая о близости этого класса, и \mathbf{NL} . Мы исследуем класс языков, определяемый в некоторой модификации указанной вычислительной модели, и доказываем, что рассматриваемый класс языков равен \mathbf{NL} .

Ключевые слова: машина Тьюринга; недетерминированные программы; класс сложности; логарифмическая память; сложность вычислений; универсальная алгебра.

1. Введение

Мы предполагаем, что читатель знаком с понятиями машины Тьюринга, входа, времени работы, и памяти, в которой работает машина Тьюринга, а также со сложностными классами \mathbf{P} и \mathbf{PSPACE} . Говорят, что язык L принадлежит сложностному классу \mathbf{NL} , если существует недетерминированная машина Тьюринга, которая распознает язык L в логарифмической от длины входа памяти.

Изначально проблема, совпадают ли сложностные классы \mathbf{NL} и \mathbf{P} , была сформулирована в работах С. Кука [1], [2], [3]. В этих работах доказаны утверждения, называемые, соответственно, первой, и второй теоремой Кука.

В обеих теоремах Кука рассматривается машина Тьюринга, работающая на входе длины n , и $S(n)$ — функция, которая мажорирует двоичный логарифм от n .

Первая теорема Кука утверждает, что класс языков, распознаваемых на детерминированной машине Тьюринга за время $2^{cS(n)}$, где c — любая

положительная константа, распознается также детерминированными многоленточными машинами Тьюринга со стэком, память которых ограничена $S(n)$. Эта теорема не доказывает совпадение классов \mathbf{NL} и \mathbf{P} , именно из-за наличия стэка неограниченного размера. Не получается доказать никаких соотношений между классами языков, распознаваемых машинами Тьюринга со стэком, классов \mathbf{NL} и \mathbf{P} .

Вторая теорема Кука, утверждает, что язык, который распознается недетерминированной односторонней многоленточной машиной Тьюринга со стэком, память которой ограничена $S(n)$, распознается также недетерминированной односторонней многоленточной машиной Тьюринга без стэка, за время $2^{cS(n)}$, где константа c не зависит от длины входа машины Тьюринга.

Иммерман доказал, что $\mathbf{NL} = \mathbf{coNL}$ (см. [4]).

Теорема Савича утверждает, что класс $\mathbf{NL} \subseteq \mathbf{L}^2$, где \mathbf{L}^2 — класс языков, распознаваемых машинами Тьюринга, работающими на квадратичной от логарифма длины входа памяти (см. [5]).

Подробнее о сложностных классах можно узнать в [6] и [7].

Определение класса недетерминированных программ PR^+ было дано в работе [8]. Авторы доказали, что класс языков, распознаваемых недетерминированными программами из PR^+ совпадает с $\mathbf{NL} \cap L_M$, где L_M — все языки, являющиеся кодами универсальных алгебр. Насколько нам известно, полное доказательство этой теоремы не опубликовано. Данная теорема свидетельствует о том, что сложностной класс \mathbf{NL} близок к классу языков, распознаваемых недетерминированными программами из PR^+ . Однако вопрос, можно ли модифицировать модель вычислений основанную на недетерминированных программах таким образом, чтобы сложностной класс \mathbf{NL} совпадал с классом языков, распознаваемых недетерминированными программами, оставался открытым. В данной работе на этот вопрос дается положительный ответ. В классе PR^+ выделяется специальный подкласс недетерминированных программ. Для этого подкласса дается новое определение языка, распознаваемого недетерминированной программой. Пусть \mathbf{Z}_Ψ^+ — класс всех таких языков. В данной работе доказано, что $\mathbf{NL} = \mathbf{Z}_\Psi^+$.

Автор выражает глубокую благодарность научному руководителю, Варновскому Н. П., благодаря которому эта работа обрела законченный облик, а также рецензенту Анохину М.И. за ряд ценных замечаний.

2. Недетерминированные программы

2.1. Сигнатура недетерминированных программ

Каждая недетерминированная программа из класса PR^+ работает на некоторой конечной стандартно заданной универсальной алгебре A . Носителем данной универсальной алгебры является начальный отрезок натуральных чисел $A = \{0, 1, \dots, LAST\}$. Сигнатура рассматриваемых универсальных алгебр имеет вид $\Omega = \langle c_1, c_2, \dots, c_{n_1}, 0, LAST, \dots$

$\dots, f_1^{k_1}, f_2^{k_2}, \dots, f_{n_2}^{k_{n_2}} \rangle$, где c_1, c_2, \dots, c_{n_1} — константы, каждая из функций

$f_j^{k_j}$ — k_j -арная операция на A . Заметим, что в сигнатуре всегда есть

унарная операция $'$, возвращающая следующий элемент универсальной алгебры. Другими словами, на носителе универсальной алгебры задана некоторая нумерация κ . 0 — константа, всегда равная первому (по κ) элементу; $LAST$ — константа, всегда равная последнему (по κ) элементу; унарная операция $'$ всегда возвращает следующий элемент универсальной алгебры. Так как $LAST$ — последний элемент, то $LAST'$ всегда равен $LAST$.

Сигнатура каждой недетерминированной программы Λ включает в себя Ω , а также конечное множество X — набор используемых в недетерминированной программе переменных. $\Lambda = \langle \Omega, X \rangle$.

2.2. Синтаксис класса недетерминированных программ PR^+

Недетерминированные программы класса PR^+ сигнатуры Λ задаются по индукции.

- $\mathbf{x = f(x_1, \dots, x_k)}$; — программа, если x, x_1, \dots, x_k — переменные из X , $f : |A|^k \rightarrow A$; f из Ω . Это оператор присваивания.

- $\mathbf{x = x + 1}$; — программа, если x — переменная из X . Это оператор инкремента.

- $\mathbf{x = y}$; — программа, если x, y — переменные из X . Это оператор присваивания.

- $\mathbf{x = c}$; — программа, если c — константа из Ω , x — переменная из X . Это оператор присваивания.

- \mathbf{Y} — программа, если Y — тест.

- $\mathbf{P_1; P_2}$ — программа, если P_1 и P_2 — программы. Это последовательная композиция программ.

- $\mathbf{if(Y) S endif}$ — программа, если Y — тест, S — программа. Это ветвление.

- $\mathbf{P_1 \cup P_2}$ — программа, если P_1, P_2 — программы. Это параллельная композиция программ.

- $\mathbf{while(Y) S endwhile}$ — программа, если Y — тест, S — программа. Это цикл.

Тесты:

- $\mathbf{x = y?}$ — программа, если x, y — переменные из X . Это тест.

- $\mathbf{x \neq y?}$ — программа, если x, y — переменные из X . Это тест.

Количество переменных в программе конечно, фиксировано, и равно мощности множества X .

Заметим, что $\mathbf{x = x + 1}$; — это оператор недетерминированной программы, который в результате выполнения меняет значение переменной x на x' . Семантика всех операторов будет рассмотрена ниже.

Определение 1 Для теста $\mathbf{x = y?}$, где x, y — переменные из X , $\mathbf{x \neq y?}$ — это обратный тест. Мы будем обозначать обратный тест к T через $!T$. Аналогично, для теста $\mathbf{x \neq y?}$, $\mathbf{x = y?}$ — это обратный тест.

Определение 2 Класс недетерминированных программ PR^+ — это все недетерминированные программы всех сигнатур.

2.3. Семантика недетерминированных программ

Определение 3 Путь вычислений — конечная последовательность операторов и тестов.

По каждой недетерминированной программе можно построить не более чем счетное множество путей вычислений, которое однозначно определяет работу этой недетерминированной программы.

Построение будем осуществлять по индукции.

Через $\alpha = \{\alpha_1, \alpha_2, \dots\}$ обозначим множество путей вычислений недетерминированной программы P_1 ; $\beta = \{\beta_1, \beta_2, \dots\}$ — это множество путей вычислений недетерминированной программы P_2 . α_i и β_i в данных обозначениях — пути вычислений.

Определение 4 Множество путей вычислений программы P .

• Программе P , состоящей из одного оператора присваивания, инкремента или теста, соответствует множество путей вычислений программы P , в котором ровно один путь вычислений, состоящий из одного этого оператора(теста).

• Программе P , являющейся последовательной композицией программ, $P = P_1; P_2$ соответствует множество путей вычислений, состоящее из всех путей вычислений вида $\alpha_i; \beta_j$, для каждого α_i из α и β_j из β . Множества путей вычислений α и β недетерминированных программ P_1 и P_2 уже построены по индукции. Через $\alpha_i; \beta_j$ мы обозначаем конкатенацию путей вычислений.

• Программе P , являющейся параллельной композицией программ, $P = P_1 \cup P_2$ соответствует множество путей вычислений являющееся объединением (в теоретико-множественном смысле) множеств α и β . Как и в предыдущем случае, множества путей вычислений α и β недетерминированных программ P_1 и P_2 уже построены по индукции.

• Программе P , являющейся условным оператором вида **if(Y) P₁ endif** соответствует объединение (в теоретико-множественном смысле) двух множеств путей вычислений:

- $Y; \alpha_i$ — конкатенация теста Y с каждым α_i из α .
- $!Y; \alpha_i$ — конкатенация теста, обратного к Y с каждым α_i из α .

• Программе P , являющейся оператором цикла **while(Y) P₁ endwhile** соответствует счетное множество путей вычислений:

- $!Y$; — тест, обратный к Y .
- $Y; \alpha_i; !Y$ — конкатенация теста Y с каждым α_i из α , далее тест, обратный к Y .
- $Y; \alpha_{i_1}; Y; \alpha_{i_2}; !Y$ — дважды повторенная конкатенация теста Y с каждым α_{i_1} , и с каждым α_{i_2} из α , далее тест, обратный к Y .
- $\{Y; \alpha_{i_k}; \}_j !Y$ — j раз повторенная конкатенация теста Y с каждым α_{i_k} из α , далее тест, обратный к Y . Пути вычисления α_{i_k} на разных

итерациях принимают все возможные значения из α независимо друг от друга.

В этом случае α — не более чем счетное по индукции, а значит, мы имеем счетное объединение не более чем счетных множеств, которое, также, не более чем счетно.

3. Класс Z^+

Мы рассматриваем недетерминированные программы произвольной сигнатуры $\Lambda = \langle \Omega, X \rangle$.

Определение 5 Вход недетерминированной программы сигнатуры $\Lambda = \langle \Omega, X \rangle$ — это универсальная алгебра A сигнатуры Ω .

Другими словами, вход недетерминированной программы — это конкретные значения всех констант и функций сигнатуры Λ недетерминированной программы.

Пусть $\alpha_i = (h_1, h_2, \dots, h_k)$ — это некоторый путь вычислений из множества путей вычислений программы P ; каждый из h_i — это оператор или тест; универсальная алгебра A — вход программы P .

Определение 6 Состояние пути вычислений α_i при работе на универсальной алгебре A — это функция $s(t, x)$, действующая в $|A|$. $t = 0, 1, 2 \dots k$ — это номер текущего оператора (теста), $x \in X$ — переменная.

• В начальный момент времени все переменные принимают значение 0, то есть $s(0, x) = 0$, для каждой переменной x .

• Пусть в момент времени t выполняется оператор $x = f(x_1, \dots, x_k)$. В этом случае $s(t+1, x) = f(s(t, x_1), \dots, s(t, x_k))$; $s(t+1, y) = s(t, y)$ для любой переменной y , не совпадающей с x ; значение функции f определяется универсальной алгеброй A .

- $?x = y$. В этом случае $s(t+1, x) = s(t, x)$ для любой переменной x .
- $?x \neq y$. В этом случае $s(t+1, x) = s(t, x)$ для любой переменной x .
- $x = y$. В этом случае $s(t+1, x) = s(t, y)$; $s(t+1, z) = s(t, z)$ для любой переменной z не совпадающей с x .

• $x = c$. В этом случае $s(t+1, x) = c$; $s(t+1, y) = s(t, y)$ для любой переменной y , не совпадающей с x . Значение константы c определяется универсальной алгеброй A .

• $x = x+1$. В этом случае $s(t+1, x) = s(t, x)'$. Данное определение корректно в силу того, что операция ' по определению есть в каждой сигнатуре. Соответственно, если $s(t, x)$ есть $LAST$, то $s(t+1, x) = s(t, x)$. В обоих случаях, $s(t+1, y) = s(t, y)$ для любой переменной y , не совпадающей с x .

Каждый из тестов в рассматриваемом пути вычислений, на входе A , может быть либо истинным, либо ложным.

Определение 7 Тест $?x = y$ истинен в рассматриваемом пути вычислений, на входе A , тогда и только тогда, когда $s(t, x) = s(t, y)$.

Определение 8 Тест $?x \neq y$ ложен в рассматриваемом пути вычислений, на входе A , тогда и только тогда, когда $s(t, x) = s(t, y)$.

Недетерминированная программа, также как и машина Тьюринга-распознаватель либо принимает вход, или не принимает.

Определение 9 Недетерминированная программа P принимает вход A , если существует путь вычислений α_i , из множества путей вычислений P , при работе которого на A , все тесты в α_i истинны.

Код универсальной алгебры A сигнатуры Ω — слово в конечном алфавите, однозначно соответствующее A .

Пусть мы кодируем универсальную алгебру A сигнатуры

$$\Omega = \langle c_1, c_2, \dots, c_{n_1}, 0, LAST, f_1^{k_1}, f_2^{k_2}, \dots, f_{n_2}^{k_{n_2}} \rangle.$$

Слово, кодирующее эту систему:

- разделитель #;
- N символов |, где $N = LAST + 1$ — количество элементов в носителе $|A|$ универсальной алгебры A ;
- разделитель #;
- значения констант c_1, c_2, \dots, c_{n_1} , в унарной системе счисления, разделенные специальным символом %;
- разделитель #;

• таблицы функций $f_1^{k_1}, f_2^{k_2}, \dots, f_{n_2}^{k_{n_2}}$, разделенные специальным символом %;

• разделитель #.

Таблица функции $f_i^{k_i}$ — это последовательность из N^{k_i+1} символов 0 и 1, где на месте $s = x + j_1 * N + j_2 * N^2 + \dots + j_{k_i} * N^{k_i}$ (где каждое из чисел x, j_1, \dots, j_{k_i} меньше N) стоит 1 тогда и только тогда, когда $f(j_1, \dots, j_{k_i}) = x$.

Определение 10 $L_{M\Omega}$ — это множество всех слов, кодирующих какую-либо универсальную алгебру сигнатуры Ω .

Определение 11 L_M — это множество, элементами которого являются все возможные подмножества всех множеств $L_{M\Omega}$, для всех возможных сигнатур Ω .

Определим класс языков, распознаваемых недетерминированными программами. Итоговый класс зависит от принятого кодирования универсальных алгебр.

Определение 12 Недетерминированная программа P сигнатуры $\Lambda = \langle \Omega, X \rangle$ распознает язык L в алфавите $\{0, \#, 1\}$, если выполнено следующее условие: $w \in L$ тогда и только тогда, когда

- w является кодом некоторой конкретной универсальной алгебры A сигнатуры Ω , то есть $w \in L_{M\Omega}$;
- недетерминированная программа P принимает вход A .

Определение 13 Класс языков Z^+ — это все языки, распознаваемые недетерминированными программами из класса PR^+ .

Справедлива следующая теорема, авторами которой являются М. А. Тайцлин и А. П. Столбоушкин, [8].

Теорема 14 $Z^+ = \mathbf{NL} \cap L_M$.

4. Класс Z_{Ψ}^+

В классе Z_{Ψ}^+ , в отличие от предыдущего класса, рассматриваются недетерминированные программы фиксированной сигнатуры. Всюду далее мы будем считать, что сигнатура $\Psi = \langle 0, LAST, a, b, err, ', f^{(1)} \rangle$. $\Lambda = \langle \Psi, X \rangle$. X , как и ранее конечное множество переменных недетерминированной программы; $0, LAST, a, b, err$ — константы.

Определение 15 Класс недетерминированных программ PR_{Ψ}^+ — это все недетерминированные программы сигнатуры $\Lambda = \langle \Psi, X \rangle$.

Определение 16 Универсальная алгебра A принадлежит классу универсальных алгебр $A(\Psi)$, если

- сигнатура A — это $\Psi = \langle 0, LAST, a, b, err, ', f^{(1)} \rangle$;
- носитель $A = \{err, a, b, 0, \dots, N\}$;
- $f(i) \in \{a, b\}$, если $i \in 0, \dots, N$;
- $f(i) = err$ в противном случае;
- значения констант err, a, b это, соответственно, элементы err, a, b носителя универсальной алгебры.

Определение 17 Код универсальной алгебры $A \in A(\Psi)$ — это слово w длины $N+1$ в алфавите $\{a, b\}$, такое, что на i -ой позиции в слове w стоит значение $f(i)$. N — это число из носителя $|A| = \{err, a, b, 0, \dots, N\}$.

Утверждение 18 Существует взаимно однозначное соответствие кодов $A(\Psi)$ и слов в алфавите $\{a, b\}$.

Доказательство. Очевидно следует из того, что каждая универсальная алгебра из $A(\Psi)$ однозначно характеризуется значениями своей функции f .

Утверждение 19 Множество кодов универсальных алгебр $A(\Psi)$ — это все слова в алфавите $\{a, b\}$.

Доказательство. Это утверждение очевидно, так как выше доказано, что существует взаимно однозначное соответствие кодов $A(\Psi)$ и слов в алфавите $\{a, b\}$.

Определение 20 Недетерминированная программа $P \in PR_{\Psi}^+$ распознает язык L , если $w \in L$ тогда и только тогда, когда

- w является кодом некоторой конкретной универсальной алгебры A из $A(\Psi)$;
- недетерминированная программа P принимает вход A из пункта 1.

Определение 21 Класс языков Z_{Ψ}^+ — это все языки, распознаваемые недетерминированными программами из класса PR_{Ψ}^+ .

Теорема 22 $Z_{\Psi}^+ = \mathbf{NL}$.

Доказательство. Всюду далее, без ограничения общности, будем считать, что рассматриваемая машина Тьюринга начинает работу в состоянии q_0 , и имеет два заключительных состояния — q_{acc} и q_{rej} . В случае, если машина Тьюринга принимает слово, записанное на входной ленте, то существует конечная последовательность команд, применяя которые машина перейдет в состояние q_{acc} . В противном случае машина Тьюринга заикливется, или переходит в q_{rej} . У нескольких команд машины Тьюринга может быть одинаковая левая часть, в этом случае на исполнение будет недетерминированно выбрана одна из этих команд.

В доказательстве мы не будем рассматривать случай, когда слово w пусто. В этом, и только в этом случае входная лента машины Тьюринга пуста. В этом, и только в этом случае носитель универсальной алгебры A состоит только из $\{err, a, b\}$; значения констант $0, LAST$ программы P равны err ; а также для любого i $f(i) = err$. Длины всех рабочих лент в этом случае ограничены константой.

1. Докажем, что $Z_{\Psi}^+ \subseteq \mathbf{NL}$.

Пусть $L \in Z_{\Psi}^+$, P — недетерминированная программа из класса PR_{Ψ}^+ , распознающая этот язык. Это значит, что для каждого слова $w \in L$ существует универсальная алгебра A из $A(\Psi)$, такая что w является кодом A , и P принимает вход A . Нам необходимо доказать, что $L \in \mathbf{NL}$. Для этого достаточно построить недетерминированную машину Тьюринга M , работающую в логарифмической памяти с одной входной и несколькими рабочими лентами, которая принимает все слова $w \in L$, и только их.

Количество рабочих лент машины M в данном доказательстве зависит от количества переменных в недетерминированной программе. Точное число рабочих лент:

- по одной рабочей ленте для каждой переменной недетерминированной программы,
- одна рабочая лента для выполнения оператора функционального присваивания,
- по одной ленте для каждой константы.

До начала работы машины M на входной ленте записано слово w , все рабочие ленты пусты. На всех лентах машины Тьюринга мы будем использовать заранее фиксированный алфавит.

Утверждение 23 *Значение любой переменной программы P может быть записано на одной рабочей ленте.*

Доказательство. Рассмотрим слово w длины $N + 1$, записанное на входной ленте машины M . Носитель универсальной алгебры $A = \{err, a, b, 0, \dots, N\}$. Каждая из переменных недетерминированной программы P может принимать значение от 0 до N , а также значения a, b, err . На входной ленте машины M записано ровно $N + 1$ символов, а длина рабочей ленты ограничена логарифмом от длины входа, следовательно, любое значение переменной недетерминированной программы может быть записано в двоичной системе счисления на любой рабочей ленте.

Пусть в рассматриваемой недетерминированной программе k переменных, тогда текущее состояние всех переменных может быть записано на k рабочих лентах машины Тьюринга.

До начала выполнения программы, все переменные принимают значение 0, соответственно на каждой рабочей ленте, отвечающей за соответствующую переменную, записано значение, соответствующее 0. В начале работы машина Тьюринга M запишет на специальные рабочие ленты значения констант недетерминированной программы, в том числе констант 0, $LAST$, err .

В ходе доказательства мы сначала построим машины Тьюринга, которые соответствуют операторам недетерминированной программы. Эти "базисные" машины изменяют состояние рабочих лент, и переходят в свое состояние q_{acc} , когда моделирование оператора недетерминированной программы завершено. Когда моделируются тесты недетерминированной программы, то соответствующая "базисная" машина Тьюринга переходит в q_{acc} , или в q_{rej} .

Далее мы рассмотрим, как построить машины Тьюринга, соответствующие более сложным программам, по индукции. Эти построения будут осуществляться, в основном, изменением состояния машины Тьюринга, и не будут менять ее рабочие ленты.

Следующее утверждение является базисом индукции по сложности программы.

Утверждение 24 *Если программа P состоит из одного оператора, то соответствующая "базисная" машина Тьюринга M существует.*

Доказательство. Рассмотрим различные программы из одного оператора.

- $P = x = f(y)$; — оператор присваивания. В этом случае, соответствующая машина M должна записать на ленту, соответствующую переменной x , значение функции f от y . Напомним, что слово, кодирующее функцию f записано на входной ленте. Для выполнения данного оператора нам понадобится дополнительная рабочая лента, которая, как и все ленты, ограничена логарифмом от длины входа. Работа данного оператора происходит следующим образом: машина M смещается по входной ленте, соответствующей данной функции, на y ячеек. Для этого используется дополнительная лента. Далее необходимо присвоить значение a или b , в зависимости от обозреваемого символа на входной ленте, переменной x . Для этого надо скопировать значение, соответствующее a или b , хранящееся на специальной рабочей ленте, на ленту соответствующую переменной x . После этого машина Тьюринга M переходит в заключительное состояние q_{acc} .

- $P = x = x + 1$; — оператор инкремента. В этом случае, соответствующая машина M должна прибавить единицу в двоичной системе счисления на ленте, соответствующей переменной x , и перейти в заключительное состояние q_{acc} .

- $P = x = y$; — оператор присваивания. В этом случае, соответствующая машина M должна скопировать ленту, соответствующую переменной y , на ленту, соответствующую переменной x , и перейти в заключительное состояние q_{acc} .

- $P = x = c$; — оператор присваивания. В этом случае, соответствующая машина M должна скопировать ленту, соответствующую константе c , на ленту, соответствующую переменной x , и перейти в заключительное состояние q_{acc} .

- $P = \mathbf{x} = \mathbf{y}?$ — положительный тест. В этом случае, соответствующая машина M должна удостовериться, что ленты, соответствующие переменным x и y совпадают. Иными словами, если ленты совпадают, соответствующая машина M переходит в состояние q_{acc} , а если не совпадают — то в q_{rej} .

- $P = \mathbf{x} \neq \mathbf{y}?$ — отрицательный тест. В этом случае, соответствующая машина M должна удостовериться, что ленты, соответствующие переменным x и y различны хотя бы в одном символе. Иными словами, если ленты совпадают соответствующая машина M переходит в состояние q_{rej} , а если не совпадают — то в q_{acc} .

Следующее утверждение является шагом индукции по сложности программы.

Утверждение 25 Если программа P является композицией программ, ветвлением, или циклом, и машины Тьюринга, моделирующие составные части программы P , существуют, то существует машина Тьюринга M , моделирующая программу P .

Доказательство.

Определение 26 Безусловный переход из состояния q_1 в состояние q_2 машины Тьюринга M — это переход из состояния q_1 в состояние q_2 , при котором не происходит движения головок машины Тьюринга по лентам, не происходит изменения рабочих лент, и который происходит при любых символах, которые читаются в текущий момент головками машины Тьюринга.

Введем условные обозначения: машину Тьюринга, соответствующую недетерминированной программе P_1 мы обозначим через M_1 ; машину Тьюринга, соответствующую недетерминированной программе P_2 мы обозначим через M_2 ; машину Тьюринга, соответствующую тесту Y мы обозначим через M_T ; результирующую машину Тьюринга мы будем обозначать через M . Здесь недетерминированные программы P_1 и P_2 — это составные части программы P .

- $P = \mathbf{P}_1; \mathbf{P}_2$ — последовательная композиция программ P_1 и P_2 . В этом случае результирующая машина M делает безусловный переход из q_{acc} машины Тьюринга M_1 в начальное состояние машины Тьюринга M_2 , и безусловный переход из q_{rej} машины Тьюринга M_1 в q_{rej} машины Тьюринга, соответствующей P_2 (M_2). Начальным состоянием результирующей машины M будет начальное состояние M_1 ; заключительными — заключительные состояния M_2 . Далее необходимо преобразовать полученную машину, чтобы в ней было одно входное состояние, и два заключительных — q_{acc} и q_{rej} .

- $P = \mathbf{if}(Y) \mathbf{P}_1 \mathbf{endif}$ — ветвление. Начальным состоянием результирующей машины M будет начальное состояние машины M_T . заключительными состояниями — заключительные состояния M_1 . Также в результирующей машине M есть безусловный переход из q_{acc} машины M_T в начальное состояние M_1 , и безусловный переход из q_{rej} машины M_T в q_{acc} машины M_1 .

- $P = \mathbf{P}_1 \cup \mathbf{P}_2$ — параллельная композиция программ. Введем новые состояния — начальное и два заключительных для машины M . В результирующей машине M будут следующие безусловные переходы:

- из начального состояния машины M в начальное состояние M_1 ;
- из начального состояния машины M в начальное состояние M_2 ;
- из q_{acc} машины M_1 в q_{acc} машины M ;
- из q_{acc} машины M_2 в q_{acc} машины M ;
- из q_{rej} машины M_1 в q_{rej} машины M ;
- из q_{rej} машины M_2 в q_{rej} машины M ;

- Пусть программа P это $\mathbf{while}(Y) \mathbf{P}_1 \mathbf{endwhile}$ — цикл. Мы вводим новые состояния — начальное и два заключительных для машины M . Мы добавляем безусловные переходы:

- из начального состояния машины M в начальное состояние M_T ;

- из q_{rej} машины M_T в q_{acc} машины M ;
- из q_{rej} машины M_1 в q_{rej} машины M ;
- из q_{acc} машины M_T в начальное состояние машины M_1 ;
- из q_{acc} машины M_1 в начальное состояние машины M_T ;

На этом построение машины Тьюринга M завершено.

Лемма 27 Если программа P приняла вход w , то машина Тьюринга M принимает вход w .

Доказательство. Если программа P приняла вход w , то в ней существует путь вычислений α_i , при работе которого на входе w , все тесты в нем истинны. Машина Тьюринга M принимает вход w тогда и только тогда, когда в ней есть ветвь вычислений, завершающаяся в состоянии q_{acc} .

Рассмотрим ветвь вычислений u , соответствующую этому пути вычислений α_i . В этой ветви вычислений все циклы отработают в точности столько же раз, как и в α_i ; все операторы объединения будут работать так же, как и в α_i . В результате ветвь вычислений u , после выполнения каждого блока команд, соответствующих одному оператору, будет моделировать состояние α_i . Но это значит, что, так как все тесты в α_i истинны, в ветви вычислений u все блоки команд, соответствующих одному оператору, будут завершаться в промежуточном состоянии q_{acc} . А значит, и вся ветвь u целиком завершится в состоянии q_{acc} , что и требуется.

Лемма 28 Если программа P не принимает вход w , то и машина Тьюринга M не принимает вход w .

Доказательство. Машина Тьюринга M не принимает вход w тогда и только тогда, когда в M нет ни одной ветви вычислений, завершающаяся в состоянии q_{acc} . Если программа P не приняла вход w , значит каждый путь вычислений в ней либо бесконечен, либо содержит ложный тест.

Будем проводить доказательство от противного — пусть некоторая ветвь вычислений u машины M завершается в состоянии q_{acc} . Заметим, что, по построению машины M , переход из промежуточного состояния q_{rej} в состояние q_{acc} возможен **только** в цикле. Построим путь вычисления α_y по

ветви вычислений u . Он конечен. Так как u закончилась в состоянии q_{acc} , то в α_y нет ни одного теста, который ложен. Но существование такого пути вычислений невозможно по исходному предположению. Из этого следует, что если программа P не принимает вход w , то и машина Тьюринга M не принимает вход w .

2. Докажем, что $\mathbf{NL} \subseteq \mathbf{Z}_\Psi^+$.

Пусть язык $L \in \mathbf{NL}$. Это значит, что язык L распознается машиной Тьюринга M , работающей в логарифмической памяти. Нам необходимо построить недетерминированную программу P из класса PR_Ψ^+ , которая распознает этот язык L .

Без ограничения общности можно считать, что моделируемая машина Тьюринга M имеет одну входную ленту, одну рабочую ленту, каждая лента машины Тьюринга является полубесконечной, и длина рабочей ленты ограничена константой, умноженной на логарифм от длины входной ленты. Благодаря этой константе нам не важно основание логарифма.

Вход машины Тьюринга — слово в алфавите $\{a, b\}$, которому однозначно соответствует универсальная алгебра из $A(\Psi)$, на которой работает недетерминированная программа.

При длине входа N , количество различных вариантов заполнения рабочей ленты равно $d^{\log_e N} = N^v$, для некоторой константы v . Учитывая, что каждая переменная недетерминированной программы принимает значения от 0 до $N-1 = LAST$, а также a, b, err , то с помощью v переменных недетерминированной программы можно закодировать рабочую ленту. Количество состояний в машине Тьюринга конечно, следовательно их можно закодировать с помощью конечного числа переменных.

Позиция на входной ленте — это число от 0 до N^u , где u — максимальная аргументность функции подающейся на вход, а значит также может быть закодирована с помощью u переменных.

Определение 29 Кортёж переменных — это последовательность переменных недетерминированной программы, кодирующая

- состояние машины Тьюринга;
- или состояние одной рабочей ленты машины Тьюринга;
- или позицию на входной ленте.

Обозначим кортеж, соответствующий состоянию q через x_q . Кортеж, соответствующий текущему состоянию обозначим через x_{curQ} . Кортеж, соответствующий рабочей ленте обозначим $work$. Подчеркнем, что в программе нет кортежа, соответствующего входной ленте (но есть кортеж с позицией на входной ленте). Будем обозначать кортеж — текущую позицию на входной ленте за $posV$. Позиция на рабочей ленте — число, меньше $\log_c N$ и также может быть записана в одной переменной. Для единообразия, можно считать ее кортежем из одной переменной. Будем обозначать позицию на рабочей ленте за $posW$.

Так как все операции осуществляются с кортежами переменных (разной длины), операции сравнения кортежей, добавления и вычитания единицы к кортежу являются элементарными подпрограммами.

Элементарные подпрограммы

При моделировании машин Тьюринга мы будем использовать следующие подпрограммы:

- Программа $Input(i)$, вычисляющая i -ый символ на входной ленте машины M .
- Программа $WorkSymbol(w, i)$, вычисляющая i -ый символ на рабочей ленте, соответствующей кортежу W .
- Программа $WriteSymbol(w, s, i)$, моделирующая запись символа s на i -ую позицию на рабочей ленте, соответствующей кортежу W .
- Программа $ShiftInput(posV, d)$, выполняющая оператор $i = i + d$ для счетчика, соответствующего входной ленте. $d \in \{-1, 0, 1\}$ — это сдвиг.
- Программа $ShiftWork(posW, d)$, выполняющая оператор $i = i + d$ для счетчика, соответствующего рабочей ленте. $d \in \{-1, 0, 1\}$ — это сдвиг.

Определение 30 Подпрограмма — это набор операторов недетерминированной программы.

Лемма 31 Каждая команда машины Тьюринга M может быть промоделирована подпрограммой.

Доказательство. Рассмотрим произвольную команду com машины Тьюринга. Она имеет вид:

$$com = \langle cq, ca, cb \rangle \rightarrow \langle cq', cw, cs_1, cs_2 \rangle, \text{ где}$$

- cq и cq' — состояния;

- ca — символ, который машина Тьюринга наблюдает в текущий момент на входной ленте;
- cb — символ, который машина Тьюринга наблюдает в текущий момент на рабочей ленте;
- cw — символ, который машина Тьюринга запишет на рабочую ленту в текущую позицию;
- cs_1 — сдвиг на входной ленте;
- cs_2 — сдвиг на рабочей ленте.

Как уже говорилось выше, несколько команд машины Тьюринга могут иметь одинаковую левую часть.

Каждая команда машины Тьюринга преобразуется в собственную подпрограмму.

В подпрограмме, соответствующей команде com машины Тьюринга M вначале проверяется, что

- кортеж текущего состояния x_{curQ} равен x_{cq} , соответствующему состоянию cq из левой части команды com ;
- на входной ленте в текущей позиции записан правильный символ, то есть $Input(posV)$ совпадает с ca ;
- на рабочей ленте в текущей позиции записан правильный символ, то есть $WorkSymbol(work, posW)$ совпадает с cb .

Все эти проверки являются вложенными условными операторами.

Если все эти условия выполнены, то подпрограмма далее

- меняет кортеж состояния x_{curQ} на $x_{cq'}$;
- меняет кортеж, кодирующий рабочую ленту, выполняя $WriteSymbol(work, cw, posW)$;
- меняет текущие позиции на входной и рабочей лентах (выполняя элементарные подпрограммы $ShiftInput(posV, cs_1)$ и $ShiftWork(posW, cs_2)$).

Заметим, что подпрограмма, соответствующая одной команде машины Тьюринга, обладает следующим свойством: в этой программе все тесты выполнены тогда и только тогда, когда данная команда может быть применена, то есть выполнены все условия левой части команды. Это и означает, что подпрограмма моделирует команду машины Тьюринга.

Построение программы P , моделирующей машину Тьюринга M .

Далее все полученные таким образом программы s_1, \dots, s_d , где d — количество команд машины Тьюринга объединяются следующим образом: Пока текущее состояние моделируемой машины M не заключительное (то есть не q_{acc} или q_{rej}), мы выполняем **объединение** всех подпрограмм, соответствующих командам машины Тьюринга.

(p = 0)

while(p == 0)

$P_1 \cup P_2 \cup \dots \cup P_d$

if($x_{curQ} == x_{q_{acc}}$)p = err;

endwhile .

На этом построение программы P завершено.

По определению, машина Тьюринга распознает некоторое слово тогда и только тогда, когда существует последовательность команд машины Тьюринга, переводящая ее в заключительное состояние q_{acc} . В случае, если существует последовательность команд машины Тьюринга, переводящая ее в заключительное состояние, будет существовать и конечный путь вычислений, соответствующий данной последовательности команд.

Лемма 32 Если машина Тьюринга M принимает вход w , то и программа P принимает вход w .

Доказательство. Если машина Тьюринга M принимает вход w , то в ней есть ветвь вычислений, завершающаяся в состоянии q_{acc} . Если в программе P при работе на входе w есть путь вычислений, на котором все тесты истинны, то программа P принимает вход w . Соответственно, нам необходимо доказать, что если в исходной машине Тьюринга M была ветвь вычислений, завершающаяся в состоянии q_{acc} , то в программе P при работе на входе w будет путь вычислений, на котором все тесты истинны.

Рассмотрим ветвь вычислений y , завершающаяся в состоянии q_{acc} . Ей соответствует набор подпрограмм $P_{i_1}, P_{i_2}, \dots, P_{i_k}$, таких что P_{i_j} соответствует j -ой команде ветви y . Рассмотрим путь вычислений r , полученный последовательной композицией этих подпрограмм, однозначно определяемый по одной ветви вычислений M . В конец r добавим тест

$x_{curQ} = x_{q_{acc}}$. Дополним путь вычислений r тестами $p == 0$, $x_{curQ} \neq x_{q_{acc}}$, между каждой из подпрограмм P_{i_j} и $P_{i_{j+1}}$. В результате r будет путем вычислений из множества путей вычислений итоговой программы P . Докажем, что на r все тесты истинны.

- Все тесты в подпрограммах P_{i_j} истинны, так как они соответствуют условиям, при которых применялись команды ветви вычислений y машины Тьюринга M .

- Все добавленные тесты $p == 0$, $x_{curQ} \neq x_{q_{acc}}$ истинны, так как при выполнении ветви вычислений y состояние машины Тьюринга равно q_{acc} только после выполнения последней из команд ветви вычислений y , но не ранее.

- Добавленный тест $x_{curQ} == x_{q_{acc}}$ истинен, так как в конце выполнения ветви вычислений y состояние машины Тьюринга действительно равно q_{acc} .

Никаких других тестов в рассматриваемом пути вычислений нет, а, значит, программа P действительно принимает вход w .

Лемма 33 Если машина Тьюринга M не принимает вход w , то и программа P не принимает вход w .

Доказательство. Если машина Тьюринга M не принимает вход w , то в ней нет ни одной ветви вычислений, завершающаяся в состоянии q_{acc} . Программа P не принимает вход w тогда и только тогда, когда в ней при работе на входе w нет ни одного пути вычислений, на котором все тесты истинны. Соответственно, нам необходимо доказать, что если в исходной машине Тьюринга M не было ветви вычислений, завершающейся в состоянии q_{acc} , то в программе P при работе на входе w не будет ни одного пути вычислений, на котором все тесты истинны.

Пусть в исходной машине Тьюринга M не было ветви вычислений, завершающейся в состоянии q_{acc} . Будем доказывать утверждение от противного. Предположим, что существует путь вычислений программы P , на котором все тесты истинны. По построению итоговой программы, чтобы выйти из цикла результирующей программы P , должен был быть истинным тест $x_{curQ} == x_{q_{acc}}$. Однако это значит, что в результате выполнения

некоторого конечного набора подпрограмм $P_{i_1}, P_{i_2}, \dots, P_{i_k}$ верно, что $x_{curQ} \equiv x_{q_{acc}}$. По построению подпрограмм P_{i_j} , это значит, что в машине Тьюринга M существует ветвь вычислений, завершающаяся в состоянии q_{acc} , что неверно. Полученное противоречие доказывает утверждение.

Список литературы.

- [1] Cook S.A. Variations on push-down machines In Proc. 1st ACM Symp. on Theory of Computing, pp. 229–232, 1969.
- [2] Cook S.A. Characterizations of push-down machines in terms of time-bounded computers Journal of the ACM, 18(1), pp. 4–18, 1971.
- [3] Cook S.A. Senti R. Storage requirements for deterministic polynomial time recognizable languages Journal of Computer and System Sciences, 13(1), pp. 25–37, 1976.
- [4] Immerman N. Nondeterministic space is closed under complementation, SIAM Journal on Computing 17, pp. 935–938, 1988.
- [5] Savitch W.J. Relationships between nondeterministic and deterministic tape complexities, Journal of Computer and System Sciences 4 (2), pp. 177–192, 1970.
- [6] Papadimitriou C. Computational Complexity. Boston: Addison-Wesley. ISBN 0-201-53082-1, 1994.
- [7] Arora S., Barak B. Computational complexity. A modern approach. Cambridge University Press. ISBN 978-0-521-42426-4, 2009.
- [8] Столбоушкин А.П., Тайтлин М.А. Динамические логики. Кибернетика и вычислительная техника под ред. В.А. Мельникова, том 2, Наука, Москва, стр. 180–230, 1986.

Syntactical characterization of nondeterministic logspace complexity class

D.A. Nosov
Yandex, Moscow, Russia
dmitrytvr@gmail.com

Abstract. \mathbf{NL} is defined as the class of languages recognizable by logspace nondeterministic Turing machines. One of the main unsolved problems in complexity theory is that of relation between classes \mathbf{P} and \mathbf{NL} . It is known that \mathbf{NL} is contained in \mathbf{P} , since there is a polynomial-time algorithm for 2-satisfiability, but it is not known whether $\mathbf{NL} = \mathbf{P}$ or whether $\mathbf{NL} = \mathbf{L}$. A possible approach to these problems can be based on searching for an alternative suitable definition of the class \mathbf{NL} . Taitlin et al. propose such a definition in terms of nondeterministic programs. The syntax of such programs is similar to that of usual computer programs. Each nondeterministic program takes a finite universal algebra as input. Taitlin et al. defined a class of languages recognizable by such programs and proved that this class is a subclass of \mathbf{NL} . In the present paper, we slightly modify their syntactical definition. Namely, we modify a definition of nondeterministic program input and give a new definition of a language recognized by a given program. We prove that the class of languages recognizable by nondeterministic programs according to our definition is just \mathbf{NL} .

Keywords: Turing machine; nondeterministic program; complexity class; logspace complexity; universal algebra

References

- [1] Cook S.A. Variations on push-down machines In Proc. 1st ACM Symp. on Theory of Computing, pp. 229–232, 1969.
- [2] Cook S.A. Characterizations of push-down machines in terms of time-bounded computers Journal of the ACM, 18(1), pp. 4–18, 1971.
- [3] Cook S.A. Senti R. Storage requirements for deterministic polynomial time recognizable languages Journal of Computer and System Sciences, 13(1), pp. 25–37, 1976.
- [4] Immerman N. Nondeterministic space is closed under complementation, SIAM Journal on Computing 17, pp. 935–938, 1988.
- [5] Savitch W.J. Relationships between nondeterministic and deterministic tape complexities, Journal of Computer and System Sciences 4 (2), pp. 177–192, 1970.
- [6] Papadimitriou C. Computational Complexity. Boston: Addison-Wesley. ISBN 0-201-53082-1, 1994.
- [7] Arora S., Barak B. Computational complexity. A modern approach. Cambridge University Press. ISBN 978-0-521-42426-4, 2009.
- [8] Stolboushkin A.P., Taitlin M.A. Dinamicheskie logiki. [Dynamic logics] Кибернетика и вычислительная техника [Cybernetics and computer technology] V.A. Melnikov (editor). Moscow, Nauka, pp. 180–230, 1986 (in Russian).