Теоретическая информатика, осень 2020 г. Лекция 4. Теорема Майхилла–Нероуда. Вероятностные конечные автоматы (PFA). Двухсторонние вероятностные конечные автоматы (2PFA). Введение в формальные грамматики*

Александр Охотин

25 сентября 2020 г.

Содержание

1 Теорема Майхилла–Нероуда

Доказательство правильности приводившегося ранее алгоритма минимизации DFA основывалось на рассуждениях о разбиении множества строк на классы эквивалентности. Эти рассуждения удобно представить в виде следующего абстрактного результата.

Теорема 1 (теорема Майхилла–Нероуда, см. Нероуд [1958])**.** *Пусть L —язык, и пусть $\equiv_L$ — отношение эквивалентности на множестве строк, где $u \equiv_L u'$, если для всякой строки v строки uv и $u'v$ или обе лежат в L или обе не лежат. Утверждается, что язык L регулярен тогда и только тогда, когда число классов эквивалентности в отношении $\equiv_L$ конечно.*

Доказательство. $\Rightarrow$ Классы эквивалентности — это состояния минимального DFA.

$\Leftarrow$ Обозначение: $[u]$ — класс эквивалентности, в который попала строка u.

Строится DFA, состояниями которого станут классы эквивалентности. Начальный класс — $q_0 = [\varepsilon]$. Переход из класса $S \subseteq \Sigma^*$ по символу $a \in \Sigma$ определяется как $\delta(S, a) = [ua]$, где u — произвольная строка из S; ниже будет показано, что это определение не зависит

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


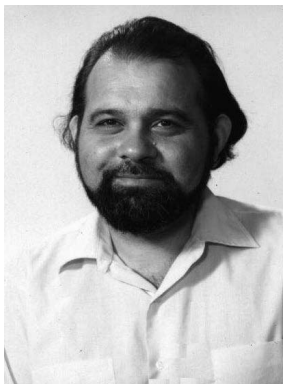

Рис. 1: Джон Майхилл (1923–1987), Анил Нероуд (род. 1932).

от выбора строки u. Наконец, класс $S \subseteq \Sigma^*$ — принимающее состояние построенного DFA, если $S \subseteq L$.

Утверждается, что, прочитав строку u, построенный DFA приходит в класс S, содержащий эту строку. Доказательство — индукция по длине u. Базовый случай: начальное состояние, $[\varepsilon]$, содержит ε.

Переход: если автомат пришёл в класс S по строке $\widehat{u}$, то известно, что $\widehat{u} \in S$. Для вычисления перехода по a выбирается какая-то строка $u \in S$, и автомат переходит в состояние $[ua]$. Утверждается, что $[ua]$ и $[u'a]$ — это один и тот же класс. Нужно показать, что $uav \in L$ тогда и только тогда, когда $u'av \in L$. Действительно, поскольку $u, \widehat{u} \in S$, для их продолжений строкой av искомая эквивалентность выполнена.

Наконец, пусть автомат прочитал всю входную строку w и пришёл в состояние S, содержащее w. Тогда, если $w \in L$, то $S \subseteq L$, и потому это состояние будет принимающим. Если же $w \notin L$, то $S \cap L = \varnothing$, и состояние, соответственно, окажется отвергающим. □

Рассуждение в приведённом доказательстве — то же, что и в доказательстве правильности алгоритма минимизации. В следующем разделе это же рассуждение потребуется в более сложном контексте, и тогда теорема Майхилла–Нероуда окажется весьма полезной.

2 Вероятностные конечные автоматы (PFA)

Вероятностный конечный автомат (probabilistic finite automaton, PFA) похож на NFA в том, что он может сделать разные переходы по одному и тому же символу в одном и том же состоянии. Однако при этом возможным переходам сопоставлены *вероятности* — и у каждого вычисления тоже, соответственно, есть вероятность. В отличие от NFA, это физически реализуемая модель.

Определение 1 (Рабин [1963]). ***Вероятностный конечный автомат*** *(probabilistic finite automaton, PFA) — это пятёрка $\mathcal{B} = (\Sigma, Q, q_0, \delta, F)$, со следующим значением компонентов.*

- Σ — *алфавит.*
- Q — *конечное множество состояний.*
- $q_0 \in Q$ — *начальное состояние.*

- *Функция переходов* $\delta\colon Q \times \Sigma \to [0,1]^Q$, *где* $[0,1]^Q$ — *это множество стохастических функций* f *из* Q *в* $[0,1]$, *для которых верно* $\sum_{q \in Q} f(q) = 1$. *Эта функция даёт вероятность перехода в каждое состояние. Находясь в состоянии* $q \in Q$ *и читая символ* $a \in \Sigma$, *автомат переходит во всякое состояние* r *с вероятностью* $\delta(q,a)(r)$.
- *Множество* ***принимающих состояний*** $F \subseteq Q$.

Вычисление на входной строке $w = a_1 \dots a_\ell$ — *это всякая последовательность состояний* $p_0, p_1, \dots, p_{\ell-1}, p_\ell$.

$$p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \dots \xrightarrow{a_{\ell-1}} p_{\ell-1} \xrightarrow{a_\ell} p_\ell$$

Такое вычисление называется вычислением из p_0 *в* p_ℓ. *Вероятность вычисления — произведение* $\prod_{i=1}^{\ell} \delta(p_{i-1}, a_i)(p_i)$.

Вероятность придти из p *в* q, *прочитав строку* w — *это сумма вероятностей всех таких вычислений. Вероятность принять из данного состояния* p — *это сумма вероятностей придти из* p *во все принимающие состояния. Вероятность принять строку — это вероятность принять её из начального состояния.*

Автомат должен удовлетворять следующему условию ограниченной ошибки: *всякая строка принимается или с вероятностью хотя бы* $\frac{2}{3}$, *или с вероятностью не более чем* $\frac{1}{3}$. *Язык, распознаваемый автоматом, обозначаемый через* $L(\mathcal{B})$ — *это множество всех входных строк, которые принимаются с вероятностью более чем* $\frac{1}{2}$.

Оказывается, что всякий PFA распознаёт регулярный язык. Это доказывается прямым построением DFA, который, прочитав некоторую входную строку, с некоторой точностью вычислит вероятности, с которыми исходный PFA будет находится в различных своих состояниях после прочтения той же самой строки. Благодаря условию ограниченной ошибки, ограниченной точности окажется достаточно, чтобы воспроизвести решение вероятностного автомата о принятии или отвержении строки. Какой точности достаточно, устанавливается в следующем доказательстве.

Теорема 2 (Рабин [1963]). *Всякий PFA распознаёт регулярный язык.*

Доказательство. Пусть $\mathcal{B} = (\Sigma, Q, q_0, \delta, F)$ — произвольный PFA, и пусть $|Q| = n$. Для всякой строки $u \in \Sigma^*$ рассматривается вектор $p(u)$, составленный из вероятностей придти в каждое из состояний по строке u. Аналогично, для всякой строки $v \in \Sigma^*$ вводится вектор $q(v)$, состоящий из вероятностей принять v из каждого из состояний. Тогда скалярное произведение $(p(u), q(v))$ — это вероятность принять строку uv.

На таких векторах вводится норма — максимум. Утверждается, что если два вектора $p(u) = (u_1, \dots, u_n)$ и $p(u') = (u'_1, \dots, u'_n)$ достаточно близки, то тогда из них принимаются одни и те же строки. Для некоторого $\varepsilon > 0$, пусть $||p(u) - p(u')|| \leqslant \varepsilon$. Исходя из этого, для данной строки v с вектором $q(v) = (v_1, \dots, v_n)$ предлагается оценить разность между вероятностью принять uv и вероятностью принять $u'v$.

$$\begin{aligned} \big|p(u)q(v) - p(u')q(v)\big| = \Big|\sum_i u_i v_i - \sum_i u'_i v_i\Big| = \Big|\sum_i (u_i - u'_i) v_i\Big| \leqslant \\ \leqslant \Big|\sum_i (u_i - u'_i)\Big| \leqslant n \max_i |u_i - u'_i| \leqslant n\varepsilon \end{aligned}$$


Рис. 2: Русиньш Фрейвальдс (1942–2016), Андрис Амбайнис (род. 1975).

Теперь пусть $\varepsilon = \frac{1}{3n+1}$. Тогда из $||p(u) - p(u')|| \leqslant \varepsilon$ следует $|p(u)q(v) - p(u')q(v)| \leqslant \frac{n}{3n+1} < \frac{1}{3}$. Поскольку вероятность принять строку или не превосходит $\frac{1}{3}$, или больше или равна $\frac{2}{3}$, это означает, что строки uv и $u'v$ или обе принимаются, или обе отвергаются.

Гиперкуб $[0,1]^n$ можно разделить на $(3n+1)^n$ кубиков со стороной ε. Внутри каждого кубика любые два вектора отстоят друг от друга не более, чем на ε, и, стало быть, PFA принимает из всех этих распределений вероятностей один и тот же язык. Это значит, гиперкуб $[0,1]^n$ можно разбить не более чем на $(3n+1)^n$ непересекающихся классов, из которых принимаются попарно различные языки. Следовательно, согласно теореме Майхилла–Нероуда, язык регулярен. □

Амбайнис [1996] построил примеры PFA с n состояниями, для которых всякий равносильный им DFA требует $\Omega(2^{\frac{n \log\log n}{\log n}})$ состояний — это называется *сложностью преобразования PFA в DFA по числу состояний.*

3 Двухсторонние вероятностные конечные автоматы (2PFA)

По образцу определяются двухсторонние вероятностные конечные автоматы (2PFA) — однако их выразительная мощность неожиданно оказывается большей, чем у PFA и 2DFA: Фрейвальдс [1981] построил следующий 2PFA, распознающий нерегулярный язык.

Пример 1 (Фрейвальдс [1981])**.** *Существует 2PFA, распознающий язык* $\{\, a^n b^n \mid n \geqslant 0 \,\}$.

Доказательство. Автомат распознаёт этот язык за счёт долгого хождения взад-вперёд в ожидании двух различных крайне маловероятных событий. Математическое ожидание времени работы автомата будет экспоненциальным от длины входной строки, определение 2PFA этого не запрещает.

Сперва автомат проверяет, что строка имеет вид $a^i b^j$, где $i \equiv j \pmod 3$, если нет — отвергает. А дальше автомат ходит туда-сюда, на каждом проходе для каждого символа подбрасывая монету, с вероятностью $\frac{1}{2}$ выбирая орёл или решку. На каждом проходе он собирает следующую статистику:

- выпадал ли хоть раз орёл на каком-нибудь символе a,
- выпадал ли хоть раз орёл на каком-нибудь символе b.

По итогам прохода, если орёл выпадал только на символах одного типа, а на других символах не выпадал — это «успех этого символа». Вероятнее успех того символа, который встречается чаще. Далее автомат будет следить за успехами обоих символов, и если он заметит, что один символ успешнее другого, он сделает на основании этого вывод, что, вероятно, $i \neq j$; если же успехи символов будут казаться одинаковыми, то автомат решит, что их, наверное, равное количество.

Для этого автомат запоминает, сколько раз имел место успех каждого из двух символов, считая до трёх. Если один символ трижды имел успех, а другой — ни разу, то автомат отвергает. Иначе же, если оба символа имели успех хотя бы по одному разу, автомат принимает. Поскольку успех символа случается пусть с небольшой, но ненулевой вероятностью, условие остановки со временем выполнится с вероятностью 1.

Чтобы всё это реализовать, достаточно $6 + 5 + 4 \cdot 5 + 1 = 32$ состояний: 6 для проверки общего вида $a^i b^j$, где $i \equiv j \pmod 3$; ещё 5 для возвращения в начало строки, храня в памяти значения двух счётчиков количества успехов; $4 \cdot 5$ для каждого прохода, когда кроме счётчиков количества успехов запоминается выпадение орла на символах каждого из типов; и наконец, 1 принимающее состояние.

Если $i = j$, то успех a и успех b равновероятны, и потому три успеха a без успехов b происходят с вероятностью $\frac{1}{8}$, и с такой же вероятностью при перемене символов. Стало быть, автомат отвергает с вероятностью $\frac{1}{8} + \frac{1}{8} = \frac{1}{4}$ и принимает с вероятностью $\frac{3}{4}$.

Пусть $i \neq j$. Успех a означает, что на b орёл не выпадал ни разу, и, стало быть, всегда выпадала решка — и вероятность этого равна $\frac{1}{2^j}$; на a же хоть раз выпадал орёл — иными словами, решка выпадала не всегда — что происходит с вероятностью $1 - \frac{1}{2^i}$. Отсюда вероятность успеха a равна $\frac{1}{2^j}(1 - \frac{1}{2^i})$, а вероятность успеха одного из символов равна $\frac{1}{2^i} + \frac{1}{2^j} - 2\frac{1}{2^{i+j}}$.

Пусть $i > j$; а так как $i \equiv j \pmod 3$, из этого следует, что $j \leqslant i - 3$. Тогда утверждается, что условная вероятность события «если какой-то символ имел успех, то это символ a» достаточно высока, чтобы три таких события с большой вероятностью произошли подряд. Эта условная вероятность оценивается так.

$$\frac{\frac{1}{2^j}(1 - \frac{1}{2^i})}{\frac{1}{2^i} + \frac{1}{2^j} - 2\frac{1}{2^{i+j}}} = \frac{2^i - 1}{2^j + 2^i - 2} \geqslant \frac{2^i - 1}{2^{i-3} + 2^i - 2} =$$

$$= \frac{2^i + 2^{i-3} - 2 - 2^{i-3} + 1}{2^i + 2^{i-3} - 2} = 1 - \frac{2^{i-3} - 1}{2^i + 2^{i-3} - 2} \approx 1 - \frac{1}{9}$$

Тогда вероятность того, что первые 3 успеха будут успехами для a равна $(1 - \frac{1}{9})^3 = \frac{512}{729} > 0.7$, и с этой вероятностью автомат благополучно отвергнет.

В случае $i < j$ так же рассчитывается, что условная вероятность события «если какой-то символ имел успех, то это символ b» приближённо равна $1 - \frac{1}{9}$, и автомат тоже отвергает с вероятностью не менее чем 0.7. □

Квантовые конечные автоматы. Ещё одна интересная теоретическая модель, отчасти схожая с вероятностными конечными автоматами — это *квантовые конечные автоматы* (QFA, 2QFA), которые в каждый момент времени одновременно находятся во всех своих состояниях, в каждом со своей комплексной амплитудой. Это простейшая модель квантовых вычислений; но квантовые вычисления как таковые — это особая тема, лежащая за рамками базового курса теоретической информатики.

4 Введение в формальные грамматики

Формальная грамматика — математическая модель синтаксиса языков, как естественных (русский, и т.д.), так и искусственных (C++, XML и т.д.). С помощью формальных грамматик можно строго определить такие задачи, как *разбор предложений по составу* (как это делают в школе). Разработанные в этой теории алгоритмы позволяют автоматизировать эту задачу и построить программы — синтаксические анализаторы, разбирающие поданные на вход строки согласно заранее заданной грамматике языка. Эти алгоритмы используются в системах распознавания естественных языков, в компиляторах языков программирования и в других задачах анализа текстовых данных.

4.1 Синтаксис языков

Общие черты синтаксиса языков:

1. представление информации в виде строки символов из некоторого алфавита;
2. не всякая символьная строка — правильно построенное предложение, и синтаксис — это принцип построения правильных предложений;
3. предложения строятся из фрагментов меньшего размера (словосочетаний), каждый со своим смыслом, а те в свою очередь могут состоять из ещё более коротких фрагментов, и так далее до уровня слова; при этом смысл общего выводится из смысла частей в соответствии с иерархической структурой предложения.

Представление информации в виде строк естественно для людей, поскольку самый естественный способ передачи информации — речь — представляет собой последовательность членораздельных звуков.

В этом курсе будет изучаться основная разновидность формальных грамматик, в которых фрагменты предложения — это *подстроки* этого предложения, и *правила грамматики* описывают, как несколько подстрок с известными синтаксическими категориями могут быть записаны одна за другой, чтобы получить новую подстроку с определённой синтаксической категорией. Так как никаких других видов грамматик в курсе не будет, такие грамматики будут называться просто *грамматиками*[1].

4.2 Синтаксис естественных языков

При разборе предложения по составу в нём выделяются подстроки — слова, из которых составляются словосочетания, обороты, и так до тех пор, пока они все не соединяются во всё предложение. Структура соединения меньших частей в бóльшие образует *дерево разбора*.

В традиционных грамматиках описываются синтаксические категории и то, как они могут сочетаться. Например, английское предложение «`Every man is mortal`» разделяется на подлежащее с зависимыми словами «`Every man`» (noun phrase, «состав подлежащего») и сказуемое с зависимыми словами «`is mortal`» (verb phrase, «состав сказуемого»). Американские лингвисты, такие как Блумфилд и Сапир, называли этот подход *анализом непосредственных составляющих* (immediate constituent analysis).

[1] К ним прилипло странное название «бесконтекстные грамматики» (context-free; на птичьем языке — «контекстно-свободные»), возникшее в конце 1950-х в связи тогдашними попытками придумать модель грамматик, где задавались бы правила, применимые в отдельных контекстах. Плоды этих попыток давно и безнадёжно устарели, но от них осталось сомнительное название для обыкновенных грамматик, мешающее правильно их понимать уже нескольким поколениям. Однако это название всё равно нужно запомнить, поскольку оно широко используется.

Рис. 3: Леонард Блумфилд (1887–1949), Эдвард Сапир (1884–1939), Ноам Хомский (род. 1928).

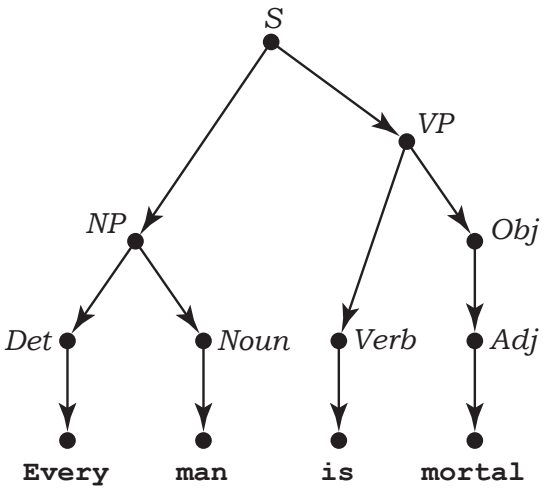

Рис. 4: Дерево разбора предложения `Every man is mortal`.

Эти правила можно выразить в виде математического определения грамматически верных предложений, состоящего из высказываний вида «Если вслед за любым составом подлежащего (noun phrase) записать любой состав сказуемого (verb phrase), то получится предложение». В обозначениях формальных грамматик это правило записывается так.

$$S \to \text{NP VP}$$

В таком виде правила грамматик были впервые записаны Хомским [1956], и с этой формализации очевидных законов языка началось математическое изучение формальных грамматик.

Синтаксис естественных языков не столь упорядочен, чтобы совершенно точно описываться формальными грамматиками. Как в своё время остроумно заметил Сапир [1921], *все грамматики подтекают* («all grammars leak») — и, несмотря на все последующие математические изыскания, грамматики продолжают подтекать и сегодня. Тем не менее, формальные грамматики остаются наилучшим математическим приближением синтаксиса естественных языков, и более точные модели в лингвистике получаются их уточнением.

4.3 Синтаксис языков программирования

Первым систематически разработанным языком программирования стал *Алгол 60*. В определении языка формальные грамматики использовались в качестве средства определения

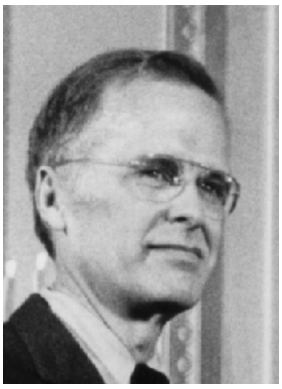
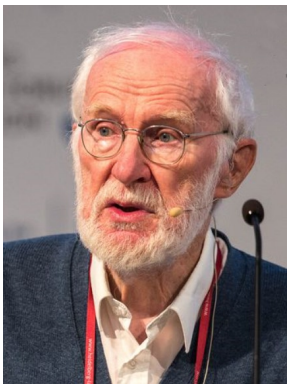

Рис. 5: Джон Бэкус (1924–2007), Петер Наур (1928–2016).

синтаксиса — причём разработчики не знали об изобретениях лингвистов и легко переоткрыли ту же самую модель. В программистской среде грамматики получили название BNF (Backus–Naur form, форма Бэкуса–Наура), в честь двух членов комитета разработчиков.

Арифметические выражения.

Пример 2 (Синтаксис арифметических выражений)**.** *На общематематическом языке арифметические выражения обычно определяются примерно так.*

- *Переменная x — арифметическое выражение.*
- *Число — арифметическое выражение.*
- *Если e и e' — выражения, то e `+` e' и e `*` e' — тоже выражения.*
- *Если e — выражение, то `(`e`)` — тоже.*

Ничто другое не является арифметическим выражением.

Для простоты. пусть x — переменная, `1` — число. Тогда `x*(x+1)` — арифметическое выражение, а `x*)1+` — нет.

Это определение переписывается в виде формальной грамматики следующим образом.

$$E \to x \mid \texttt{1} \mid E\texttt{+}E \mid E\texttt{*}E \mid \texttt{(}E\texttt{)}$$

Здесь E — синтаксическая категория арифметических выражений.

Операторы языка программирования `while(x<0) x=x+1;`

Продолжение: логические выражения.

$$B \to E\texttt{==}E \mid E\texttt{<}E$$

Пример 3.

i. Если x — переменная, а e — арифметическое выражение, то x `=` e `;` — оператор.

ii. Если s — оператор и e — логическое выражение, то `while` e `do` s — тоже оператор.

iii. Если $s_1, \ldots, s_k$ — операторы, где $k \geqslant 0$, то `begin` $s_1 \ldots s_k$ `end` — оператор.

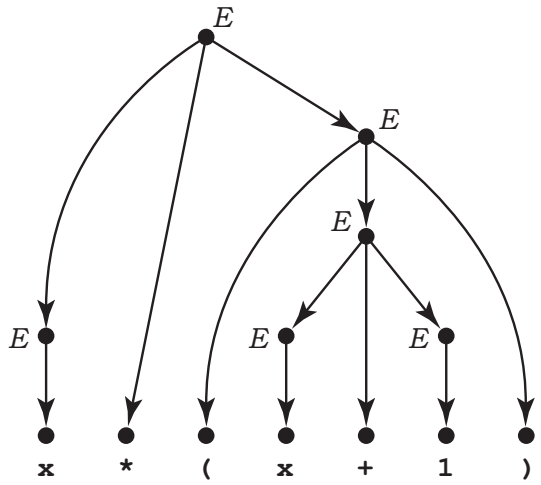

Рис. 6: Дерево разбора выражения `x*(x+1)` по грамматике из примера 2.

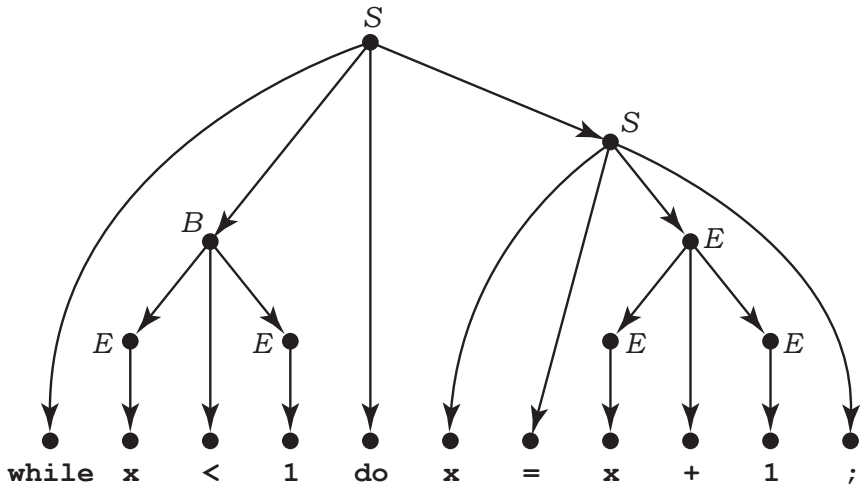

Рис. 7: Дерево разбора оператора `while x<1 do x=x+1;` по грамматике из примера 3.

И т.д., ещё несколько видов операторов.

Пусть S — категория синтаксически правильных операторов.

$$S \to \texttt{x = } E \texttt{;} \mid \texttt{while } B \texttt{ do } S \mid \texttt{begin } S' \texttt{ end}$$
$$S' \to SS' \mid \varepsilon$$

Например, поскольку `x<1` — выражение, а `x=x+1;` — оператор, то `while x<1 do x=x+1;` — также оператор. Дерево разбора дано на рис. 7.

Список литературы

[1996] A. Ambainis, “The complexity of probabilistic versus deterministic finite automata”, *ISAAC 1996*, 233–238.

[1956] N. Chomsky, “Three models for the description of language”, *IRE Transactions on Information Theory*, 2:3 (1956), 113–124.

[1981] R. Freivalds, “Probabilistic two-way machines”, *MFCS 1981*, 33–45.

[1958] A. Nerode, “Linear automaton transformations”, *Proceedings of the AMS*, 9:4 (1958), 541–544.

[1963] M. O. Rabin, “Probabilistic automata”, *Information and Control*, 6:3 (1963), 230–245.

[1921] E. Sapir, *Language: An Introduction to the Study of Speech*, 1921.