Math-Net.Ru

Общероссийский математический портал

А. С. Охотин, О сложности задачи генерации строк, *Дискрет. матем.*, 2003, том 15, выпуск 4, 84–99

DOI: <https://doi.org/10.4213/dm217>

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением
<http://www.mathnet.ru/rus/agreement>

Параметры загрузки:

IP: 34.239.49.27

26 апреля 2023 г., 12:11:57

О сложности задачи генерации строк

© 2003 г. А. С. Охотин

В работе рассматривается сложность задачи генерации строк, принадлежащих определенным языкам и удовлетворяющих некоторым дополнительным ограничениям. Языки задаются при помощи формальных грамматик и автоматов. Предлагается следующая формулировка этой задачи в виде задачи распознавания свойств: для заданного языка, представленного формальной грамматикой или автоматом, и для заданной пары строк выяснить, существует ли строка из этого языка, лежащая лексикографически между этими строками. Установлено, что такая задача NLOGSPACE-полна для детерминированных и недетерминированных конечных автоматов, а также для линейных контекстно-свободных грамматик; P-полна для контекстно-свободных грамматик общего вида; NP-полна для альтернирующих конечных автоматов, для конъюнктивных грамматик и для линейных конъюнктивных грамматик; PSPACE-полна для контекстно-зависимых грамматик и линейно ограниченных по памяти машин Тьюринга.

1. Введение

Среди задач, изучаемых в теории формальных языков и автоматов, можно выделить две наиболее важных с практической точки зрения — это задача определения правильности некоторой заданной строки относительно заданной системы правил (задача принадлежности строки языку) и задача построения правильных строк, удовлетворяющих данной системе правил (задача генерации строки).

В то время как задача принадлежности является типичной задачей распознавания свойств, и потому необходимый для ее решения объем вычислительных ресурсов может быть оценен с позиций теории сложности, трудность задачи генерации в ее исходном виде оценить не так легко.

Однако можно заметить, что задача генерации строк, принадлежащих данному языку, может быть представлена как задача распознавания некоторых специальных свойств самого языка. Так, например, на практике генерация строки состоит в поиске строк, удовлетворяющих определенным ограничениям (в отсутствие ограничений алгоритм генерации может уметь генерировать ровно одну принадлежащую языку строку, что не соответствует практическому смыслу задачи), и в качестве способа задания таких ограничений может выступать, например, отношение лексикографического порядка. В таком случае задача порождения требуемых строк, с одной стороны, включает в себя проверку существования строки языка, удовлетворяющей заданным лексикографическим ограничениям, а с другой стороны, может быть сведена к конечному числу таких проверок.

Точная формулировка этой задачи такова: по данному вычислительному устройству X , порождающему строки над алфавитом Σ , и по данной паре строк $u, v \in \Sigma^*$, определить, существует ли строка, порождаемая устройством X , которая лексикографически

больше или равна u и меньше или равна v . Заметим, что эта задача разрешима для любого класса вычислительных устройств, для которого разрешима задача принадлежности, поскольку она может быть решена простым перебором конечного числа строк и проверки принадлежности каждой языку.

В разделе 2 эта задача сформулирована более строго и рассмотрены четыре ее частных случая. В разделе 3 исследована сложность задачи для детерминированных, недетерминированных и альтернирующих конечных автоматов, и показано, что во всех рассматриваемых формулировках задача NLOGSPACE-полна для конечных автоматов первых двух типов и NP-полна для альтернирующих.

Раздел 4 посвящен исследованию сложности задачи для контекстно-свободных и линейных контекстно-свободных грамматик. В линейном случае легко устанавливается NLOGSPACE-полнота задачи, а для контекстно-свободных грамматик общего вида строится полиномиальный алгоритм ее решения и показывается, что задача P-полна.

Сложность задачи для конъюнктивных грамматик — обобщения контекстно-свободных грамматик, допускающего использование явной операции пересечения языков — рассмотрена в разделе 5, где доказано, что задача NP-полна как для конъюнктивных грамматик общего вида, так и для подкласса линейных конъюнктивных грамматик, определяемого по аналогии с линейными контекстно-свободными.

Задача оказывается PSPACE-полной для контекстно-зависимых грамматик и для эквивалентных им линейно ограниченных машин Тьюринга, что устанавливается в разделе 6.

2. Постановка задач распознавания свойств

Чтобы формулируемая нами задача распознавания свойств языков могла действительно считаться представлением задачи генерации строк в виде задачи распознавания, эти задачи должны быть связаны между собой в двух аспектах: с одной стороны, алгоритм решения задачи распознавания должен быть вынужден осуществлять в процессе своей работы фактическую генерацию строки, принадлежащей языку и удовлетворяющей данным условиям; с другой стороны, исходная задача генерации строки должна легко сводиться по Тьюрингу к вводимой задаче распознавания свойств, то есть иметь простое решение, использующее вызовы алгоритма решения задачи распознавания.

Как будет показано ниже, этим двум условиям лучше всего удовлетворяет задача определения существования строки, удовлетворяющей двум лексикографическим ограничениям; кроме того, будут рассмотрены четыре других формулировки задачи с более простыми ограничениями, которые для всех рассматриваемых нами классов вычислительных устройств будут иметь такую же сложность, как и задача с ограничениями общего вида.

Дадим формальное определение лексикографического порядка.

Определение 1 (Лексикографический порядок). Пусть конечный непустой алфавит Σ некоторым образом полностью упорядочен, то есть для любых двух различных символов $a, b \in \Sigma$ выполняется либо $a < b$, либо $b < a$.

Лексикографический порядок на множестве Σ^* определяется следующим образом.

- (1) Если $|u| < |v|$, то $u < v$.
- (2) Если $|u| = |v|$, $u = ax$ и $v = by$ ($a, b \in \Sigma$, $x, y \in \Sigma^*$), то
 - (а) если $a < b$, то $u < v$,

(b) если $a = b$, то $u < v$ в том и только в том случае, когда $x < y$.

Наконец, $u \preccurlyeq v$ тогда и только тогда, когда либо $u < v$, либо $u = v$.

Нетрудно видеть, что так определенное отношение лексикографического порядка на Σ^* является отношением полного порядка, то есть для любых различных строк $u, v \in \Sigma^*$ верно или $u < v$, или $v < u$. Более того, для каждой строки $u \in \Sigma^*$ однозначно определена лексикографически следующая строка $\text{next}(u) \in \Sigma^*$, и, таким образом, все элементы Σ^* могут быть записаны в виде одной лексикографически возрастающей последовательности.

Задача 1. Фиксируем класс вычислительных устройств $\mathcal{X}$. Пусть заданы упорядоченный конечный непустой алфавит Σ , вычислительное устройство X над алфавитом Σ , принадлежащее классу $\mathcal{X}$ и порождающее язык $L(X)$, и пара строк $u, v \in \Sigma^*$, $u \preccurlyeq v$. Требуется выяснить, существует ли строка $w \in \Sigma^*$ такая, что $u \preccurlyeq w \preccurlyeq v$ и $w \in L(X)$.

Определим язык входных данных, для которых задача имеет положительное решение:

$$\mathcal{P}(\mathcal{X}, \cdot \preccurlyeq w \preccurlyeq \cdot) = \{(X, u, v) \mid X \in \mathcal{X}, u \preccurlyeq w \preccurlyeq v \text{ для некоторой строки } w \in L(X)\}. \quad (1)$$

Имея алгоритм для решения такой задачи, легко написать программу, генерирующую строки методом двоичного поиска.

Алгоритм 1 (генерация строк с помощью $\mathcal{P}(\mathcal{X}, \cdot \preccurlyeq w \preccurlyeq \cdot)$). Пусть $\mathcal{X}$ — некоторый класс вычислительных устройств. Получив на входе запись устройства $X \in \mathcal{X}$, работающего над некоторым упорядоченным алфавитом Σ , и две строки $u, v \in \Sigma^*$ такие, что $u \preccurlyeq v$ и $(X, u, v) \in \mathcal{P}(\mathcal{X}, \cdot \preccurlyeq w \preccurlyeq \cdot)$, запустим с начальными параметрами (u, v) следующую рекурсивную процедуру:

```

generate( $u, v \in \Sigma^*$ )
{
    если  $u = v$ , то
        выдать строку  $u$  и завершить работу,
    выбрать строку  $w \in \Sigma^*$  такую, что  $u \preccurlyeq w \preccurlyeq v$  и
         $|\{x \mid u \preccurlyeq x \preccurlyeq w\}| = |\{x \mid u \preccurlyeq x \preccurlyeq v\}|/2$ 
    если  $(X, u, w) \in \mathcal{P}(\mathcal{X}, \cdot \preccurlyeq w \preccurlyeq \cdot)$ , то
        вернуть generate( $u, w$ ),
    иначе
        вернуть generate(next( $w$ ),  $v$ )
}
```

Процедура возвращает лексикографически наименьшую строку, находящуюся между u и v и принадлежащую $L(X)$. Нетрудно видеть, что потребуется не более $\lceil \log_2 |\{x \mid u \preccurlyeq x \preccurlyeq v\}| \rceil = O(|v|)$ обращений к алгоритму решения задачи $\mathcal{P}(\mathcal{X}, \cdot \preccurlyeq w \preccurlyeq \cdot)$.

Возможна также более простая постановка, в которой накладывается лишь лексикографическое ограничение сверху.

Задача 2. Пусть задан упорядоченный конечный непустой алфавит Σ , вычислительное устройство X над алфавитом Σ , принадлежащее заранее фиксированному классу $\mathcal{X}$ и порождающее язык $L(X)$, и строка $v \in \Sigma^*$. Требуется выяснить, существует ли строка $w \in \Sigma^*$ такая, что $w \preccurlyeq v$ и $w \in L(X)$.

Для этой задачи введем обозначение

$$\mathcal{P}(\mathcal{X}, w \preccurlyeq \cdot) = \{(X, v) \mid X \in \mathcal{X}, \exists w \in L(X): w \preccurlyeq v\}. \quad (2)$$

Недостаток этого подхода в том, что существование алгоритма для решения задачи $\mathcal{P}(\mathcal{X}, w \leq \cdot)$ позволяет генерировать лишь лексикографически наименьшую строку, принадлежащую языку (таким же двоичным поиском, как и в алгоритме 1), но не дает возможности построить никакие другие строки.

Рассмотрим еще более простые частные случаи задачи $\mathcal{P}(\mathcal{X}, \cdot \leq w \leq \cdot)$, в которых лексикографические ограничения заменены ограничениями длины, заданными в унарной записи.

Задача 3. Пусть задан упорядоченный конечный непустой алфавит Σ , вычислительное устройство X над алфавитом Σ , принадлежащее заранее фиксированному классу $\mathcal{X}$ и порождающее язык $L(X)$, и пара чисел l и m , $0 \leq l \leq m$. Требуется выяснить, существует ли строка $w \in \Sigma^*$ такая, что $l \leq |w| \leq m$ и $w \in L(X)$.

Для этой задачи введем обозначение

$$\mathcal{P}(\mathcal{X}, \cdot \leq |w| \leq \cdot) = \{(X, 1^l, 1^m) \mid X \in \mathcal{X}, \exists w \in L(X): l \leq |w| \leq m\}. \quad (3)$$

Задача 4. Пусть задан упорядоченный конечный непустой алфавит Σ , вычислительное устройство X над алфавитом Σ , принадлежащее заранее фиксированному классу $\mathcal{X}$ и порождающее язык $L(X)$, и число $m \geq 0$. Требуется выяснить, существует ли строка $w \in \Sigma^*$ такая, что $|w| \leq m$ и $w \in L(X)$.

Для этой задачи введем обозначение

$$\mathcal{P}(\mathcal{X}, |w| \leq \cdot) = \{(X, 1^m) \mid X \in \mathcal{X}, \exists w \in L(X): |w| \leq m\}. \quad (4)$$

Задача 5. Пусть задан упорядоченный конечный непустой алфавит Σ , вычислительное устройство X над алфавитом Σ , принадлежащее заранее фиксированному классу $\mathcal{X}$ и порождающее язык $L(X)$, и число $m \geq 0$. Требуется выяснить, существует ли строка $w \in \Sigma^m$ такая, что $w \in L(X)$.

Для этой задачи введем обозначение

$$\mathcal{P}(\mathcal{X}, |w| = \cdot) = \{(X, 1^m) \mid X \in \mathcal{X}, \exists w \in L(X): |w| = m\}. \quad (5)$$

Унарная запись длины необходима для поддержания связи с исходной задачей генерации строк, поскольку ее сложность разумно считать зависящей от суммарной длины записи данного на входе устройства и получаемой на выходе строки.

Задачи $\mathcal{P}(\mathcal{X}, \cdot \leq |w| \leq \cdot)$ и $\mathcal{P}(\mathcal{X}, |w| \leq \cdot)$ легко могут быть сведены к задачам $\mathcal{P}(\mathcal{X}, \cdot \leq w \leq \cdot)$ и $\mathcal{P}(\mathcal{X}, w \leq \cdot)$, соответственно, путем использования простой замены нижнего ограничения длины вида 1^l лексикографическим ограничением $u = (a_{\min})^l$, где $a_{\min}$ — наименьший элемент Σ , а верхнего ограничения длины вида 1^m — лексикографическим ограничением $v = (a_{\max})^m$, где $a_{\max}$ — наибольший элемент. С другой стороны, задачи $\mathcal{P}(\mathcal{X}, w \leq \cdot)$ и $\mathcal{P}(\mathcal{X}, |w| \leq \cdot)$ сводятся соответственно к $\mathcal{P}(\mathcal{X}, \cdot \leq w \leq \cdot)$ и $\mathcal{P}(\mathcal{X}, \cdot \leq |w| \leq \cdot)$ путем подстановки тривиального нижнего ограничения ε в случае лексикографического ограничения и 0 в случае ограничения длины. Наконец, задача $\mathcal{P}(\mathcal{X}, |w| = \cdot)$ сводится к $\mathcal{P}(\mathcal{X}, \cdot \leq |w| \leq \cdot)$, если положить нижнее ограничение равным верхнему.

Сводимости между рассмотренными нами задачами приведены на рис. 1. Заметим, что все эти сводимости являются DLOGSPACE-сводимостями, и потому могут быть использованы в доказательстве полноты для всех классов сложности, начиная с NLOGSPACE.

Оказывается, что для каждого из рассматриваемых в данной работе классов грамматик и автоматов все пять задач имеют одинаковую сложность. Всякий раз, когда мы будем устанавливать применительно к некоторому классу вычислительных устройств $\mathcal{X}$ полноту

Рис. 1. Сводимости между различными формулировками задачи.

всех пяти задач для некоторого класса сложности $\mathcal{C}$, мы будем доказывать $\mathcal{C}$ -трудность задач $\mathcal{P}(\mathcal{X}, |w| \leq \cdot)$ и $\mathcal{P}(\mathcal{X}, |w| = \cdot)$ и принадлежность задачи $\mathcal{P}(\mathcal{X}, \cdot \leq w \leq \cdot)$ классу $\mathcal{C}$. Отсюда, в силу указанных сводимостей, будет немедленно следовать, что все они $\mathcal{C}$ -трудны и все принадлежат классу $\mathcal{C}$.

3. Конечные автоматы

Рассмотрим сложность наших задач для различных видов конечных автоматов — детерминированных, недетерминированных и альтернирующих.

Определение 2. Детерминированным конечным автоматом (DFA) называется пятерка

$$A = (\Sigma, Q, q_0, \delta, F),$$

где Σ — конечный непустой алфавит, Q — конечное непустое множество состояний, $q_0 \in Q$ — начальное состояние, $\delta: Q \times \Sigma \rightarrow Q$ — функция перехода, $F \subseteq Q$ — множество допускающих состояний. Вычисление A на строке $w = a_1 \dots a_n$ — это последовательность из $n + 1$ состояний $q^{(0)}, q^{(1)}, \dots, q^{(n)}$, однозначно определяемая условиями $q^{(0)} = q_0$ и $q^{(i+1)} = \delta(q^{(i)}, a_i)$; строка принимается тогда и только тогда, когда $q^{(n)} \in F$.

Определение 3. Недетерминированным конечным автоматом (NFA) называется пятерка

$$A = (\Sigma, Q, s_0, \Delta, F),$$

где Σ — алфавит, Q — множество состояний, $s_0 \subseteq Q$ — множество начальных состояний, $\Delta: Q \times \Sigma \rightarrow 2^Q$ — функция перехода, $F \subseteq Q$ — множество допускающих состояний. Вычислением A на $w = a_1 \dots a_n$ называется всякая последовательность из $n + 1$ состояний $q^{(0)}, q^{(1)}, \dots, q^{(n)}$, в которой $q^{(0)} \in s_0$ и $q^{(i+1)} \in \Delta(q^{(i)}, a_i)$; строка принимается тогда и только тогда, когда существует вычисление, для которого $q^{(n)} \in F$.

Задача принадлежности DLOGSPACE-полна для детерминированных конечных автоматов [2, 11] и NLOGSPACE-полна для недетерминированных [3, 11]. Задача определения пустоты языка NLOGSPACE-полна для обоих классов автоматов [3, 11]. Докажем NLOGSPACE-полноту этих задач для детерминированных и недетерминированных конечных автоматов.

NLOGSPACE-трудность задач $\mathcal{P}(\text{DFA}, |w| \leq \cdot)$ и $\mathcal{P}(\text{NFA}, |w| \leq \cdot)$ следует из NLOGSPACE-полноты задачи пустоты для соответствующих классов автоматов; для этого

достаточно заметить, что принимаемый автоматом язык непуст тогда и только тогда, когда в графе, определяемом функцией перехода, существует путь из начального состояния в одно из принимающих. Так как длина этого пути не может превосходить длины записи всего автомата, возможно сведение задачи пустоты к $\mathcal{P}(\text{DFA}, |w| \leq \cdot)$ и $\mathcal{P}(\text{NFA}, |w| \leq \cdot)$, при котором по автомату A , пустоту которого требуется определить, строится пара $(A, |A|)$; очевидно, что язык $L(A)$ непуст тогда и только тогда, когда он содержит строки длины, не большей $|A|$.

NLOGSPACE-трудность задачи $\mathcal{P}(\text{DFA}, |w| = \cdot)$ доказывается чуть более сложным сведением задачи пустоты для детерминированных конечных автоматов: по исходному автомату $A = (\Sigma, Q, q_0, \delta, F)$ строится автомат

$$A' = (\Sigma, Q \cup \{q_{\text{acc}}\}, q_0, \delta', F'),$$

в котором

$$\begin{aligned} \delta'(q, a) &= \delta(q, a) && \text{для всех } q \in Q \setminus F \quad \text{и } a \in \Sigma, \\ \delta'(q, a) &= q_{\text{acc}} && \text{для всех } q \in F \cup \{q_{\text{acc}}\} \text{ и } a \in \Sigma, \\ F' &= F \cup \{q_{\text{acc}}\}, \end{aligned}$$

после чего нетрудно видеть, что построенный автомат A' принимает язык $L(A) \cdot \Sigma^*$, и потому язык $L(A)$ непуст тогда и только тогда, когда $L(A')$ содержит хотя бы одну строку длины $|A'|$.

Теперь используем тот факт, что недетерминированные конечные автоматы являются обобщением детерминированных автоматов, чтобы сделать вывод, что задача $\mathcal{P}(\text{NFA}, |w| = \cdot)$ также NLOGSPACE-трудна. Действительно, к ней тривиально сводится только что рассмотренная $\mathcal{P}(\text{DFA}, |w| = \cdot)$.

Построим недетерминированный алгоритм решения задачи $\mathcal{P}(\text{NFA}, \cdot \preceq w \preceq \cdot)$, использующий логарифмическую память, показывая таким образом принадлежность задачи классу NLOGSPACE.

Алгоритм 2. По недетерминированному конечному автомату $A = (\Sigma, Q, s_0, \Delta, F)$ определить, существует ли строка $w \in L(A)$, находящаяся лексикографически между данными строками u и v .

```

положить uflag=true, vflag=true
если  $s_0 = \emptyset$ 
    вернуть "нет"
недетерминированно выбрать  $q \in s_0$ 
для всех  $i$  от 0 до  $|v|$ 
{
    если  $q \in F \wedge (i > |u| \vee (i = |u| \wedge \text{uflag})) \wedge (i < |v| \vee (i = |v| \wedge \text{vflag}))$ 
        вернуть "да"
    если  $i = |v|$ 
        вернуть "нет"
    недетерминированно выбрать  $a \in \Sigma$ 
    если  $\Delta(a, q) = \emptyset$ 
        вернуть "нет"
    если  $i \leq |u| \wedge a > u_i$ 
        uflag=false
    если  $a < v_i$ 
        vflag=false
    недетерминированно выбрать  $q' \in \Delta(a, q)$ 
     $q = q'$ 
}

```

Запись u_i означает i -й символ строки u .

Нетрудно видеть, что алгоритм действительно занимается построением строки, удовлетворяющей условиям задачи, но за отсутствием памяти не может хранить ее всю одновременно. Фактически строка генерируется слева направо и сразу же по частям подается на вход встроенному алгоритму решения задачи принадлежности; одновременно осуществляется сравнение частей генерируемой строки с соответствующими частями строк u и v , и запоминаемый результат этого сравнения позволит в конце работы алгоритма лексикографически сравнить порожденную и уже забытую строку со строками u и v .

Так как $\mathcal{P}(\text{DFA}, \cdot \preceq w \preceq \cdot)$ — частный случай $\mathcal{P}(\text{NFA}, \cdot \preceq w \preceq \cdot)$, алгоритм 2 применим и к детерминированным конечным автоматам, откуда сразу следует, что $\mathcal{P}(\text{DFA}, \cdot \preceq w \preceq \cdot) \in \text{NLOGSPACE}$.

Перейдем к рассмотрению нашей задачи для класса альтернирующих конечных автоматов, где она будет иметь принципиально другое решение и другую оценку сложности.

Определение 4. Альтернирующим конечным автоматом (АФА) называется пятерка

$$A = (\Sigma, Q, q_0, g, F),$$

где Σ — алфавит, Q — конечное непустое множество состояний, $q_0 \in Q$ — начальное состояние, g — отображение, ставящее в соответствие каждой паре (состояние, символ) некоторую формулу алгебры логики $g[q, a]: \mathbf{B}^{|Q|} \rightarrow \mathbf{B}$, зависящую от $|Q|$ переменных, и $F \subseteq Q$ — множество допускающих состояний.

Строка w принимается из состояния $q \in Q$ тогда и только тогда, когда $w = \varepsilon$ и $q \in F$; или $w = au$, $a \in \Sigma$, $u \in \Sigma^*$, и выполняется условие

$$g[q, a](b_0, \dots, b_{|Q|-1}) = 1,$$

где

$$b_i = \begin{cases} 1, & \text{если строка } u \text{ принимается из состояния } q_i, \\ 0 & \text{в противном случае.} \end{cases} \quad (6)$$

Язык, порождаемый автоматом, — множество всех строк, принимаемых из состояния q_0 .

Альтернирующие конечные автоматы часто записывают при помощи систем языковых уравнений вида

$$X_i = \sum_{a \in \Sigma} g[q_i, a](X_0, \dots, X_{|Q|-1}) + \varepsilon_i, \quad (7)$$

где $\varepsilon_i = \varepsilon$, если $q_i \in F$, и $\varepsilon_i = \emptyset$ в противном случае, $i = 0 \dots |Q| - 1$ (см. [11]). Единственное решение такой системы — вектор из языков, принимаемых из каждого состояния автомата.

Несмотря на существенно расширенные описательные средства, класс языков, распознаваемых альтернирующими конечными автоматами, — класс регулярных языков. Известно, что задача принадлежности для альтернирующих конечных автоматов P-полна [2], а задача пустоты — PSPACE-полна [2]. Покажем что рассматриваемые нами задачи в случае альтернирующих конечных автоматов NP-полны.

NP-трудность задачи $\mathcal{P}(\text{AFA}, |w| \leq \cdot)$ устанавливается сведением к ней NP-полной задачи о выполнимости (satisfiability, SAT) формулы алгебры логики. По заданной (произвольной) формуле алгебры логики $\varphi(x_1, \dots, x_n)$ построим альтернирующий конечный автомат над алфавитом $\Sigma = \{0, 1\}$, заданный системой уравнений

$$\begin{aligned} X_0 &= 1 \cdot (\varphi(X_1, \dots, X_n) \wedge X'_n), & X_i &= 0 \cdot X'_{i-1} + 1 \cdot X'_{i-1}, & 2 \leq i \leq n, \\ X_1 &= 1 \cdot X_{\text{acc}}, & X_{\text{acc}} &= 0 \cdot X_{\text{acc}} + 1 \cdot X_{\text{acc}} + \varepsilon, \\ X'_i &= 0 \cdot X'_{i-1} + 1 \cdot X'_{i-1}, & 0 < i \leq n, & & X'_0 = \varepsilon. \end{aligned}$$

Нетрудно видеть, что i -е уравнение для переменной X'_i имеет решение Σ^i , в то время как уравнение для переменной X_i определяет язык $\Sigma^{i-1}1\Sigma^*$, $i = 1, \dots, n$. Таким образом, первое уравнение требует, чтобы строка принадлежала языку $1\Sigma^n$, и при этом формула φ была истинна на векторе, который образуют n последних цифр этой строки.

Стало быть, все строки языка, определяемого полученным автоматом, имеют вид $1x_1x_2\dots x_n$, $x_i \in \{0, 1\}$, и всякая такая строка принимается автоматом тогда и только тогда, когда $\varphi(x_1, \dots, x_n) = 1$. Отсюда, положив $m = n + 1$, легко свести задачу выполнимости формулы алгебры логики как к задаче $\mathcal{P}(\text{AFA}, |w| \leq \cdot)$, так и к задаче $\mathcal{P}(\text{AFA}, |w| = \cdot)$.

Принадлежность $\mathcal{P}(\text{AFA}, \cdot \leq w \leq \cdot)$ классу NP доказывается очевидным алгоритмом, вначале недетерминированно генерирующим некоторую строку, удовлетворяющую лексикографическим ограничениям, а затем использующим известный алгоритм для решения задачи принадлежности, чтобы установить, принимается ли эта строка данным на входе автоматом.

4. Контекстно-свободные и линейные контекстно-свободные грамматики

Определение 5. Контекстно-свободной (CF) грамматикой называется четверка

$$G = (\Sigma, N, P, S),$$

в которой Σ — конечный непустой алфавит терминальных символов, N — конечное множество нетерминальных символов, множество P состоит из правил вида $A \rightarrow \alpha$, $A \in N$, $\alpha \in (\Sigma \cup N)^*$, $S \in N$ — начальный символ грамматики.

Вывод — конечная последовательность строк $\gamma_1, \dots, \gamma_n$, $\gamma_i \in (\Sigma \cup N)^*$, $i = 1, \dots, n$, в которой $\gamma_1 = S$, $\gamma_n \in \Sigma^*$, γ_{i+1} получается из γ_i заменой некоторого нетерминала A на подстроку α (для некоторого правила $A \rightarrow \alpha \in P$); такой вывод называется выводом строки γ_n . Язык, порождаемый грамматикой, — множество всех терминальных строк, для которых существует вывод.

Контекстно-свободная грамматика называется линейной (LinCF), если каждое ее правило имеет вид

$$\begin{aligned} A &\rightarrow uBv, & u, v &\in \Sigma^*, & B &\in N, \\ A &\rightarrow w, & w &\in \Sigma^*. \end{aligned}$$

Рассмотрим вначале случай линейных контекстно-свободных грамматик, имеющих много общего с детерминированными и недетерминированными конечными автоматами.

Задача принадлежности для линейных контекстно-свободных грамматик NLOGSPACE-полна [9]; задача пустоты имеет такую же сложность. Докажем NLOGSPACE-полноту нашей задачи.

Задача $\mathcal{P}(\text{NFA}, |w| \leq \cdot)$ легко сводится к $\mathcal{P}(\text{LinCF}, |w| \leq \cdot)$, поскольку недетерминированные конечные автоматы изоморфны подмножеству линейных контекстно-свободных грамматик, в которых для каждого правила вида $A \rightarrow uBv$ строка v пуста. Отсюда следует NLOGSPACE-трудность $\mathcal{P}(\text{LinCF}, |w| \leq \cdot)$. Возможно также прямое сведение задачи пустоты для самих линейных контекстно-свободных грамматик.

Приведенный ниже недетерминированный алгоритм 3 для решения задачи $\mathcal{P}(\text{LinCF}, \cdot \leq w \leq \cdot)$, использующий логарифмическую память, обобщает идею алгоритма 2, решающего задачу $\mathcal{P}(\text{NFA}, \cdot \leq w \leq \cdot)$. Различие состоит в том, что в линейном контекстно-свободном случае приходится генерировать, сравнивать с u и v , и сразу же по частям распознавать строку с двух концов одновременно.

Алгоритм 3. Пусть $G = (\Sigma, N, P, S)$ — поданная на вход алгоритма линейная контекстно-свободная грамматика, $u, v, u \preceq v$ — нижнее и верхнее лексикографические ограничения. Не ограничивая общности, предположим, что грамматика содержит лишь правила вида $A \rightarrow xBy$ и вида $A \rightarrow \varepsilon$. Действительно, правила вида $A \rightarrow w$ ($w \in \Sigma^+$) могут рассматриваться как правила вида $A \rightarrow wE$, где E — дополнительный нетерминал, имеющий единственное правило $E \rightarrow \varepsilon$. Поэтому это дополнительное требование упрощает лишь формулировку алгоритма.

```

положить uflagl=true, uflagr=true, vflagl=true, vflagr=true
положить A = S
положить i = 0, j = 0
{
    если  $A \rightarrow \varepsilon \in P \wedge (i + j > |u| \vee (i + j = |u| \wedge \text{uflagl} \wedge \text{uflagr})) \wedge$ 
        $(i + j < |v| \vee (i + j = |v| \wedge \text{vflagl} \wedge \text{vflagr}))$ 
        вернуть "да"
    недетерминированно выбрать  $A \rightarrow xBy \in P$ 
    если  $i + j + |x| + |y| \geq |v|$ 
        вернуть "нет"
    if  $i + |x| \leq |u| \wedge x \preceq u_{i+1} \dots u_{i+|x|}$ 
        uflagl=false
    если  $j + |y| \leq |u| \wedge y \preceq u_{|u|-j-|y|+1} \dots u_{|u|-j}$ 
        uflagr=false
    если  $i + |x| \leq |v| \wedge x \succeq v_{i+1} \dots v_{i+|x|}$ 
        vflagl=false
    если  $j + |y| \leq |v| \wedge y \succeq v_{|v|-j-|y|+1} \dots v_{|v|-j}$ 
        vflagr=false
    i = i + |x|, j = j + |y|
}

```

Перейдем к случаю контекстно-свободных грамматик общего вида. Задача принадлежности и задача пустоты для этого семейства P-полны [10]; возникает соблазн решать задачу $\mathcal{P}(\text{CF}, \cdot \leq w \leq \cdot)$ недетерминированно-полиномиальным алгоритмом, как и в случае альтернирующих конечных автоматов. Оказывается, однако, что для нее существует детерминированный полиномиальный алгоритм, и она P-полна.

Прежде чем приступить к решению задачи $\mathcal{P}(\text{CF}, \cdot \leq w \leq \cdot)$, решим вначале ее частный случай, задачу $\mathcal{P}(\text{CF}, w \leq \cdot)$. Методы решения этой задачи будут затем расширены для общего случая. Выведем из каждого нетерминала лексикографически наименьшую строку.

Алгоритм 4. Дана контекстно-свободная грамматика $G = (\Sigma, N, P, S)$; не ограничивая общности, предположим, что грамматика находится в нормальной форме Хомского (пре-

образование грамматики к нормальной форме Хомского может быть проделано за полиномиальное время [1]). Установить, существует ли строка $w \in L(G)$, лексикографически предшествующая данной строке v .

Пусть $s[\cdot]$ — массив из элементов типа $\Sigma^* \cup \{\text{нет}\}$, индексируемый нетерминалами.

```

если  $S \rightarrow \varepsilon \in P$ 
    вернуть true
для всех  $A \in N$ 
    если  $A \rightarrow a_i \in P$  для некоторого  $i$ 
         $s[A] = \min_{i: A \rightarrow a_i \in P} a_i$ ;
    иначе
         $s[A] = \text{нет}$ ;
пока  $s$  можно изменить
    для всех  $A \in N$ 
         $s[A] = \min \left( s[A], \min_{\substack{A \rightarrow BC \in P, \\ s[B], s[C] \neq \text{нет}}} s[B] \cdot s[C] \right)$ ;

```

После построения массива s задача $\mathcal{P}(\text{CF}, w \leq \cdot)$ решается проверкой $s[S] \neq \text{нет}$ и затем еще одной проверкой $s[S]$ на лексикографическое предшествование полученной на входе строке v .

Используемая в алгоритме операция взятия минимума вычисляет лексикографически наименьший элемент конечного языка; в случае пустоты языка результатом операции является нет; $\min(\text{нет}, x)$ также определяется как нет.

Теперь построим алгоритм для решения общей задачи $\mathcal{P}(\text{CF}, \cdot \leq w \leq \cdot)$. Для этого потребуется работать с более сложными структурами данных.

Если $|u| < |v|$, то для каждого нетерминала A и каждого числа i , $1 \leq i \leq |v|$, построим лексикографически наименьшую и лексикографически наибольшую строки длины i , порождаемые нетерминалом A .

Если $|u| = |v|$, то достаточно построить лексикографически наибольшую строку, порождаемую грамматикой и лексикографически не превосходящую v . Для этого для каждого нетерминала и каждой подстроки x данной строки v построим лексикографически наибольшую строку, которая лексикографически меньше или равна v и порождается этим нетерминалом.

Алгоритм 5. Пусть $G = (\Sigma, N, P, S)$ — контекстно-свободная грамматика в нормальной форме Хомского. Установить, существует ли строка $w \in L(G)$, находящаяся лексикографически между данными строками u и v .

В случае $|u| < |v|$ построим массивы $\underline{s}[\cdot][\cdot]$ и $\bar{s}[\cdot][\cdot]$ элементов типа $\Sigma^* \cup \{\text{нет}\}$, индексируемые парами (нетерминал, длина):

```

для всех  $A \in N$ 
{
     $\underline{s}[A][1] = \min_{i: A \rightarrow a_i \in P} a_i$ ;
     $\bar{s}[A][1] = \max_{i: A \rightarrow a_i \in P} a_i$ ;
}
для всех  $i$  от 2 до  $|v|$ 
    для всех  $A \in N$ 
    {
         $\underline{s}[A][i] = \min_{\substack{i: 1 \leq i < j, \\ B, C: A \rightarrow BC \in P, \\ \underline{s}[B][j], \underline{s}[C][j-i] \neq \text{нет}}} \underline{s}[B][j] \cdot \underline{s}[C][j-i];$ 
         $\bar{s}[A][i] = \max_{\substack{i: 1 \leq i < j, \\ B, C: A \rightarrow BC \in P, \\ \bar{s}[B][j], \bar{s}[C][j-i] \neq \text{нет}}} \bar{s}[B][j] \cdot \bar{s}[C][j-i];$ 
    }
}

```

После построения массивов $\underline{s}[\cdot][\cdot]$, $\bar{s}[\cdot][\cdot]$, искомая строка w , если она существует, будет находиться среди $\bar{s}[S][|u|]$, $\underline{s}[S][i]$, $|u| < i < |v|$, $\bar{s}[S][i]$, $|u| < i < |v|$, и $\underline{s}[S][|v|]$.

Если $|u| = |v|$, то, используя уже построенный массив $\bar{s}[A][i]$, построим массив $s'[A][x]$ элементов типа $\Sigma^* \cup \{\text{нет}\}$, индексируемый парой (нетерминал, подстрока строки v):

$$\begin{aligned} &\text{для всех } A \in N \\ &\quad \text{для всех } a \in \Sigma: v = xay \\ &\quad \quad s'[A][a] = \max_{b: A \rightarrow b \in P, b < a} b; \\ &\text{для всех подстрок } x \text{ строки } v \text{ (в лексикографическом порядке)} \\ &\quad \text{для всех } A \in N \\ &\quad \quad s'[A][x] = \max_{\substack{y, z: x = yz, \\ B, C: A \rightarrow BC \in P}} \left(\max_{\substack{s'[B][y] = y \\ s'[C][z] \neq \text{нет}}} y \cdot s'[C][z], \max_{\substack{s'[B][y] \neq \text{нет}, \\ s'[B][y] < y \\ \bar{s}[C][|z|] \neq \text{нет}}} s'[B][y] \cdot \bar{s}[C][|z|] \right); \end{aligned}$$

Теперь, если $s'[S][v] \neq \text{нет}$, то $s'[S][v]$ — лексикографически наибольшая строка длины $|v|$, лексикографически не превосходящая строку v и порождаемая грамматикой. Положительный ответ дается в том и только в том случае, когда $s'[S][v] \neq \text{нет}$ и $s'[S][v] \succ u$.

Остается доказать Р-трудность задач $\mathcal{P}(\text{CF}, |w| \leq \cdot)$ и $\mathcal{P}(\text{CF}, |w| = \cdot)$. Для этого используем тот известный факт, что задача принадлежности пустой строки языку (то есть, задача, состоящая в выяснении, порождает ли пустую строку данная на вход контекстно-свободная грамматика) является Р-полной, и сведем ее отдельно к каждой из задач $\mathcal{P}(\text{CF}, |w| \leq \cdot)$ и $\mathcal{P}(\text{CF}, |w| = \cdot)$, положив верхнее ограничение длины или точную длину строки равными нулю.

5. Конъюнктивные и линейные конъюнктивные грамматики

Конъюнктивные грамматики, представляющие собой расширение контекстно-свободных грамматик явной операцией пересечения, были введены автором в [4, 5].

Определение 6. Конъюнктивная грамматика — четверка

$$G = (\Sigma, N, P, S),$$

где Σ , N и S — такие же, как в контекстно-свободных грамматиках, а правила из P имеют вид

$$A \rightarrow \alpha_1 \& \dots \& \alpha_n, \quad \alpha_i \in (\Sigma \cup N)^*, \quad n \geq 1. \quad (8)$$

Конъюнктивная грамматика G называется линейной, если всякое правило (8) имеет вид

$$\begin{aligned} A &\rightarrow u_1 B_1 v_1 \& \dots \& u_n B_n v_n, \quad u_i, v_i \in \Sigma^*, \quad B_i \in N, \quad i = 1, \dots, n, \\ A &\rightarrow w, \quad w \in \Sigma^*. \end{aligned}$$

Язык, порождаемый грамматикой, определяется с помощью вывода, в котором на каждом шаге выполняется одно из двух действий:

- нетерминал заменяется телом правила, заключенным в скобки,
- подстрока вида $(w \& \dots \& w)$, где $w \in \Sigma^*$, то есть конъюнкция нескольких одинаковых терминальных строк, заменяется одной строкой w .

Семантика конъюнктивных грамматик хорошо описывается представлением их в виде систем языковых уравнений с операцией пересечения [7]

$$A = \bigcup_{A \rightarrow \alpha_1 \& \dots \& \alpha_n \in P} \bigcap_{i=1}^n \alpha_i \quad (9)$$

для всех $A \in N$.

Известно, что вектор языков, порождаемых нетерминалами грамматики, совпадает с наименьшим решением соответствующей ей системы.

Задача принадлежности для конъюнктивных грамматик Р-полна [6, 8], как и аналогичная задача для контекстно-свободных грамматик. Задача принадлежности для линейных конъюнктивных грамматик также Р-полна [8], что отличает их от линейных контекстно-свободных грамматик. Задача определения пустоты языка неразрешима для обоих классов конъюнктивных грамматик.

Известно, что все основные алгоритмы распознавания и синтаксического анализа для контекстно-свободных грамматик имеют аналоги для случая конъюнктивных грамматик (см., например, краткий обзор, данный в [7]), которые в большинстве случаев сохраняют свою сложность. Возникает вопрос, не может ли еще один основанный на методе динамического программирования алгоритм анализа свойств грамматик, а именно, построенный в разделе 4 алгоритм решения задачи $\mathcal{P}(\text{CF}, \cdot \leq w \leq \cdot)$, быть обобщен на случай конъюнктивных грамматик с сохранением полиномиальности времени выполнения. Оказывается, однако, что существование такого обобщения означало бы совпадение классов сложности Р и NP, представляющееся маловероятным; покажем, что рассматриваемая нами задача для случая конъюнктивных грамматик NP-полна.

Принадлежность задачи $\mathcal{P}(\text{Conj}, \cdot \leq w \leq \cdot)$ классу NP очевидна, поскольку достаточно недетерминированно сгенерировать строку, удовлетворяющую данным лексикографическим ограничениям и затем использовать существующий полиномиальный алгоритм решения задачи принадлежности. Докажем NP-трудность задач $\mathcal{P}(\text{Conj}, |w| \leq \cdot)$ и $\mathcal{P}(\text{Conj}, |w| = \cdot)$, полиномиально сведя к каждой из них NP-полную задачу о выполнимости конъюнктивной нормальной формы.

Для этого рассмотрим произвольную КНФ

$$\&_{i=1}^n (x_{k_{i1}}^{\sigma_{i1}} \vee \dots \vee x_{k_{i l_i}}^{\sigma_{i l_i}}), \quad (10)$$

зависящую от переменных $x_1, \dots, x_m$, и построим по ней грамматику

$$G = (\{0, 1\}, \{S, A_1, \dots, A_n, X\}, P, S), \quad (11)$$

состоящую из следующих правил:

- (1) одно правило для S

$$S \rightarrow A_1 \& \dots \& A_n,$$

- (2) для каждой i -й дизъюнкции длины l_i , входящей в (10), грамматика содержит l_i правил

$$A_i \rightarrow X \dots X \sigma_{ij} X \dots X, \quad (12)$$

где вначале стоит $k_{ij} - 1$, а в конце $m - k_{ij}$ символов X и $j = 1, \dots, l_i$, $i = 1, \dots, n$,

- (3) два правила для X

$$X \rightarrow 0, \quad X \rightarrow 1.$$

Лемма 1. Пусть (10) — некоторая КНФ, зависящая от m переменных,

$$G = (\{0, 1\}, \{S, A_1, \dots, A_n, X\}, P, S)$$

— грамматика, построенная на основе данной КНФ указанным выше способом.

Тогда строка $w = b_1 b_2 \dots b_t \in \mathbf{B}^t$ принадлежит $L(G)$ тогда и только тогда, когда $t = n$ и исходная КНФ принимает значение 1 на наборе $\bar{b} = (b_1, \dots, b_n)$.

Доказательство леммы 1 не представляет трудности.

Из леммы следует, что КНФ выполнима тогда и только тогда, когда язык, порождаемый соответствующей грамматикой, непуст. Учитывая, что язык не содержит ни одной строки длины, отличной от m , можно заключить, что КНФ выполнима, с одной стороны, тогда и только тогда, когда языку $L(G)$ принадлежит хотя бы одна строка длины меньшей или равной m ; с другой стороны, тогда и только тогда, когда языку $L(G)$ принадлежит хотя бы одна строка длины, равной m . Это утверждение завершает доказательство NP-полноты всех пяти рассматриваемых нами задач для конъюнктивных грамматик.

Задача остается NP-полной для важного подкласса линейных конъюнктивных грамматик. Докажем NP-трудность задач $\mathcal{P}(\text{LinConj}, |w| \leq \cdot)$ и $\mathcal{P}(\text{LinConj}, |w| = \cdot)$, сведя к ним SAT. Построение в целом повторяет построение для случая конъюнктивных грамматик общего вида; достаточно удалить из приведенной выше грамматики нетерминал X и заменить построение (12) на следующее построение:

$$\begin{aligned} A_i &\rightarrow B_{i,j,0}, & B_{i,j,t-1} &\rightarrow 0B_{i,j,t} \mid 1B_{i,j,t} \quad \text{если } 1 \leq t \leq m \text{ и } t \neq k_{ij}, \\ B_{i,j,k_{ij}-1} &\rightarrow \sigma_{ij} B_{i,j,k_{ij}}, & B_{i,j,m} &\rightarrow \varepsilon. \end{aligned}$$

Полученная линейная конъюнктивная грамматика будет порождать тот же язык, что и исходная грамматика, и потому приведенное выше обоснование корректности сводимости будет справедливо и в этом случае.

6. Контекстно-зависимые грамматики

Определение 7. Контекстно-зависимой (CS) грамматикой называется четверка

$$G = (\Sigma, N, P, S),$$

в которой $S \in N$, а множество P состоит из правил вида $\alpha \rightarrow \beta$, $\alpha, \beta \in (\Sigma \cup N)^+$, $|\alpha| \leq |\beta|$, или $S \rightarrow \varepsilon$, если S не появляется в правых частях правил.

Вывод — конечная последовательность строк $\gamma_1, \dots, \gamma_n$, $\gamma_i \in (\Sigma \cup N)^*$, $i = 1, \dots, n$, в которой $\gamma_1 = S$, $\gamma_n \in \Sigma^*$, γ_{i+1} получается из γ_i заменой подстроки α на подстроку β для некоторого правила $\alpha \rightarrow \beta \in P$; такой вывод называется выводом строки γ_n . Язык, порождаемый грамматикой, — множество всех терминальных строк, для которых существует вывод.

Задача принадлежности для контекстно-зависимых грамматик PSPACE-полна, а вопрос о пустоте неразрешим.

Хорошо известно [1], что контекстно-зависимые грамматики эквивалентны линейно ограниченным машинам (LBA). Построение эквивалентной грамматики по машине Тьюринга и машины Тьюринга по грамматике сводится к простым синтаксическим преобразованиям и потому может быть проделано за полиномиальное время.

Докажем PSPACE-полноту всех пяти наших задач как для контекстно-зависимых грамматик, так и для линейно ограниченных машин Тьюринга. Принадлежность задачи $\mathcal{P}(\text{LBA}, \cdot \leq w \leq \cdot)$ классу PSPACE доказывается построением очевидного алгоритма, который перебирает все строки, находящиеся лексикографически между данными строками u и v и для каждой из них моделирует выполнение полученной на входе линейно ограниченной машины Тьюринга. Задача $\mathcal{P}(\text{CS}, \cdot \leq w \leq \cdot)$ может быть решена с использованием полиномиальной памяти простым сведением ее к $\mathcal{P}(\text{LBA}, \cdot \leq w \leq \cdot)$.

PSPACE-трудность задач $\mathcal{P}(\text{LBA}, |w| \leq \cdot)$ и $\mathcal{P}(\text{LBA}, |w| = \cdot)$ доказывается сведением задачи принадлежности для контекстно-зависимых грамматик. Получив на входе контекстно-зависимую грамматику G над некоторым алфавитом Σ и строку $w \in \Sigma^*$, вначале построим линейно ограниченную машину Тьюринга, эквивалентную G , а затем, на основе M и w — линейно ограниченную машину Тьюринга M' , выполняющую следующие действия:

```

M'(u ∈ Σ*)
{
    Если u ≠ w
        отвергнуть
    иначе
    {
        выполнить M(u)
        принять, если M(u) принимает; отвергнуть, если M(u) отвергает
    }
}

```

Теперь $(M', |w|) \in \mathcal{P}(\text{LBA}, |w| \leq \cdot)$ тогда и только тогда, когда на некоторой строке u , $|u| \geq |w|$, в M' выполняется вторая ветвь и вычисление $M(u)$ успешно, то есть тогда и только тогда, когда успешно вычисление $M(w)$, что и означает $w \in L(G)$. Аналогично доказывается, что $(M', |w|) \in \mathcal{P}(\text{LBA}, |w| = \cdot)$ в том и только в том случае, когда M принимает строку w .

PSPACE-трудность задач $\mathcal{P}(\text{CS}, |w| \leq \cdot)$ и $\mathcal{P}(\text{CS}, |w| = \cdot)$ легко доказывается сведением к ним $\mathcal{P}(\text{LBA}, |w| \leq \cdot)$ и $\mathcal{P}(\text{LBA}, |w| = \cdot)$ соответственно.

7. Заключение

Было предложено представление задачи генерации строк в виде задачи распознавания свойств и обоснована тесная взаимосвязь между сложностью сформулированной задачи распознавания и сложностью исходной задачи. Получены оценки сложности предложенной задачи для ряда формализмов задания языков. В соответствии с этими оценками, рассмотренные формализмы можно разделить на несколько приведенных ниже классов.

Таблица 1. Сложность задач для автоматов и грамматик, подразумевается полнота для указанных классов

	Принадлежность	Пустота	Рассматриваемая задача
DFA	DLOGSPACE	NLOGSPACE	NLOGSPACE
NFA	NLOGSPACE	NLOGSPACE	NLOGSPACE
AFA	P	PSPACE	NP
LinCF	NLOGSPACE	NLOGSPACE	NLOGSPACE
CF	P	P	P
Conj	P	неразрешима	NP
LinConj	P	неразрешима	NP
CS	PSPACE	неразрешима	PSPACE
LBA	PSPACE	неразрешима	PSPACE

- Слабые по мощности вычислительные устройства с задачей принадлежности, разрешимой с использованием логарифмической памяти, и проблемой пустоты, являющейся переформулировкой задачи достижимости в ориентированном графе (детерминированные и недетерминированные конечные автоматы, линейные контекстно-свободные грамматики). Для таких устройств рассмотренная задача является NLOGSPACE-полной; она может быть решена генерацией строки по частям, подачей этих частей на вход алгоритму, решающему задачу принадлежности, и одновременной проверкой выполнения лексикографических ограничений.
- Устройства с полиномиально разрешимой задачей принадлежности, выразительные средства которых включают операции объединения и пересечения (альтернирующие конечные автоматы, конъюнктивные и линейные конъюнктивные грамматики). Для таких устройств задача NP-полна, поскольку в их формализме может быть выражена формула алгебры логики (в виде языка всех двоичных векторов, на которых соответствующая функция принимает значение 1) и к существованию строки определенной длины может быть сведена задача о выполнимости формулы алгебры логики; построение недетерминированного алгоритма генерации строки очевидно.
- Устройства с большими выразительными возможностями, задача принадлежности для которых PSPACE-полна (контекстно-зависимые грамматики, линейно ограниченные автоматы). Следующий отсюда тривиальный NPSPACE-алгоритм решения нашей задачи детерминируется с использованием известного результата о совпадении классов PSPACE и NPSPACE. С другой стороны, вычислительные возможности этих устройств позволяют свести задачу принадлежности к задаче существования строки заданной длины, принимаемой автоматом.
- Случай контекстно-свободных грамматик стоит особняком. Вместо очевидного недетерминированно полиномиального алгоритма, решающего рассмотренную задачу, можно построить детерминированный алгоритм, основанный на идее метода динамического программирования. С другой стороны, задача принадлежности пустой строки языку, порождаемому контекстно-свободной грамматикой, уже P-полна и может быть легко сведена к рассмотренной задаче.

Автор выражает свою признательность рецензенту за ценные замечания.

Список литературы

1. Harrison M. A., *Introduction to formal language theory*. Addison–Wesley, Reading, Mass., 1978.
2. Jiang T, Ravikumar B., A note on the space complexity of some decision problems for finite automata. *Inform. Process. Lett.* (1991) **40**, 25–31.
3. Jones N., Space-bounded reducibility among combinatorial problems, *J. Comp. System Sci.* (1975) **11**, 68–85.
4. Охотин А. С., О расширении формализма контекстно-свободных грамматик операцией пересечения. *Труды IV Международной конференции «Дискретные модели в теории управляющих систем»*. 2000, с. 106–109.
5. Okhotin A., Conjunctive grammars. *J. Automata, Languages and Combinatorics* (2001) **4**, 519–535.
6. Охотин А. С., О Р-полноте задачи принадлежности для конъюнктивных грамматик. *Дискретная математика и математическая кибернетика. Труды международной школы-семинара*. Патмино, 2001.
7. Охотин А. С., Конъюнктивные грамматики и системы языковых уравнений. *Программирование* (2002) **28**, №5, 243–249.
8. Okhotin A., A recognition and parsing algorithm for arbitrary conjunctive grammars. *Theoretical Computer Sci.* (2003) **302**, 365–399.
9. Sudborough I. H., A note on tape-bounded complexity classes and linear context-free languages. *J. ACM* (1975) **22**, 499–500.
10. Vollmer H., *Introduction to circuit complexity*. Springer, Berlin, 1999.
11. Yu S., Regular languages. In: *Handbook of formal languages*, v. 1. Springer, Berlin, 1997, pp. 41–110.

Статья поступила 10.06.2002.