Синтаксический анализ графов и задача генерации строк с ограничениями

Рустам Азимов, Семён Григорьев
Санкт-Петербургский государственный университет,
Россия, 199034, Санкт-Петербург, Университетская наб. 7/9/
rustam.azimov19021995@gmail.com, Semen.Grigorev@jetbrains.com

6 февраля 2017 г.

Аннотация

Одной из задач, изучаемых в теории формальных языков и автоматов, является задача генерации строк, удовлетворяющих заданной системе правил. С другой стороны, существует задача синтаксического анализа графов, то есть поиска определенных путей в графе, метки на ребрах которых образуют строку, принадлежащую заданному формальному языку. В данной работе будет показана связь между этими двумя задачами.

Ключевые слова: синтаксический анализ графов, генерация строк, формальные языки, конъюнктивные грамматики.

Введение

В таких областях, как графовые базы данных [6, 3], биоинформатика [1], возникают задачи поиска путей в графах, удовлетворяющих определенным условиям. Примерами простых условий являются ограничения на длину или тип искомых путей. Но при работе со сложными системами зачастую таких ограничений бывает недостаточно. Поэтому широко распространено использование ограничений на метки ребер/вершин путей помеченного графа. В качестве таких ограничений естественно выбрать формальный язык L [2]. Тогда при заданном алфавите Σ и ориентированном графе G, ребра которого помечены символами из Σ, для искомых путей p графа G выполняется $l(p) \in L$, где $l(p)$ означает слово из Σ^*, полученное последовательной конкатенацией меток пути p. Задача поиска путей в графе, которые используют такие ограничения с формальными языками, называется задачей синтаксического анализа графов.

Кроме того, существует задача генерации строк, суть которой в построении строк, принадлежащих некоторому формальному языку. В работе [11]

получены оценки сложности задачи генерации строк с дополнительными ограничениями для различных классов формальных языков.

Некоторые вариации задач синтаксического анализа графов могут быть сведены к задаче генерации строк. Так, например, в большинстве задач синтаксического анализа графов недостаточно просто определить существование пути, соответствующего строке некоторого формального языка L, но также требуется предъявить такой путь. Так как все пути в графе соответствуют строкам из некоторого регулярного языка R, то в данной задаче требуется найти путь, соответствующий строке из языка $L \cap R$. Эта задача может быть решена с помощью генератора строк рассматриваемого пересечения языков.

В данной работе исследуется связь задачи генерации строк и некоторых типов задач синтаксического анализа графов. В качестве формальных языков будут рассматриваться широко используемые в области синтаксического анализа графов контекстно-свободные языки. Также будут рассмотрены конъюнктивные [7] языки, обладающие большей выразительной мощностью.

1 Синтаксический анализ графов

В данном разделе будут даны основные определения из области синтаксического анализа графов, и рассмотрены различные формулировки задач этой области с использованием контекстно-свободных грамматик.

В данной работе пустую строку будем обозначать ε, а конкатенацию двух строк S_1 и S_2 — $S_1 \cdot S_2$.

Определение 1 *Граф — это тройка $G = (Q, \Sigma, \delta)$, где $Q \cap \Sigma = \emptyset$, Q — конечное множество вершин графа, Σ — конечный алфавит символов, используемых в качестве меток на ребрах графа, и $\delta \subseteq Q \times \Sigma \times Q$ — конечное множество ребер, помеченных символами из Σ.*

Если $m \in Q$ и $\sigma \in \Sigma$, тогда $\delta(m, \sigma) = \{n \mid (m, \sigma, n) \in \delta\}$ обозначает множество вершин графа, которые имеют входящее ребро из вершины m, помеченное символом σ. Если $m \in Q$ и $S \in \Sigma^*$, то

$$\delta^*(m, S) = \begin{cases} \{m\} & \text{if } S = \varepsilon; \\ \cup_{n \in \delta(m,\sigma)} \delta^*(n, S') & \text{if } S = \sigma \cdot S'. \end{cases}$$

Язык, который порождается графом G и выделенными в нем вершинами $m, n \in Q$, обозначим $L(G, m, n)$, где

$$L(G, m, n) = \{S \in \Sigma^* \mid n \in \delta^*(m, S)\}.$$

В дальнейшем будут рассматриваться контекстно-свободные грамматики, которые находятся в нормальной форме Хомского [10], и в которых исключается вывод пустой строки. Для простоты представления введем следующее определение.

Определение 2 *Контекстно-свободная грамматика — это тройка* $C = (N, \Sigma, P)$*, где* $N \cap \Sigma = \emptyset$*,* N *— конечное множество нетерминалов грамматики,* Σ *— конечный алфавит символов,* P *— множество правил грамматики. Кроме того, все правила грамматики записываются в одной из следующих форм:* $a \to b\ c$ *или* $a \to \sigma$*, где* $a, b, c \in N$ *и* $\sigma \in \Sigma$*.*

В данном определении не указывается стартовый нетерминал грамматики, поэтому для определения языка, порождаемого данной грамматикой, необходимо указать соответствующий нетерминал. Тогда язык, порождаемый грамматикой C, со стартовым нетерминалом $a \in N$ обозначим $L(C, a)$.

В контексте задач синтаксического анализа графов бывает необходимо отвечать на различного рода вопросы, связанные с искомыми в графе путями. Тип вопросов, на которые отвечает задача принято называть *семантикой запроса*. Далее сформулируем задачи синтаксического анализа графов с использованием КС-грамматики C и различных семантик запросов, рассмотренных в работах [4, 5].

Использование *relational* семантики запроса означает, что для нетерминала $a \in N$ и графа G необходимо построить множество $\{(m, n) \mid L(C, a) \cap L(G, m, n) \neq \emptyset\}$.

Использование *all-path* семантики запроса означает, что для нетерминала $a \in N$, графа G и его вершин $m, n \in Q$, необходимо предъявить все пути из вершины m в вершину n, такие что метки на ребрах этих путей образуют строку из языка $L(C, a)$.

Использование *single-path* семантики запроса означает, что для нетерминала $a \in N$, графа G и его вершин $m, n \in Q$, необходимо предъявить какой-нибудь путь (если он существует) из вершины m в вершину n, такой что метки на ребрах этого пути образуют строку из языка $L(C, a)$.

В качестве грамматики, порождающей язык $L(C, a) \cap L(G, m, n)$, в работе [5] используется *аннотированная грамматика*, определенная следующим образом.

Определение 3 *Пусть имеется КС-грамматика* $C = (N, \Sigma, P)$ *и граф* $G = (Q, \Sigma, \delta)$*. Тройки* $(a, m, n) \in N \times Q \times Q$ *будем обозначать* $a[m, n]$*. Тогда аннотированная грамматика — это КС-грамматика* $C_G = (N_G, \Sigma, P_G)$*, в которой* $N_G \subseteq N \times Q \times Q$*; каждое правило из* P_G *вида* $a[m, n] \to b[m, o]\ c[o, n]$ *или вида* $a[m, n] \to \sigma$*, где* $a, b, c \in N$*,* $m, n, o \in Q$ *и* $\sigma \in \Sigma$*; и выполняются следующие три правила:*

1. $a[m, n] \in N_G \Leftrightarrow L(C, a) \cap L(G, m, n) \neq \emptyset$,
2. $(a[m, n] \to b[m, o]\ c[o, n]) \in P_G \Leftrightarrow (a \to b\ c) \in P$,
3. $(a[m, n] \to \sigma) \in P_G \Leftrightarrow (m, \sigma, n) \in \delta \wedge (a \to \sigma) \in P$.

2 Конъюнктивные грамматики

В работе будут рассматриваться конъюнктивные грамматики [7], которые представляют собой расширение контекстно-свободных грамматик опера-

цией пересечения.

Определение 4 *Конъюнктивная грамматика — это четверка* $C = (\Sigma, N, P, S)$*, где* $N \cap \Sigma = \emptyset$*,* N *— конечное множество нетерминалов грамматики,* Σ *— конечный алфавит символов,* P *— множество правил грамматики, а* S *— стартовый нетерминал. Кроме того, все правила грамматики имеют вид* $a \to \alpha_1 \,\&\ldots\&\, \alpha_n$*, где* $a \in N$*,* $\alpha_i \in (\Sigma \cup N)^*$*,* $n \geq 1$*.*

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

- нетерминал заменяется телом правила, заключенным в скобки;
- подстрока вида $(w \,\&\ldots\&\, w)$, где $w \in \Sigma^*$, заменяется одной строкой w.

3 Сложность задачи генерации строк

В данном разделе будут рассмотрены формулировки задачи генерации строк, приведенные в работе [11], и оценки сложности этих задач.

Далее рассмотрим формулировки задачи генерации строк, приведенные в работе [11]. В данной работе генерируемую строку ограничивают с одной или с двух сторон либо длиной, либо лексикографическим отношением $(\preceq, \succeq)$ с другой строкой. Всего приведены пять формулировок задачи генерации строк, использующие такого рода ограничения. Эти задачи и сводимости между ними приведены на рис. 1.

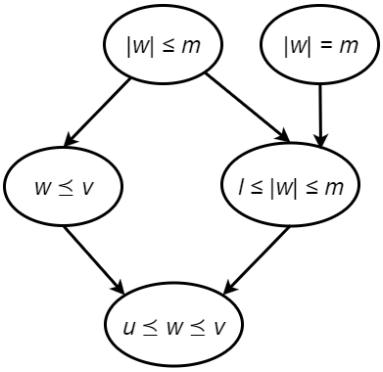

Рис. 1: Сводимости между различными формулировками задачи (взято из работы [11])

Из-за существования такого рода сходимостей все пять задач имеют одинаковую сложность для рассмотренных в работе [11] классов грамматик, среди которых имеются классы контекстно-свободных и конъюнктивных

грамматик. В таблице 1 приведена сложность задач принадлежности, пустоты и генерации строк с ограничениями для классов CF — контекстно-свободных и $Conj$ — конъюнктивных грамматик.

Таблица 1: Сложность задач для грамматик, подразумевается полнота для указанных классов (фрагмент таблицы из работы [11])

	Принадлежность	**Пустота**	**Рассматриваемая задача**
CF	P	P	P
Conj	P	неразрешима	NP

4 Связь задачи генерации строк с синтаксическим анализом графов для КС-языков

Рассмотрим задачу синтаксического анализа графа $G = (Q, \Sigma, \delta)$ с использованием контекстно-свободной грамматики $C = (N, \Sigma, P)$ и различных семантик запроса.

***Relational* семантика запроса.** Зафиксируем нетерминал $a \in N$ и вершины графа $m, n \in Q$. При использовании данной семантики запроса необходимо проверить пустоту КС-языка $L(C, a) \cap L(G, m, n)$. Чтобы использовать для этой задачи генератор строк, необходимо построить КС-грамматику, порождающую язык $L(C, a) \cap L(G, m, n)$. В качестве такой грамматики используется аннотированная грамматика $C_G = (N_G, \Sigma, P_G)$. А построив данную грамматику мы автоматически решаем поставленную задачу. Действительно, если $a[m, n] \notin N_G$, то $L(C, a) \cap L(G, m, n) = \emptyset$, а если $a[m, n] \in N_G$, то $L(C, a) \cap L(G, m, n) \neq \emptyset$. Поэтому нет необходимости в применении генератора строк для поиска ответа на запрос с *relational* семантикой.

***All-path* семантика запроса.** Зафиксируем нетерминал $a \in N$ и вершины графа $m, n \in Q$. В данной вариации задачи синтаксического анализа графов необходимо предъявить все такие пути из вершины m в вершину n, что метки на их ребрах образуют строку из языка $L(C, a)$. Совокупность всех строк, порождаемых такими путями, будет являться КС-языком $L(C, a) \cap L(G, m, n)$. Чтобы использовать генератор строк этого языка, необходимо построить КС-грамматику его порождающую. В качестве такой грамматики используется аннотированная грамматика $C_G = (N_G, \Sigma, P_G)$. Причем, в работе [5] эту грамматику и предлагают в качестве ответа на запрос с *all-path* семантикой. Одно из преимуществ данного решения заключается в том, что имеется возможность компактно хранить бесконечное множество путей, которое может возникнуть при наличии цикла в графе G. Поэтому нет необходимости в применении генератора строк для поиска ответа на запрос с *all-path* семантикой. Но генератор строк может быть использован для получения конкретных строк пользователем из КС-грамматики $L(C, a) \cap L(G, m, n)$. Это похоже на использование следующей (*single-path*)

семантики запроса, но в ней выдается один путь (и, соответственно, одна строка), когда здесь можно последовательно выдавать все новые строки из языка $L(C, a) \cap L(G, m, n)$.

Таким образом, в рассматриваемой задаче имеется возможность использовать генератор строк аннотированной КС-грамматики C_G, порождающей язык $L(C, a) \cap L(G, m, n)$. Как уже было замечено, $L(C, a) \cap L(G, m, n) \neq \emptyset \Leftrightarrow a[m, n] \in N_G$. Поэтому после построения аннотированной грамматики можно сразу определить пуст ли язык $L(C, a) \cap L(G, m, n)$. Необходимость применения генератора строк остается только в случае $L(C, a) \cap L(G, m, n) \neq \emptyset$. Поэтому далее считаем, что $L(C, a) \cap L(G, m, n) \neq \emptyset$, и будем исследовать возможность применения генератора строк КС-языка $L(C, a) \cap L(G, m, n)$ для последовательного предоставления отдельных строк пользователю.

Будем генерировать строку, соответствующую одному из пяти видов ограничений, представленных на рис. 1. Во всех случаях имеется ограничение сверху (лексикографическое или по длине строки), что позволяет свести задачу генерации строк заданного языка к конечному числу проверок на принадлежность строк этому языку. А так как рассматриваемый лексикографический порядок в работе [11] также упорядочивает строки по длине, то во всех пяти видах ограничений мы ограничиваем генерируемую строку длиной сверху. Поэтому чтобы гарантировать генерацию строки, необходимо знать строку минимальной длины из рассматриваемой грамматики (или лексикографически наименьшую строку). Для КС-грамматик существует алгоритм, позволяющий находить строку минимальной длины, что и используется в работе [11] для генерации строк КС-языка и в работе [5] в задаче синтаксического анализа графов с использованием *single-path* семантики запроса. Поэтому при последовательной генерации строк мы сначала решим задачу для *single-path* семантики, как и в работе [5], а после, меняя ограничения, сможем генерировать новые строки. Таким образом, каждый раз будет решаться задача для *single-path* семантики, но предъявляться будет не строка наименьшей длины, а строка наименьшей длины из еще не рассмотренных строк.

***Single-path* семантика запроса.** Зафиксируем нетерминал $a \in N$ и вершины графа $m, n \in Q$. При использовании данной семантики запроса необходимо проверить пустоту КС-языка $L(C, a) \cap L(G, m, n)$ и, если он не пуст, то предъявить путь из вершины m в вершину n, такой что метки на ребрах этого пути образуют строку из языка $L(C, a)$. В качестве такого пути в работе [5] было решено использовать путь, соответствующий строке наименьшей длинны языка $L(C, a) \cap L(G, m, n)$. Как и в случае *all-path* семантики запроса сначала строится аннотированная грамматика $C_G = (N_G, \Sigma, P_G)$, и если она порождает не пустой язык, то в ней ищется строка минимальной длины, которая и будет соответствовать искомому пути в графе G.

Так как построение аннотированной грамматики и строки минимальной длины КС-языка $L(C, a) \cap L(G, m, n)$ необходимо для использования рассматриваемой генерации строк этого языка из работы [11], то алгоритм решения задачи синтаксического анализа графов с использованием *single-*

path семантики запроса, предложенный в работе [5], и является примером использования генерации строки из КС-языка $L(C, a) \cap L(G, m, n)$.

5 Связь задачи генерации строк с синтаксическим анализом графов для конъюнктивных языков

Рассмотрим задачу синтаксического анализа графа $G = (Q, \Sigma, \delta)$ с использованием конъюнктивной грамматики C. Аналогично случаю КС-грамматик будем обозначать $L(C, a)$ — язык, порождаемый конъюнктивной грамматикой C со стартовым нетерминалом a. Гарантировать завершаемость алгоритма генерации строк из бесконечного языка можно либо умея решать задачу определения пустоты этого языка, либо заранее зная его непустоту. Первая проблема, с которой мы сталкиваемся, рассматривая конъюнктивные грамматики вместо КС-грамматик — это неразрешимость задачи определения пустоты языка. Поэтому далее будем предполагать, что пересечение рассматриваемых языков $L(C, a) \cap L(G, m, n) \neq \emptyset$ и если это предположение неверно, то мы не можем гарантировать завершаемость алгоритмов, использующих генератор строк данного языка. Отсюда следует неразрешимость задачи синтаксического анализа с использованием конъюнктивного языка и *relational* семантики запроса, о чем также упоминается в работе [4].

Мы легко можем задать конъюнктивную грамматику, порождающую пересечение языков $L(C, a) \cap L(G, m, n)$, так как в данном формализме присутствует явная операция пересечения. Так как при всех видах ограничений, рассматриваемых в работе [11], применяется ограничение сверху длины генерируемой строки, то, не имея оценки сверху на $minlen$ — минимальную длину строки непустого языка $L(C, a) \cap L(G, m, n)$, нельзя гарантировать нахождения хотя бы одной строки. То есть для любого ограничения на генерируемую строку, накладывающего, в частности, ограничение сверху на длину строки ($|w| \leq m$), может оказаться, что $minlen > m$ и генератор строк не сможет сгенерировать ни одной строки. Но никакой оценки сверху на $minlen$ найти не удастся, так как в противном случае за конечное число проверок принадлежности строки к конъюнктивному языку (эта задача разрешима и P-полна) можно было бы решить задачу проверки пустоты этого языка (эта задача является неразрешимой). Таким образом, для любых ограничений заранее неизвестно, есть ли хотя бы одна строка, удовлетворяющая этим ограничениям. Но если известно для какого-то ограничения, что такая строка существует, то задача генерации строк для конъюнктивных грамматик и этих ограничений, согласно работе [11], NP-полна.

Предположим, что найдется хотя бы одна строка, удовлетворяющая рассматриваемым ограничениям. Тогда при использовании *all-path* семантики запроса, применяя алгоритм генерации строки, происходил бы просто перебор всех возможных строк и проверка на принадлежность этих строк к

языку $L(C, a) \cap L(G, m, n)$, что не соответствует практическому смыслу задачи. Для задачи синтаксического анализа графов с использованием *single-path* семантики запроса есть возможность сгенерировать некоторую строку непустого языка $L(C, a) \cap L(G, m, n)$ (в работе [5] подчеркнута логичность выбора строки минимальной длины, но это необязательно).

6 Заключение

В данной работе была показана связь между задачей генерации строк, рассмотренной в работе [11], и некоторыми вариациями задачи синтаксического анализа графов, предложенных в работах [4, 5] и использующих контекстно-свободные и конъюнктивные грамматики.

Стоит отметить, что использование конъюнктивных языков в задачах синтаксического анализа графов мало изучено. Полученные результаты могут быть использованы в дальнейших исследованиях данной области. Одной из тем таких исследований, например, является применимость булевых [8, 9] грамматик в синтаксическом анализе графов.

Список литературы

[1] Anderson J., Novák Á., Sükösd Z. Quantifying variances in comparative RNA secondary structure prediction // BMC Bioinformatics. — 2013. — P. 14–149.

[2] Barrett C., Jacob R., Marathe M. Formal-language-constrained path problems // SIAM Journal on Computing. — 2000. — Vol. 30, no. 3. — P. 809–837.

[3] Context-free path queries on RDF graphs / X. Zhang, Z. Feng, X. Wang et al. // International Semantic Web Conference / Springer. — 2016. — P. 632–648.

[4] Hellings J. Conjunctive context-free path queries. — 2014.

[5] Hellings J. Querying for Paths in Graphs using Context-Free Path Queries // arXiv preprint arXiv:1502.02242. — 2015.

[6] Mendelzon A., Wood P. Finding Regular Simple Paths in Graph Databases // SIAM J. Computing. — 1995. — Vol. 24, no. 6. — P. 1235–1258.

[7] Okhotin A. Conjunctive grammars // Journal of Automata, Languages and Combinatorics. — 2001. — Vol. 6, no. 4. — P. 519–535.

[8] Okhotin A. Boolean grammars // Information and Computation. — 2004. — Vol. 194, no. 1. — P. 19–48.

[9] Okhotin A. Conjunctive and Boolean grammars: the true general case of the context-free grammars // Computer Science Review. — 2013. — Vol. 9. — P. 27–59.

[10] Ullman J., Hopcroft J., Motwani R. Introduction to Automata Theory, Languages, and Computation, 3th edition // Pearson. — 2007.

[11] Охотин А. О сложности задачи генерации строк // Дискретная математика. — 2003. — Vol. 15, no. 4. — P. 84–99.