Приложения теории формальных языков и синтаксического анализа

Семён Григорьев

30 мая 2019 г.

Содержание

1 Введение

Поиск путей в графе.

Поиск путей с ограничениями.

Один из способов задавать ограничения на пути в графе основан на использовании языков. Базоваое определение языка говорит нам, что язык — это множество слов над некоторым алфавитом. Если рассмотреть граф, рёбра которого помечены символами из алфавита, то путь в графе будет задавать слово: достаточно соединить последовательно символы, лежащие на рёбрах пути. Множество же таких путей будет задавать множество слов или язык. Таким образом, если мы хотим найти некоторое множество путей в графе, то в качестве ограничения можно описать язык, который должно задачать это множество. Иными словами, завача поиска путей может быть сформулирована следующим образом: необходимо найти такие пути в графе, что слова, получаемые конкатенацей меток их рёбер, принадлежат заданному языку.

Рассмотрим различные варианты постановки задачи.

Различные алгоритмы решения.

2 Постановка задачи

Пусть $\mathcal{G} = \langle V, E, L \rangle$ — конечный ориентрованный граф с метками на рёбрах, где V — конечное множество вершин, E — конечное множество рёбер, L — конечное множество или алфавит меток. Рёбра будем представлять в виде упорядоченных троек из $V \times L \times V$. Путём π в графе $\mathcal{G}$ будем называть последовательность рёбер такую, что для любых двух последовательных рёбер $e_1 = (u_1, l_1, v_1)$ и $e_2 = (u_2, l_2, v_2)$ в этой последовательности, конечная вершина первого ребра является начальной вершиной второго, то есть $v_1 = u_2$. Будем обобзначать путь из вершины v_0 в вершину v_n как $v_0 \pi v_n = e_0, e_1, \ldots, e_{n-1} = (v_0, l_0, v_1), (v_1, l_1, v_2), \ldots, (v_{n-1}, l_n, v_n)$.

Функция $\omega(\pi) = \omega((v_0, l_0, v_1), (v_1, l_1, v_2), \ldots, (v_{n-1}, l_n, v_n)) = l_0 \cdot l_1 \cdot \ldots \cdot l_n$ строит слово по пути посредством конкатенации меток рёбер вдоль этого пути. Очевидно, для пустого пути данная функция будет возвращать пустое слово, а для пути длины $n > 0$ — непустое слово длины n.

Путь $G = \langle \Sigma, N, P \rangle$ — контекстно-свободная граммтика. Будем считать, что $L \subseteq \Sigma$. Мы не фиксируем стартовый нетерминал в определении граммтики, поэтому, чтобы описать язык, задаваемый ей, нам необходимо отдельно зафиксировать стартовый нетерминал. Таким образом, будем говорить, что $L(G, N_i) = \{w | N_i \overset{*}{\underset{G}{\Rightarrow}} w\}$ — это язык задаваемый граммтикой G со стартовым нетерминалом N_i.

Задача достижимости:

Задача поиска путей:

3 Алгоритм на матричных опреациях

3.1 Транзитивне замыкание графа

3.2 Нормальная форма Хомского

Данный алгоритм накладывает ограничение на форму грамматики: грамматика должна быть в “ослабленной” нормальной форме Хомского.

3.3 Алгоритм CYK

3.4 Алгоритм

3.5 Алгоритм

3.6 Конъюнктивные и будевы граммтики

3.7 Особенности реализации матричного алгоритма

4 Обзор

4.1 Вопросы и задачи

1. !!!

5 Сжатое представление леса разбора

5.1 Дерево вывода

Дерево вывода цепочки в граммтике

5.2 Неоднозначные граммтики

5.3 Лес разбора как представление контекстно-свободной грамматики

5.4 Вопросы и задачи

1. Постройте дерево вывода цепочки $w = aababb$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b\ S\}, S\rangle$.
2. Постройте все левосторонние выводы цепочки $w = ababab$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b \mid S\ S\}, S\rangle$.
3. Постройте все правосторонние выводы цепочки $w = ababab$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b \mid S\ S\}, S\rangle$.

4. Постройте все деревья вывода цепочки $w = ababab$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b \mid S\ S\}, S\rangle$, соответствующие левосторонним выводам.

5. Постройте все деревья вывода цепочки $w = ababab$ в грамматике $G = \langle\{a, b\}, \{S\}, \{S \to \varepsilon \mid a\ S\ b \mid S\ S\}, S\rangle$, соответствующе правосторонним выводам.

6. Как связаны между собой леса, полученные в предыдущих двух задачах? Какие выводы можно сделать из такой связи?

7. !!!

6 Алгоритм на основе восходящего анализа

6.1 Восходящий синтаксический анализ

7 Алгоритм на основе нисходящего анализа

W [1]

7.1 Нисходящий синтаксический анализ

8 От CFPQ к вычислению Datalog-запросов

8.1 Datalog

Конечные Эрбрановы модели. Наименьшая неподвижная точка. $C :- d$

8.2 КС-запрос как запрос на Datalog

Покажем, что для данного графа и КС-запроса можно построить эквивалентный запрос на Datalog.

Пусть дан граф Граф преобразуется в набор фактов (базу данных).

Пусть есть граммтика G: $S \to a\ b \mid a\ S\ b$. Она может быть преобразована в запрос следующего вида. $s(X, Y) :- a(X, Z), b(Z, Y)$. $s(X, Y) :- a(X, Z), s(Z, W)b(W, Y)$. $? :- s(X, Y)$

Наблюдения: появились пременные, есть порядок у конъюнктов, который задаёт порядок связывания.

8.3 Обобщение GLL для вычисления Datalog-запросов

Дескриптор — состояние процесса: состояние автомата, результат проделанной работы, подстановка. Задача — найти подстановки. На каждом шаге есть набор подстановок.

Список литературы

[1] E. Verbitskaia, I. Kirillov, I. Nozkin, and S. Grigorev. Parser combinators for context-free path querying. In *Proceedings of the 9th ACM SIGPLAN International Symposium on Scala*, Scala 2018, pages 13–23, New York, NY, USA, 2018. ACM.