**Синтаксический анализ графов с использованием конъюнктивных грамматик**

Азимов Р. Ш.,

[rustam.azimov19021995@gmail.com,](mailto:rustam.azimov19021995@gmail.com,)

Санкт-Петербургский государственный университет, Лаборатория языковых инструментов JetBrains

11 февраля 2018 г.

**Аннотация**

Графы используются в качестве структуры данных для представления больших объемов информации в компактной и удобной для анализа форме в таких областях, как биоинформатика, графовые базы данных, статический анализ кода. В этих областях часто необходимо вычислять некоторые запросы к большим графам с целью выявления сложных зависимостей между вершинами. Ответом на такие запросы обычно является множество всех троек *(A, m, n)*, для которых существует путь в графе от вершины *m* до вершины *n* такой, что метки на ребрах этого пути образуют строку, выводимою из нетерминала *A* некоторой контекстно-свободной грамматики. Говорят, что такой тип запросов вычислен с использованием *реляционной семантики запросов*. Кроме того, существуют *конъюнктивные грамматики*, образующие более широкий класс грамматик, чем традиционные контекстно-свободные. Использование конъюнктивных грамматик в задаче синтаксического анализа графов позволяет формулировать более сложные запросы к графу и решать более широкий круг задач. Известно, что задача вычисления запросов к графу с использованием реляционной семантики и конъюнктивных грамматик — неразрешима. В данной работе будет предложен алгоритм, вычисляющий приближенное решение данной задачи, а именно, аппроксимацию сверху для множества троек (*A, m, n*). Предложенный алгоритм основан на матричных операциях, что позволяет повысить производительность, используя вычисления на графическом процессоре.

**Ключевые слова: синтаксический анализ графов, конъюнктивные грамматики, транзитивное замыкание, матричные операции, вычисления на GPU**

**Введение**

Графы используются в качестве структуры данных во многих областях, например, в биоинформатике [2], в графовых базах данных [9], в статическом анализе программ [15]. В этих областях часто оказывается необходимым вычислять запросы к большим графам с целью выявления сложных зависимостей между вершинами. Результатом вычисления таких запросов является множество неявных отношений между вершинами графа, то есть пути в графе [7]. Естественно помечать ребра графа символами из некоторого конечного алфавита и выделять необходимые пути с помощью формальных грамматик над тем же алфавитом (регулярные выражения, контекстно-свободные грамматики). Наиболее популярными являются запросы, которые используют контекстно-свободные (КС) грамматики, поскольку КС-языки обладают значительной выразительной силой. Также существуют *конъюнктивные грамматики* [11], задающие более широкий класс языков, чем контекстно-свободные. Использование конъюнктивных грамматик в задаче синтаксического анализа графов позволяет формулировать более сложные запросы к графам и решать более широкий круг задач, например, задачи поиска псевдонимов и уязвимостей в исходном коде [15]. При этом необходимо отметить, что задача вычисления запросов к графу с использованием реляционной семантики и конъюнктивных грамматик является неразрешимой [7]. Один из распространенных способов найти приближенное решение неразрешимой задачи — построить аппроксимацию решения.

В данной работе предложен алгоритм, вычисляющий приближенное решение задачи синтаксического анализа графов с использованием реляционной семантики запросов и конъюнктивных грамматик. Предложенный алгоритм основан на матричных операциях, что позволяет повысить производительность, используя для вычислений графический процессор.

**Обзор**

В этом разделе мы определим задачу синтаксического анализа графов и обсудим основные подходы, применяемые для ее решения.

Пусть Σ — конечное множество терминальных символов. *Помеченным графом* будем называть пару *D* = (*V, E*), где *V* является множеством вершин, а — множеством ребер с метками из алфавита Σ. Для пути *π* в графе *D* мы будем использовать обозначение *l*(π), чтобы указать на слово, полученное конкатенацией меток на ребрах данного пути. Кроме того, запись *mπn* будет обозначать, что в рассматриваемом графе существует путь из вершины в вершину .

∈

Результатом работы алгоритма синтаксического анализа графов с использованием формальной грамматики *G* обычно является множество всех троек (*A, m, n*), для которых путь *mπn* таков, что строка *l*(*π*) выводима из нетерминала *A* в грамматике *G*. Говорят, что такой тип запросов вычислен с использованием *реляционной семантики запросов* [7].

Традиционно в качестве грамматики *G* используются регулярные выражения [9, 17, 18, 19, 20]. Но в последнее время стало популярным использовать КС-грамматики [3, 7, 12, 16], так как некоторые полезные запросы не могут быть описаны с помощью регулярных грамматик. Примером таких запросов являются классические запросы поиска всех вершин в графе, находящихся на одном уровне иерархии [1]. Все рассмотренные алгоритмы синтаксического анализа графов принимают на вход КС-грамматики в *нормальной форме Хомского* [5].

Существует ряд алгоритмов синтаксического анализа графов с использованием реляционной семантики запросов и КС-грамматик [7, 12, 16], которые основаны на методе динамического программирования. Данные алгоритмы обобщают такие алгоритмы синтаксического анализа, как CYK [8, 14] и Earley [6]. В работе [7] для заданного графа *D* = (*V, E*) и КС-грамматики *G* = (*N,* Σ*, P*) для каждого определяются *контекстно-свободные отношения* следующим образом:

,

где— язык, порожденный грамматикой *G* со стартовым нетерминалом *A*.

В [7] представлен алгоритм, вычисляющий контекстно-свободные отношения . Кроме того, существует алгоритм синтаксического анализа графов с использование реляционной семантики запросов и КС-грамматик, вычисляющий данные контекстно-свободные отношения используя матричное транзитивное замыкание [3]. Данный алгоритм обобщает алгоритм Вэлианта [13] и сводится к ряду умножений булевых матриц.

Также существуют *конъюнктивные грамматики* [11], образующие более широкий класс грамматик, чем контекстно-свободные. Как и в случае КС-грамматик мы рассматриваем только конъюнктивные грамматики в бинарной нормальной форме [10]. Мы не выделяем стартовый нетерминал, так как его можно будет определить во время синтаксического анализа графа. Так как для каждой конъюнктивной грамматики можно построить эквивалентную ей грамматику в бинарной нормальной форме, то достаточно рассмотреть только грамматики следующего вида.

*Конъюнктивная грамматика* — это тройка *G* = (*N,* Σ*, P*), где *N* — конечное множество нетерминальных символов, Σ — конечное множество терминальных символов и *P* — конечное множество правил следующего типа:

* , для
* , для *,* .

Мы будем писать, чтобы указать, что строка может быть получена из нетерминала *A* некоторой последовательностью применений правил конъюнктивной грамматики, где отношение определено следующим образом:

1. При применении правила любой подтерм *A* любого терма может быть перезаписан подтермом :
2. Конъюнкция нескольких одинаковых строк из может быть перезаписана одной такой строкой: для любого ,

*Языком*, сгенерированным конъюнктивной грамматикой *G* = (*N,* Σ*, P*)со стартовым нетерминалом , будем называть .

Определение контекстно-свободных отношений естественным образом обобщается до определения конъюнктивных отношений заменой КС-грамматики *G* на конъюнктивную. Таким образом, задача синтаксического анализа графов с использованием реляционной семантики запросов и конъюнктивных грамматик сводится к поиску конъюнктивных отношений для всех нетерминалов *A*. Но также известен факт, что данная задача является неразрешимой [7]. Поэтому единственный известный алгоритм для решения данной задачи, описанный в [15], дает приближенный результат. Данный алгоритм используется для статического анализа программ и аппроксимирует сверху конъюнктивные отношения , то есть точное решение является подмножеством предложенного решения.

**Существующие работы**

Существует ряд алгоритмов синтаксического анализа графов с использованием реляционной семантики запросов и КС-грамматик [7, 12, 16], которые, однако, демонстрируют низкую производительность на больших графах, так как имеют в худшем случае кубическую временную сложность. Одной из самых популярных техник, используемых для увеличения производительности при работе с большими объемами данных, является использование графического процессора. В работе [3] представлен алгоритм синтаксического анализа графов, использующий реляционную семантику запросов и КС-грамматики и вычисляющий матричное транзитивное замыкание с применением матричных операций, позволяя эффективно использовать вычисления на графическом процессоре [4]. Но все перечисленные алгоритмы работают только с КС-грамматиками и не позволяют обрабатывать запросы, описанные с помощью конъюнктивных грамматик.

В работе [15] описан алгоритм, работающий с конъюнктивными грамматиками. Но данный алгоритм принимает на вход только определенный подкласс конъюнктивных грамматик, а именно линейные конъюнктивные грамматики, которые имеют в правилах не более одного нетерминального символа в каждом конъюнкте.

Таким образом, построение алгоритма синтаксического анализа графов, использующего реляционную семантику запросов и работающего с произвольными конъюнктивными грамматиками — является открытой проблемой.

**Алгоритм**

В данном разделе мы представляем алгоритм синтаксического анализа графов, использующий реляционную семантику запросов и конъюнктивные грамматики. Предложенный алгоритм находит аппроксимацию сверху для конъюнктивных отношений , вычисляя некоторое матричное транзитивное замыкание.

Пусть даны помеченный граф *D* = (*V, E*) и конъюнктивная грамматика *G* = (*N,* Σ*, P*).

Определим *конъюнктивное матричное умножение* , где *a* и *b* — матрицы подходящего размера, элементами которых являются подмножества множества нетерминалов *N*, следующим образом:

,

где матрица *d* является результатом поэлементного декартово произведения матриц *a* и *b*, т.е.

.

Кроме того, определим *конъюнктивное транзитивное замыкание* квадратной матрицы *a*, как

,

где для и .

Чтобы построить аппроксимацию сверху для конъюнктивных отношений , мы построим матрицу *T*, используя ребра графа *D* и грамматику *G*. Для начала пронумеруем вершины графа *D* от 0 до (|V| - 1) и будем ассоциировать вершины с их номерами. Проинициализируем каждый элемент |V||V| матрицы *T* пустым множеством . Далее, для каждых *i* и *j* мы устанавливаем

Наконец, мы вычисляем конъюнктивное транзитивное замыкание

,

где для , а — это построенная нами матрица *T*.

**Лемма 1.** Пусть даны помеченный граф *D* = (*V, E*) и конъюнктивная грамматика *G* = (*N,* Σ*, P*) в нормальной форме. Тогда для любых *i, j* и для любого нетерминала , если и существует путь *iπj* такой, что для строки *l*(*π*) существует дерево вывода грамматики *G* = (*N,* Σ*, P*) со стартовым нетерминалом *A* с высотой , то .

**Доказательство**(методом математической индукции):

*База индукции:*

Покажем, что утверждение леммы верно при *k = 1.* Для любых *i, j* и для любого нетерминала , если и существует путь *iπj* такой, что для строки *l*(*π*) существует дерево вывода грамматики *G* = (*N,* Σ*, P*) со стартовым нетерминалом *A* с высотой , то путь *π* состоит из единственного ребра из вершины *i* в вершину *j* и , где *x = l*(*π*) . Таким образом, и утверждение леммы доказано при *k = 1*.

*Индукционный переход:*

Предположим, что утверждение леммы верно для всех , где . Покажем, что утверждение леммы также верно и при *k = p*.

Пусть существует путь *iπj*, образующий строку *l*(*π*), для которой существует дерево вывода грамматики *G* = (*N,* Σ*, P*) со стартовым нетерминалом *A* с высотой . Если , то утверждение леммы верно по индукционному предположению. Пусть , а правило, использованное для корня этого дерева — . Тогда все поддеревья для нетерминалов имеют высоту меньшую, чем . Пусть каждая пара нетерминалов в конъюнктах разбивает путь *iπj* на два подпути в вершинах . По индукционному предположению получаем, что , а , для всех . Таким образом, все пары нетерминалов принадлежат , где . Значит, по определению конъюнктивного матричного умножения, получаем, что , где . Таким образом, утверждение леммы также верно и при *k = p*, ч. т. д.

**Теорема 1.** (*Корректность*) Пусть даны помеченный граф *D* = (*V, E*) и конъюнктивная грамматика *G* = (*N,* Σ*, P*) в нормальной форме. Тогда для любых *i, j* и для любого нетерминала , если , то .

**Доказательство:**

Пусть . По определению конъюнктивных отношений , получаем, что существует путь *iπj* такой, что строка *l*(*π*) принадлежит языку, порожденному грамматикой *G* со стартовым нетерминалом *A*. Рассмотрим некоторое дерево вывода этой грамматики для строки *l*(*π*). Пусть высота этого дерева равна *h*. Тогда, по лемме 1, получаем, что . А так как , то , ч. т. д.

**Теорема 2.** (*Конечность*) Пусть даны помеченный граф *D* = (*V, E*) и конъюнктивная грамматика *G* = (*N,* Σ*, P*) в нормальной форме. Тогда конъюнктивное транзитивное замыкание матрицы *T*, построенной по графу *D* и грамматике *G* может быть вычислено за конечное число операций.

**Доказательство:**

Если матрицa для некоторого , то для любого , и конъюнктивное транзитивное замыкание . Таким образом, для нахождения матрицы достаточно вычислять матрицы до тех пор, пока не встретятся две подряд идущие одинаковые матрицы. По определению конъюнктивного транзитивного замыкания, в каждой ячейке матрицы количество нетерминалов больше либо равно, чем в соответствующей ячейке матрицы . А так как общее количество нетерминалов в любой матрице не может быть больше, чем , то . Таким образом, матрица может быть вычислена за конечное число операций, ч. т. д.

Таким образом, мы показали, как аппроксимация сверху всех конъюнктивных отношений может быть найдена с помощью вычисления конъюнктивного транзитивного замыкания .

**Пример работы алгоритма**

**Эксперименты**

**Заключение**

**Список литературы**

1. Abiteboul S., Hull R., Vianu V. Foundations of databases: the logical level. — Addison-Wesley Longman Publishing Co., Inc., 1995.
2. Anderson J. W. и др. Quantifying variances in comparative RNA secondary structure prediction // BMC bioinformatics. — 2013. — Т. 14, № 1. — С. 149.
3. Azimov R., Grigorev S. Context-Free Path Querying by Matrix Multiplication. — 2018.
4. Che S., Beckmann B. M., Reinhardt S. K. Programming GPGPU Graph Applications with Linear Algebra Building Blocks // International Journal of Parallel Programming. — 2016. — С. 1—23.
5. Chomsky N. On certain formal properties of grammars // Information and control. — 1959. — Т. 2, № 2. — С. 137—167.
6. Grune D., Jacobs C. J. H. Parsing Techniques (Monographs in Computer Science). — Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. — ISBN 038720248X.
7. Hellings J. Conjunctive context-free path queries. — 2014.
8. Kasami T. AN EFFICIENT RECOGNITION AND SYNTAXANALYSIS ALGORITHM FOR CONTEXT-FREE LANGUAGES. Тех. отч. / DTIC Document. — 1965.
9. Mendelzon A., Wood P. Finding Regular Simple Paths in Graph Databases // SIAM J. Computing. — 1995. — Т. 24, № 6. — С. 1235—1258.
10. Okhotin A. Conjunctive and Boolean grammars: the true general case of the context-free grammars // Computer Science Review. — 2013. — Т. 9. — С. 27—59.
11. Okhotin A. Conjunctive grammars // Journal of Automata, Languages and Combinatorics. — 2001. — Т. 6, № 4. — С. 519—535.
12. Sevon P., Eronen L. Subgraph queries by context-free grammars // Journal of Integrative Bioinformatics. — 2008. — Т. 5, № 2. — С. 100.
13. Valiant L. G. General context-free recognition in less than cubic time // Journal of computer and system sciences. — 1975. — Т. 10, № 2. — С. 308— 315.
14. Younger D. H. Recognition and parsing of context-free languages in time n3 // Information and control. — 1967. — Т. 10, № 2. — С. 189—208.
15. Zhang Q., Su Z. Context-sensitive data-dependence analysis via linear conjunctive language reachability // Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages. — ACM. 2017. — С. 344—358.
16. Zhang X. и др. Context-free path queries on RDF graphs // International Semantic Web Conference. — Springer. 2016. — С. 632—648.
17. Abiteboul S., Vianu V. 1997. Regular path queries with constraints. In Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. ACM, 122–133.
18. Fan W., Li J., Ma S., Tang N, and Wu Y. 2011. Adding regular expressions to graph reachability and pattern queries. In Data Engineering (ICDE), 2011 IEEE 27th International Conference on. IEEE, 39–50.
19. Nolé M. and Sartiani C. 2016. Regular path queries on massive graphs. In Proceedings of the 28th International Conference on Scientific and Statistical Database Management. ACM, 13.
20. Reutter J., Romero M., and Vardi M. 2017. Regular queries on graph databases. Theory of Computing Systems 61, 1 (2017), 31–83