**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 различных областях – биоинформатике, графовых базах данных, статическом анализе кода и др. При этом оказывается необходимо вычислять запросы к большим графам с целью выявления зависимостей между их 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* в некоторой контекстно-свободной грамматике. 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 является неразрешимой. В данной работе будет предложен алгоритм, вычисляющий приближенное решение этой задачи, а именно, аппроксимацию сверху. 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**

1. **Введение.** Графы используются во многих областях, например, в биоинформатике [2], в графовых базах данных [9], при статическом анализе программ [15]. При этом оказывается необходимым вычислять запросы к большим графам с целью выявления сложных зависимостей между их вершинами. Результатом вычисления таких запросов является множество неявных отношений между вершинами графа, то есть путей [7]. Естественно помечать ребра графа символами из некоторого конечного алфавита и выделять пути с помощью формальных грамматик над тем же алфавитом (регулярные выражения, контекстно-свободные грамматики). Говорят, что такой тип запросов вычислен с использованием *реляционной семантики запросов*. Наиболее популярными являются запросы, которые используют контекстно-свободные (КС) грамматики. 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]. 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 — построение 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.

Работа организована следующим образом: в разделе 2 даны основные определения, связанные с задачей синтаксического анализа графов, и рассмотрены основные подходы данной области; в разделе 3 проанализированы существующие решения данной задачи; в разделе 4 представлен алгоритм синтаксического анализа графов, использующий реляционную семантику запросов и конъюнктивные грамматики, а также доказана его корректность; в разделе 5 работа предложенного алгоритма разобрана на небольшом примере; в разделе 6 представлены результаты проведенных экспериментов; заключение и направления будущих исследований приведены в разделе 7.

1. **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— это 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 умножению булевых матриц.

Также существуют *конъюнктивные грамматики* [11], образующие более широкий класс грамматик, чем контекстно-свободные. *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 вида:

, для

, для *,* .

Как и в случае КС-грамматик, мы рассматриваем только конъюнктивные грамматики в бинарной нормальной форме [10], так как для каждой конъюнктивной грамматики можно построить эквивалентную ей грамматику в данной форме. Кроме того, мы не выделяем стартовый нетерминал, так как его можно будет определить во время синтаксического анализа графа.

Мы будем писать, чтобы указать, что строка может быть получена из нетерминала *A* некоторой последовательностью применений правил конъюнктивной грамматики. Отношение определим следующим образом.

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

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

Определение контекстно-свободных отношений естественным образом обобщается до определения конъюнктивных отношений путемзамены КС-грамматикина 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.

1. **Существующие работы.** Имеется ряд алгоритмов для 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 не более одного нетерминального символа в каждом конъюнкте правила.

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

1. **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 декартового 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. Инициализируем каждый элемент |V||V| матрицы *T* пустым множеством . Далее, для каждых *i* и *j* мы присваиваем в ячейки матрицы следующие значения:

Наконец, мы вычисляем конъюнктивное транзитивное замыкание , где для , а — это построенная нами матрица *T*. Спецификация алгоритма представлена на листинге 1.

*—* Входной помеченный граф

*—* Входная конъюнктивная грамматика

*—* Результат

2. for each
4. while do
5. /\*

Листинг 1.  Алгоритм синтаксического анализа графов

Для представленного алгоритма справедливы следующие утверждения.

**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.**

*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, ведущего из вершины *i* в вершину *j,* и , где *x = l*(*π*) . Таким образом, и утверждение леммы при *k = 1* доказано.

*Индукционный переход.* Предположим, что утверждение леммы верно для всех , где . Покажем, что утверждение леммы также верно 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 , и правило, использованное для корня этого дерева выглядит так:. Тогда все поддеревья для нетерминалов имеют высоту меньше, чем . Пусть каждая пара нетерминалов в конъюнктах разбивает в вершинах путь *iπj* на два подпути. 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 для *k = p*, ч. т. д.

**Теорема 1.** (*Корректность алгоритма*). 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 алгоритма*). 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 для некоторого , , то для любого , и . 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 .

1. **Пример работы алгоритма.** В данном разделе мы продемонстрируем работу предложенного алгоритма на небольшом примере.

Пример основан на грамматике G со стартовым нетерминалом *S* и следующими правилами.

Конъюнкт *AB* порождает язык , а конъюнкт *DC* порождает язык . Таким образом, конъюнктивная грамматика *G* со стартовым нетерминалом *S* порождает язык .

Рассмотрим работу предложенного алгоритма с входной грамматикой *G* и графом *D*, представленным на рис. 1.

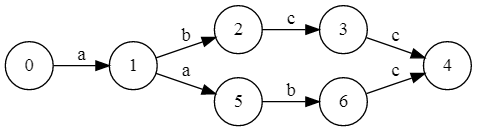


Рис. 1. Пример входного графа

Сначала матрица *T* инициализируется в строках 5-6 листинга 1. Мы будем индексировать состояния матрицы *T*. Обозначим — состояние матрицы *T*, после инициализации. Матрица приведена на рис. 2.

Рис. 2. Матрица

Далее в строках 7-8 листинга 1 вычисляются последующие состояния матрицы *T* до тех пор, пока не найдется такое , что . Таким образом, матрица будет является результатом работы алгоритма.

Для вычисления матрицы необходимо вычислить матрицу *d*, являющуюся декартовым произведением матрицы с собой же. Матрица *d* представлена на рис. 3.

Рис. 3. Матрица *d*

По матрице *d* строится матрица , представленная на рис. 4. Новые нетерминалы матрицы , по сравнению с матрицей , выделены жирным шрифтом.

Рис. 4. Матрица

Далее вычисляются последующие состояния матрицы *T* до включительно, так как . Все последующие состояния матрицы *T* представлены на рис. 5. Новые нетерминалы матрицы , по сравнению с матрицей , выделены жирным шрифтом.

Рис. 5. Последующие состояния матрицы *T*

Таким образом, результатом работы предложенного алгоритма является матрица . Далее мы можем сконструировать множества , которые являются аппроксимацией сверху конъюнктивных отношений для всех нетерминалов , т.е. . Например, так как в ячейке (0,3) результирующей матрицы имеется нетерминал *S*, то . Сконструированные множества для всех нетерминалов представлены на рис. 6.

Рис. 6. Аппроксимация конъюнктивных отношений

Данный пример демонстрирует, что не всегда удается получить точное решение. Например, пара вершин , хотя не существует пути из вершины 0 в вершину 4, образующего строку, выводимую из нетерминала *S* (таковой является единственная строка *abc*). Лишние пары вершин добавляются в том случае, если существуют различные пути из вершины *i* в вершину *j*, которые суммарно соответствуют конъюнктам одного правила, но не существует пути из вершины *i* в вершину *j*, который одновременно соответствовал бы всем конъюнктам этого правила. Например, для конъюнктов правила существует путь из вершины 0 в вершину 4, образующий строку *abcc*, и существует путь из вершины 0 в вершину 4, образующий строку *aabc*. Первый путь соответствует конъюнкту *AB*, так как строка *abcc* принадлежит языку , а второй путь соответствует конъюнкту *DC*, так как строка *aabc* принадлежит языку . Однако, очевидно, не существует пути из вершины 0 в вершину 4, образующего строку *abc*.

1. **Эксперименты.**
2. **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