

Санкт-Петербургский государственный университет

Кафедра Системного программирования

Свитков Сергей Андреевич

Реализация поиска путей с
КС-ограничениями в рамках библиотеки
YC.QuickGraph

Курсовая работа

Научный руководитель:
ст. преп, к. ф-м. н. Григорьев С. В.

Санкт-Петербург
2016

Оглавление

Введение	3
1. Постановка задачи	5
2. Используемая терминология	6
3. Обзор предметной области	7
3.1. Graph Parsing	7
3.2. YaccConstructor	9
3.3. YC.QuickGraph	11
4. Заключение	12
Список литературы	13

Введение

Модель представления данных в виде ориентированных графов с метками на ребрах имеет широкую область применения и используется в биоинформатике, социальных исследованиях (например, при представлении социальных графов), semantic web, при реализации графовых баз данных.

При наличии представления данных в виде определенной структуры становится актуальным вопрос их обработки, а именно — получения из всего набора только тех данных, которые представляют какой-либо конкретный интерес. Для этого используются языки запросов. Существует множество промышленных языков запросов к графам, например Gremlin[11], Cypher[6], и т.д.. Но данные языки являются регулярными, а значит, не могут применяться в некоторых задачах. Например, при разборе генеалогического дерева, встречаются строки вида $parent^n child^n$. Такие строки нельзя распознать с помощью регулярной грамматики, но можно с помощью КС-грамматики с правилами вывода $N \rightarrow parent\ child, N \rightarrow parentN\ child$.

Существуют работы, предлагающие различные подходы к реализации КС-запросов к графам, например [10], [3]. Но большая часть работ по данной теме представляет только теоретические сведения о возможных подходах к реализации, а те, что реализованы на практике, имеют довольно ограниченный функционал или же слишком узкую специализацию. Так, в работе [3] результатом запроса является КС-отношение — тройка вида (n, m, N) , где n и m — вершины, связанные путем, выводимым из нетерминала N . Другой пример формата представления результата запроса встречается в работе [10]. В данном случае это граф, включающий в себя только те ребра, которые встречаются хотя бы в одном пути, который выводим из грамматики, заданной в запросе. Исходя из этого, становится понятно, что класс задач, решаемых с помощью КС-запросов, является весьма обширным. Поэтому хотелось бы иметь библиотеку, средства которой позволят не только писать КС-запросы к графам, но и представлять результат в желаемой форме. Для того,

чтобы такая библиотека была широко используема на практике, она должна быть реализована на одной из популярных платформ. Одной из таких платформ является .NET. Для реализации такой библиотеки требуются инструменты, которые позволяли бы работать с графами, задавать КС-запросы к ним, а также осуществлять синтаксический анализ.

Для .NET существует ряд библиотек для работы с графами, например [5], [4], но наиболее известной является [14], однако, работа над ней прекращена в 2011 году. На кафедре Системного Программирования с 2015 года ведется разработка и поддержка библиотеки `YC.QuickGraph`[12], являющейся обновленной реализацией [14]. В данной библиотеке имеется достаточный набор средств для работы с графами, поэтому используем ее. В качестве алгоритма для синтаксического анализа графов было решено использовать `GLL`[9], поскольку он имеет хорошую асимптотику и может обрабатывать все КС-грамматики, в том числе и лево-рекурсивные, а так же в рамках работы [16] был реализован и интегрирован в `YaccConstructor` [13] — исследовательском проекте лаборатории языковых инструментов `JetBrains`, представляющим собой инструментов для решения различных задач синтаксического и лексического анализа, реализованном для платформы .NET.

Исходя из написанного выше, было принято решение реализовать библиотеку, используя .NET как основную платформу, `YC.QuickGraph` — как средство для работы с графами и `YaccConstructor` в качестве набора инструментов. Результат работы позволит осуществлять КС-запросы к ориентированным графам с помеченными ребрами, представляя результат в виде подграфа, пути, КС-отношения.

1. Постановка задачи

Исходя из сформулированных во введении проблем, была поставлена цель работы:

- Реализовать механизм поиска путей с КС-ограничениями как расширение библиотеки `YC.QuickGraph`

Так же были поставлены задачи для достижения цели:

- Спроектировать архитектуру решения
- Реализовать расширение библиотеки `YC.QuickGraph`
- Опубликовать результат в виде NuGet-пакета

2. Используемая терминология

Перед начало обзора сформулируем несколько определений, которые будут использованы в дальнейшем.

Под ориентированным графом с метками на ребрах будем понимать тройку множеств $G = (V, E, L)$, где V — множество вершин, E — множество ребер, L — множество меток над ребрами.

Под грамматикой будем понимать четверку множеств $Gr = (N, T, P, S)$, где N — множество нетерминальных, T — множество терминальных символов, P — правила вывода, S — стартовый нетерминал.

Под контекстно-свободным запросом к ориентированному графу с метками на ребрах будем понимать поиск таких строк из меток на ребрах, которые выводимы в задаваемой в качестве запроса контекстно-свободной грамматике.

3. Обзор предметной области

Контекстно-свободные (далее — КС) запросы к ориентированным графам с помеченными ребрами применимы для решения широкого класса задач. Существует ряд работ, посвященных этой теме, но их результаты в основном теоретические, или же имеют слишком узкую специализацию. Так же следует отметить отсутствие реализаций для крупных платформ. Далее будут рассмотрены существующие решения в данной области, а так же средства, которые использовались в ходе работы.

3.1. Graph Parsing

Данная область синтаксического анализа имеет прямое отношение к теме работы, поскольку синтаксический анализ графа, как правило, является одним из шагов любого алгоритма, исполняющего запрос к графу. Однако в этой секции будут рассмотрены, преимущественно, работы, посвященные реализации инструментов для исполнения запросов в целом, нежели реализации алгоритмов для синтаксического анализа графов.

Conjunctive Context-Free Path Queries

В данной работе рассматривается построение обобщения существующего регулярного языка запросов к графам с помеченными ребрами CRPQ до языка CCFPQ, средства которого позволяют задавать КС-запросы. Расширение позволяет использовать КС-грамматики вместо регулярных выражений для поиска интересующих путей в графе. Предлагаемый в статье алгоритм использует СΥК для синтаксического анализа графов, разбирая выражения грамматики снизу вверх. Результатом исполнения запроса является КС-отношение — тройка вида (n, m, N) , где n, m — вершины, а N — нетерминал, из которого выводим путь из вершины n в вершину m . К минусам данной работы можно отнести отсутствие практической реализации и возможность представления ре-

зультата запроса лишь в одном формате.

Subgraph Queries by Context-free Grammars

Данная работа рассматривает вопрос о применении КС-запросов в различных задачах биоинформатики. Предложенный в статье подход подразумевает поиск связного подграфа, порождаемого множеством путей, строки из меток на которых выводимы из задаваемой в качестве запроса КС-грамматики. Для синтаксического анализа используется Earley parser [2]. Авторами было проведено тестирование алгоритма, предлагаемого в статье, как на случайно сгенерированных, так и на реальных данных. Эксперименты проводились на компьютере с 1GB оперативной памяти, в качестве ОС использовался Linux. Графы, на которых проводилось тестирование алгоритма, генерировались со следующими ограничениями: фиксированный размер в 10000 вершин, изменяемое ограничение на максимальную длину пути, а так же регулируемая вероятность существования ребра между двумя вершинами. Например, запрос к сгенерированному графу с максимальной длиной пути в 9 вершин выполняется около 200 секунд. Эксперименты, поставленные на реальных данных, показали, что для графа с максимальной длиной пути, равной 8 вершинам, время работы алгоритма может достигать 250 секунд. Таким образом, единственный формат представления результата, а также довольно большое время исполнения запроса, являются основными причинами, по которым результаты данной работы едва ли применимы на практике.

Ослабленный синтаксический анализ динамически формируемых выражений на основе алгоритма GLL

Работа Анастасии Рагозиной, написанная на кафедре СП Математико-Механического факультета СПбГУ, не предлагает решений для написания КС-запросов к графам, но в рамках данной работы был реализован алгоритм для синтаксического анализа динамически формируемого кода на основе алгоритма GLL. Предложенный в работе алгоритм позво-

ляет обрабатывать входные данные большого размера и может быть использован, например, при поиске подпоследовательностей в метагеномных сборках. Так же была доказана корректность и завершаемость алгоритма. Следует отметить, что результатом работы алгоритма является лес разбора, представляемый в виде SPPF[8]. Поскольку имея SPPF, можно представить результат запроса в нескольких формах, просто используя преобразования, а так же учитывая наличие реализации алгоритма для платформы .NET в проекте YaccConstructor, было принято решение использовать результаты работы автора при реализации предлагаемой библиотеки.

3.2. YaccConstructor

YaccConstructor — исследовательский проект лаборатории языковых инструментов JetBrains, применимый для исследования и решения различных задач синтаксического и лексического анализа. Поддержка и разработка проекта осуществляется как работниками лаборатории, так и студентами. Проект имеет одноименный инструмент с открытыми исходниками, который включает в себя большое количество компонентов, таких, как язык спецификаций грамматик YARD, алгоритмы для преобразования грамматик, алгоритмы для синтаксического анализа графов и др. Большая часть компонент проекта YaccConstructor реализованы для платформы .NET на языке F#. Поскольку проект имеет модульную архитектуру 1, его компоненты могут быть использованы независимо.

Более подробно рассмотрим такие средства YaccConstructor, как YARD, GLLParser и GLLGenerator. YARD — язык спецификаций грамматик, позволяющий задавать различные типы грамматик (атрибутивные, в нормальной форме Бэкуса-Наура, КС и др.). Так как в рамках данной работы грамматика является запросом, то для его задания будем использовать YARD. GLL — алгоритм синтаксического анализа, поддерживающий все типы КС-грамматик (в том числе и леворекурсивные), кроме того, имеющий асимптотику $O(n)$ для однознач-

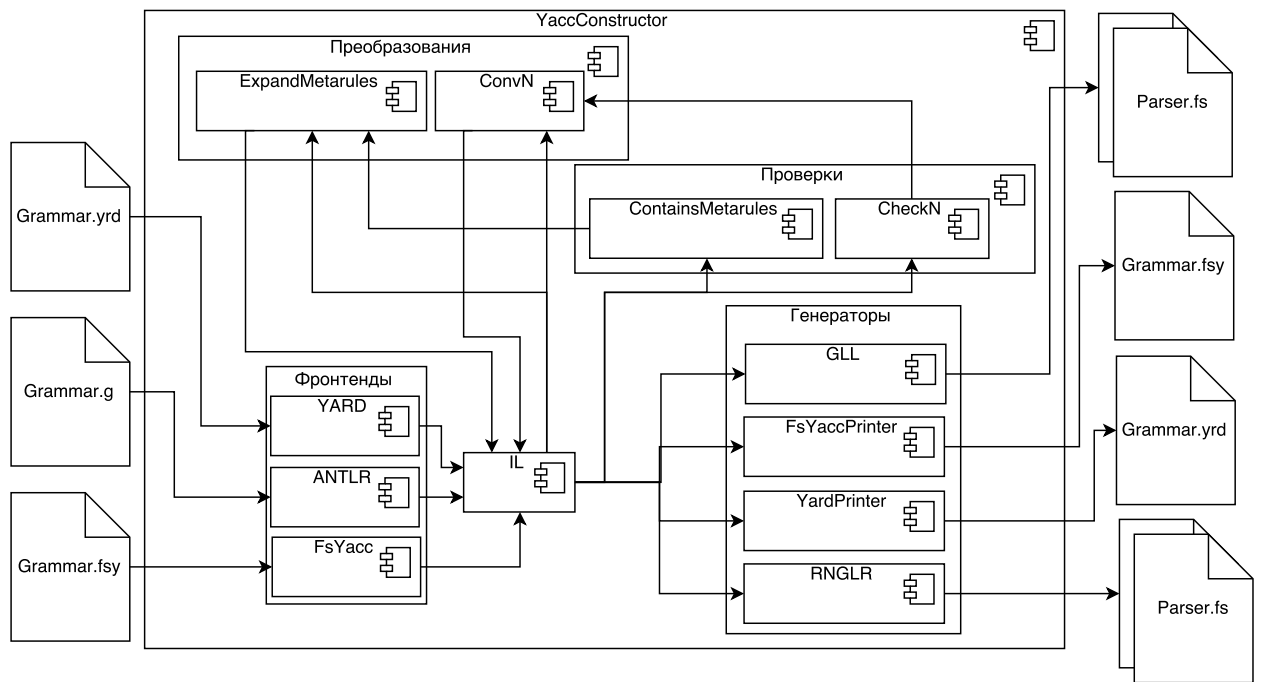


Рис. 1: Архитектура проекта YaccConstructor, заимствована из работы [15]

ных грамматики и $O(n^3)$ в худшем случае. В данной работе GLL будет использоваться для синтаксического анализа графов. YaccConstructor имеет 2 модуля, использующих GLL — GLLGenerator и GLLParser. Первый отвечает за генерацию парсеров для заданных грамматик, а второй, соответственно, за синтаксический анализ. Результатом работы GLLParser является лес разбора — SPPF, из которого с помощью различных функций, которые планируется реализовать в рамках данной работы, можно получить результат разбора в нужном формате: в виде КС-отношения, подграфа, множества путей или кратчайшего пути.

Подводя итоги, можно сделать вывод о том, что YaccConstructor предоставляет все средства, необходимые для реализации библиотеки, а так же позволяет использовать платформу .NET, поэтому подходит для исполнения поставленной в рамках данной работы задачи.

3.3. YC.QuickGraph

YC.QuickGraph — проект лаборатории языковых инструментов JetBrains, представляющий собой библиотеку для работы с графами на платформе .NET. YC.QuickGraph не является разработанным с нуля проектом, за его основу взята библиотека QuickGraph [14], работа над которой была прекращена в 2011 году. YC.QuickGraph имеет средства для задания графов, различные алгоритмы для них (DFS, BFS, поиск кратчайшего пути и др.). Инструменты, реализованные в библиотеке, планируется использовать при реализации данной работы для представления графа и построения нужного формата вывода. Поскольку для реализации библиотеки используется .NET, YC.QuickGraph является наиболее подходящим средством для достижения запланированного результата.

4. Заключение

Текущие результаты

- Изучена предметная область
- Проведен обзор статей, связанных с темой работы
- В соответствии с [7] [1] разработана архитектура предлагаемого решения
- Написана часть текста

Дальнейшая работа

- Реализовать базовое решение для фиксированных грамматики и графа
 - Переработать GLL Parser с целью избежания токенизации
 - Переработать ParserInputGraph с целью избежания дубликации графов
- Протестировать полученное решение
- Оформить результат в виде NuGet-пакета

Список литературы

- [1] Fsharp.org. F# Component Design Guidelines // Fsharp.org. — URL: <http://fsharp.org/specs/component-design-guidelines/fsharp-design-guidelines-v14.pdf> (online; accessed: 18.12.2016).
- [2] Hale John. A probabilistic Earley parser as a psycholinguistic model // Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies / Association for Computational Linguistics. — 2001. — P. 1–8.
- [3] Hellings Jelle. Conjunctive context-free path queries. — 2014.
- [4] Microsoft. AGL // Microsoft Research Site. — URL: <http://rise4fun.com/Ag1> (online; accessed: 06.12.2016).
- [5] NDepend. Graph Sharp // CodePlex. — URL: <http://graphsharp.codeplex.com/> (online; accessed: 06.12.2016).
- [6] Neo4j. Cypher // Neo4j official page. — URL: <https://neo4j.com/developer/cypher/> (online; accessed: 22.11.2016).
- [7] Petricek Tomas. Library patterns // tomasp.net. — URL: <http://tomasp.net/blog/2015/library-frameworks/> (online; accessed: 18.12.2016).
- [8] Rekers Joan Gerard. Parser generation for interactive environments : Ph.D. thesis / Joan Gerard Rekers ; Citeseer. — 1992.
- [9] Scott Elizabeth, Johnstone Adrian. GLL parsing // Electronic Notes in Theoretical Computer Science. — 2010. — Vol. 253, no. 7. — P. 177–189.
- [10] Sevon Petteri, Eronen Lauri. Subgraph queries by context-free grammars // Journal of Integrative Bioinformatics. — 2008. — Vol. 5, no. 2. — P. 100.

- [11] Titan. Gremlin // Titan official page. — URL: <https://github.com/tinkerpops/gremlin/wiki> (online; accessed: 29.11.2016).
- [12] YaccConstructor. YC.QuickGraph // YaccConstructor official page. — URL: <http://yaccconstructor.github.io/QuickGraph/> (online; accessed: 22.11.2016).
- [13] YaccConstructor. YaccConstructor // YaccConstructor official page. — URL: <http://yaccconstructor.github.io> (online; accessed: 29.11.2016).
- [14] de Halleux Jonathan Peli. QuickGraph // CodePlex. — URL: <http://quickgraph.codeplex.com/> (online; accessed: 06.12.2016).
- [15] Григорьев Семён Вячеславович. Синтаксический анализ динамически формируемых программ.
- [16] Рагозина Анастасия Константиновна, Шкредов СД. Ослабленный синтаксический анализ динамически формируемых программ на основе алгоритма GLL. — 2016.