Библиотека парсер-комбинаторов для синтаксического анализа графов

Смолина Софья и Вербицкая Екатерина

18 января 2017 г.

Ключевые слова: синтаксический анализ графов, парсер-комбинатор, графовые базы данных, Meerkat.

Графы и графовые базы имеют широкое применение во многих областях, таких как биоинформатика, логистика, социальные сети и многие другие. Одной из проблем в данной сфере является задача поиска путей в графе, удовлетворяющих некоторым ограничения. Ограничения часто формулируются некоторой контекстно-свободной грамматикой. В таком случае задача сводиться к поиску путей в графе, которые бы соответствовали строкам в контекстно-свободном языке.

Существуют различные подходы к синтаксическому анализу графов (например, [3], [2], [5]). Однако такие подходы не удобны при работе с графовыми базами данных, поскольку усложняется формирование запроса внутри целевой программы. Этот недостаток можно исправить при помощи техники парсер-комбинаторов. К сожалению, большинство существующих библиотек парсер-комбинаторов анализируют только линейный вход (строки), поэтому их непосредственное использование для решения данной задачи невозможно. В рамках данной работы была поставлена задача разработки библиотеки парсер-комбинаторов для работы с графами. За основу решения была выбрана библиотека парсер-комбинаторов Meerkat[1] [4], реализованная на языке Scala. Данная библиотека осуществляет построение леса разбора Binarized Shared Packed Parse Forest (SPPF) [1] для произвольных (в том числе неоднозначных) контекстно-свободных грамматик.

В основе библиотеки лежит четыре базовых комбинатора: terminal, epsilon, seq и rule. Первые два комбинатора представляют собой базовые распознаватели для терминала и пустой строки. Комбинатор seq выполняет последовательную композицию распознавателей. Комбинатор rule используется, для задания правила вывода в терминах контекстно-свободных грамматик: нетерминала и соответствующие ему альтернатив. С помощью этих комбинаторов можно задать произвольную контекстно-свободную грамматику.

Библиотека осуществляет поиск всех возможных способов разбора строки. Это удалось достичь за счет использования техники Continuation-Passing

[1] https://github.com/meerkat-parser/Meerkat

Style (CPS) и специальной процедуры мемоизации. Идея программирования в стиле Continuation-Passing состоит в передаче управления через механизм продолжений. Продолжение в данном контексте представляет собой состояние программы в конкретный момент времени, которое возможно сохранить и использовать для перехода в данное состояние. Для реализации данного подхода был создан такой тип данных как Result[T], который представляет собой монаду и реализует следующие три метода: map, flatMap, orElse. Таким образом, любой результат работы распознавателя можно представить как композиция двух функций, используя метод flatMap, или как комбинацию результатов, используя метод orElse. Для избежания экспоненциальной сложности применяется специальная техника мемоизации, которая сохраняет все продолжения, которые необходимо приенить, и переиспользуются при получении новых рультатов. Продолжение для конкретного символа вычисляется один раз и в дальнейшем возвращается лишь результат.

Для решения поставленной задачи потребовалось изменить тип входных данных на граф и преобразовать базовый комбинатор, разбирающий терминал. В отличие от линейного входа при анализе графа позицией во входном потоке является вершина графа, а понятие "следующего символа" заменяется на множество символов, написанных на исходящих из данной вершины ребрах. Синтаксический разбор при этом продолжается по тому пути, который начинается с ребра с соответствующим символом. Продолжения и его результаты сохраняются для конкретной вершины и переиспользуются, при попадании в ту же вершину. Таким образом решается проблема с графами, которые содержат циклы. Для случая, когда из вершины исходят ребра с одинаковыми символами, результат комбинируется с помощию метода orElse.

В дальнейшем планируется интегрировать библиотеку[2] с промышленной графовой СУБД Neo4J, что позволит использовать данное решение в таких сферах как биоинформатика, логистика, социальные сети.

Список литературы

[1] Afroozeh Ali, Izmaylova Anastasia. Faster, Practical GLL Parsing // Compiler Construction: 24th International Conference, CC 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings / Ed. by Björn Franke. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2015. — P. 89–108. — ISBN: 978-3-662-46663-6. — URL: http://dx.doi.org/10.1007/978-3-662-46663-6$_5$.

[2] Grigorev Semyon, Ragozina Anastasiya. Context-Free Path Querying with Structural Representation of Result // arXiv preprint arXiv:1612.08872. — 2016.

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

[2] https://github.com/sofysmol/Meerkat

[4] Izmaylova Anastasia, Afroozeh Ali, Storm Tijs van der. Practical, General Parser Combinators // Proceedings of the 2016 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation. — PEPM '16. — New York, NY, USA : ACM, 2016. — P. 1–12. — URL: http://doi.acm.org/10.1145/2847538.2847539.

[5] Sevon Petteri, Eronen Lauri. Subgraph queries by context-free grammars // Journal of Integrative Bioinformatics. — 2008. — Vol. 5, no. 2. — P. 100.