Санкт-Петербургский государственный университет

Кафедра Системного программирования

Соловьев Александр Александрович

Разработка архитектуры для унификации синтаксических анализаторов в проекте YaccConstructor

Курсовая работа

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

Санкт-Петербург
2016

Оглавление

Введение

В области синтаксического анализа существует множество различных задач. Наиболее известной из них является задача анализа некоторой последовательности токенов с целью проверки ее выводимости в заданной грамматике и построением соответствующего дерева вывода. Однако данная задача может быть рассмотрена более широко, подвергаться синтаксическому анализу могут не только строки, но и другие структуры данных. Например, в работах [4] и [6] в качестве объекта синтаксического анализа рассматривается граф.

В 2010 году был предложен алгоритм обобщенного анализа Generalized LL(GLL), в основе которого лежит алгоритм нисходящего синтаксического анализа[1]. Данный алгоритм и различные его модификации были реализованы в проекте YaccConstructor[3] независимо друг от друга. Различия их заключаются в первую очередь в структурах данных, которые подаются на вход, а также в возможности построения с помощью этих алгоритмов деревьев вывода. Конечно, отличается также и их внутренняя реализация, но общая структура при этом остается прежней. Обобщение различных версий алгоритма упростило бы поддержку и сопровождение. Теоретически оно возможно, однако на практике возникают различных трудностей, получившееся решение может обладать рядом недостатков по сравнению с набором отдельно реализованных алгоритмов. Например, решение может быть крайне громоздким, что может еще больше усложнить его поддержку, одним же из наиболее вероятных недостаткох, которые могут возникнуть, является значительное падение производительности. Возникать оно может в связи с ростом констант. Например, при попытке представить линейный вход в виде графа, появляются циклы для перебора всех исходящих из вершины ребер, что и приводит к росту константы.

1. Постановка задачи

Целью данной работы является разработка архитектуры для унификации существующих синтаксических анализаторов в проекте YaccConstructor. Для достижения данной цели были поставлены следующие задачи:

- спроектировать архитектуру, позволяющую объединить различные модификации алгоритма;
- реализовать предложенную архитектуру;
- разработать тестовое покрытие;
- провести эксперименты для оценки производительности.

2. Обзор

2.1. YaccConstructor

YaccConstructor – проект, разрабатываемый в лаборатории языковых инструментов JetBrains. В нем занимаются исследованиями и разработками в области лексического и синтаксического анализа. Большинство компонентов проекта реализованы на языке F#, исходники находятся в открытом доступе. Проект имеет модульную архитектуру1, что позволяет собирать требуемый инструмент из существующих модулей: можно выбрать фронтенд, задать требуемые преобразования грамматики и указать генератор. Генераторы предоставляют инструменты, позволяющие по внутреннему представлению грамматики получить полезный для конечного пользователя результат. Примером такого результата являются синтаксические анализаторы.

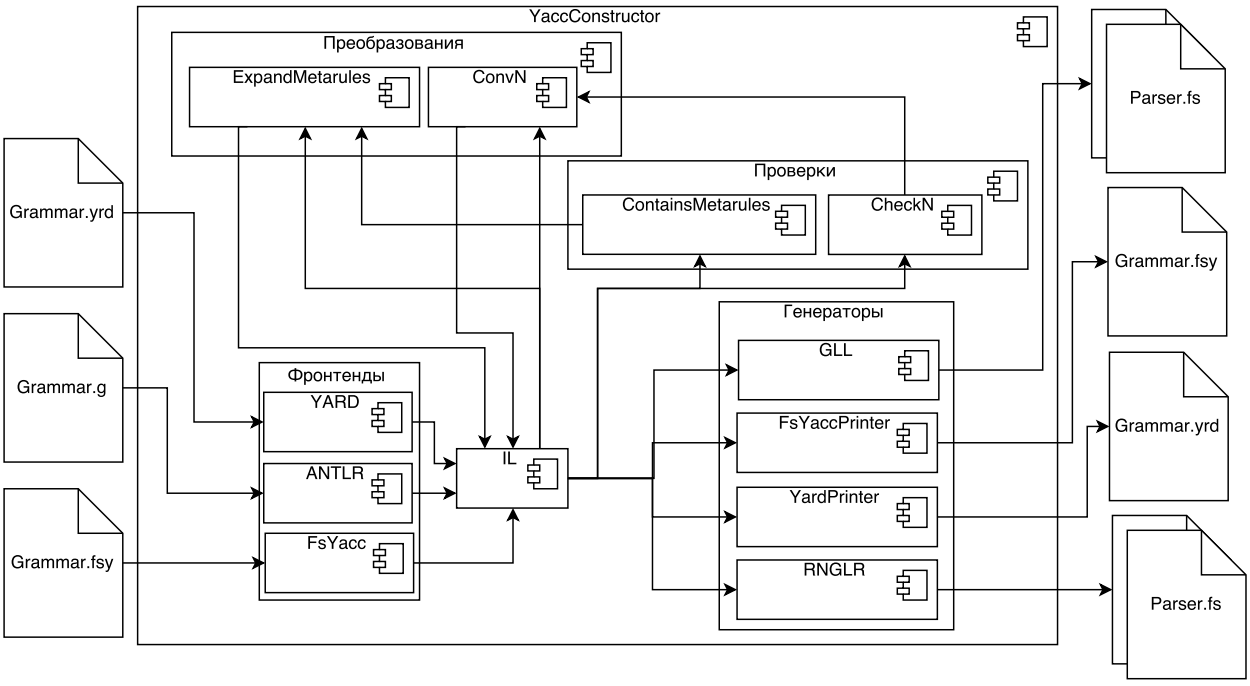

Рис. 1: Архитектура YaccConstructor, заимствована из [5]

2.2. Синтаксические анализаторы, основанные на GLL

Как говорилось выше, на данный момент в проекте реализованы GLL и несколько его модификаций. Различаются они в рассматриваемой структуре данных, в умении строить дерево, а также в том, подается ли на вход алгоритму грамматика или же конечный автомат.

Конечный автомат, упомянутый выше, можно представить в виде набора вершин, которые представляют некоторые состояния, с набором исходящих из них ребер, помеченных некоторыми символами. Переход между вершинами при разборе происходит в соответствии со следующими правилами:

- если в данный момент встречается терминал, происходит переход к некоторому новому состоянию;
- если же встречается нетерминал, данный нетерминал разбирается и уже тогда происходит переход к некоторому новому состоянию.

Далеее перечислены модификации алгоритма, которые на данный момент реализованы в YaccConstructor. Все перечисленные, помимо последней, версии умеют строить деревья вывода:

- стандартная версия алгоритма, принимающая на вход грамматику и последовательность токенов;
- алгоритм, принимающий на вход грамматику и граф ;
- алгоритм, принимающий на вход грамматику и GFG[2];
- алгоритм, принимающий на вход конечный автомат и строку;
- алгоритм, принимающий на вход конечный автомат и граф.

3. Заключение

Достигнуты следующие результаты:

- произведен обзор статей, связанных с предметной областью;
- спроектирована архитектура;
- написан обзор предметной области.

В дальнейшем планируется:

- реализовать предложенную архитектуру;
- разработать тестовое покрытие;
- провести эксперименты для оценки производительности.

Список литературы

[1] E. Scott, A. Johnstone. GLL Parsing // Electron. Notes Theor. Comput. Sci. — 2010. — Vol. 253, no. 7. — P. 177–189.

[2] A Graphical Model for Context Free Grammar Parsing.

[3] YaccConstructor. YaccConstructor // YaccConstructor official page. — URL: `http://yaccconstructor.github.io` (online; accessed: 18.12.2016).

[4] Вербицкая Екатерина Андреевна. Синтаксический анализ регулярных множеств. — 2015.

[5] Григорьев Семен Вячеславович. Синтаксический анализ динамически формируемых программ. — 2016.

[6] Рагозина Анастасия Константиновна. Ослабленный синтаксический анализ динамически формируемых выражений на основе алгоритма GLL. — 2016.