Реализация алгоритма GLR-анализа в функциональной среде для EBNF-грамматик

С. Григорьев,СПбГУ
А. Лукичев, СПбГУ

Одной из задач реинжиниринга программного обеспечения является перевод исходных кодов программных комплексов с устаревших языков и платформ на современные. Весьма существенным этапом в таком переводе является построение анализатора входных языков. Задача усложняется тем, что количество диалектов устаревших языков, многие из которых плохо документированы, и интрументальных языков весьма велико. Эти причины накладывают следующие требования на алгоритмы анализа и построения анализаторов:

1) возможность работы с неоднозначными грамматиками (поддержка диалектов в рамках одной грамматики и простота внесений изменений в грамматику)

2) возможность воспринимать основные типы грамматик, встречающиеся в формальных описаниях языков (как правило, грамматики в EBNF; при этом класс языков соответствует LR)

В рамках рассматриваемого проекта ставится цель разработки прототипа генератора анализаторов, удовлетворяющего основным перечисленным требованиям и предоставляющего, в то же время, ставшие привычными средства автоматизации трансляции (дополнение грамматики атрибутами, средства автоматического восстановления после ошибок, средства диагностики).

Необходимость поддержки неоднозначных грамматик (на практике, как правило, грамматики лишь «слегка» неоднозначные) предопределила выбор алгоритма GLR в качестве анализатора. Работу алгоритма GLR можно рассматривать как параллельное исполнение набора LR-анализаторов. При этом данный набор дополняется процедурой управления магазинами, оптимизирующей представление магазинов путем их «склеивания» и «расклеивания», что позволяет хранить и строить параллельные выводы в рамках одного LR-анализатора, лишь в моменты их различия добавляя параллельный анализатор.

Оказалось, что весьма наглядно такой алгоритм может быть представлен в виде двух взаимно-рекурсивных функций (рекурсивно-восходящий алгоритм, recursive ascent). При этом расклеивание магазина получается естественным образом как ветвление в одной из функций, а обратное склеивание может быть реализовано как кэширование результата функции. В рассматриваемом проекте данный алгоритм реализован на функциональном языке F# в среде .NET, при этом склеивание реализовано с помощью конструкции замыкания.

Естественным образом получается и поддержка регулярных выражений в правых частях правил (EBNF-грамматики). Для этого правая часть правила представляется как конечный автомат. LR-ситуация в таком случае может быть представлена парой: правило (нетерминал+КА) и номер состояния (соответствует позиции маркера в классическом определении). Проблемы определения левой границы отрезка в магазине, соответствующего текущему правилу, в данном подходе не существует, так как стек вызовов рекурсивных функций хранит информацию о начале анализа по правилу.

На данный момент работа над проектом продолжается, но уже получены следующие результаты:

1) по однозначной LR-грамматике строится анализатор с линейной сложностью
2) по неоднозначной грамматике строится анализатор, возвращающий все возможные деревья вывода для данной входной цепочки
3) показана возможность поддержки регулярных выражений в правых частях правил

Список литературы

1)Rene Leermakers Non-deterministic Recursive Ascent Parsing. Philips Research Laboratories, P.O.Box 80.000, 5600 JA Eindhoven, The Netherlands.

2)Larry Morell, David Middleton RECURSIVE-ASCENT PARSING. Arkansas Tech University Russellville, Arkansas.

3)Rene Leermakers Non-deterministic Recursive Ascent Parsing. Philips Research Laboratories, P.O.Box 80.000, 5600 JA Eindhoven, The Netherlands.

4)Ronald Veldena Jade, a recursive ascent LALR(1) parser generator. September 8,1998

5)http://www.research.microsoft.com/fsharp (дистрибутивы и документация по языку F#)