УДК 004.434
DOI: 10.17586/0021-3454-2015-58-11-934-938

ИССЛЕДОВАНИЕ И РАЗРАБОТКА ЯЗЫКОВОГО ИНСТРУМЕНТАРИЯ НА ОСНОВЕ PEG-ГРАММАТИКИ

Ю. Д. Кореньков, И. П. Логинов

Университет ИТМО, 197101, Санкт-Петербург, Россия
E-mail: ivan.p.loginov@gmail.com

Рассматриваются средства создания предметно-ориентированных языков, так называемый языковой инструментарий, и их особенности. Предложено новое решение — прототип языкового инструментария, основанного на PEG-грамматике, который позволяет интуитивно понятным образом описывать грамматики предметно-ориентированных языков. Приведен сравнительный анализ ключевых возможностей разработанного прототипа и существующих решений.

Ключевые слова: *языковой инструментарий, предметно-ориентированный язык, языково-ориентированное программирование.*

Введение. Создание программного обеспечения необходимо во множестве отраслей промышленности и экономики. Примером могут служить задачи моделирования, для которых характерны большие объемы вычислений и данных, сложные логические структуры данных, построение сложных таблиц входных/выходных данных и пр. Ручное программирование подобных задач — процесс достаточно медленный и связанный с большим количеством ошибок.

Решить эту проблему позволяет автоматическое программирование. Для компьютерного генерирования программ необходимо создание формального языка, который может быть использован для описания решения задачи в терминах ее предметной области. Такие языки называются предметно-ориентированными [1] или DSL (Domain Specific Language).

Хорошо известным примером таких языков служит SQL (Structered Query Language). Этот язык был создан специально для работы с реляционными базами данных. На основе SQL были разработаны некоторые языки для манипуляции данными, например язык интегрированных запросов LINQ (Language Integrated Query) [2], встроенный в C# для выполнения запросов к коллекциям данных.

В настоящей момент создано множество DSL для использования в различных предметных областях: например, языки для описания аппаратуры интегральных схем (VHDL, Verilog), языки для символьных вычислений (Mathematica, Maple, Maxima) и др. С другой стороны, продолжается разработка новых решений по созданию средств языково-ориентированного (ЯО) программирования, так называемого языкового инструментария [3]. Собственно, новой разработке — созданию прототипа языкового инструментария — и посвящена настоящая статья.

Предлагаемый прототип позволит:

— повысить удобство использования создаваемых DSL за счет интеграции с существующими средами разработки (например, Visual Studio, Eclipse) или создания нового standalone (автономного) средства для ЯО-программирования;

— повысить производительность труда разработчиков.

Состояние предметной области. Одним из самых известных примеров инструментария для языково-ориентированного программирования является система JetBrains MPS (Meta Programming System) [4]. В течение процесса разработки языка MPS предоставляет пользователю возможности, характерные для современных IDE, такие как: автоматическое завершение текста, выделение синтаксических конструкций в режиме реального времени, проверка

ошибок и др. MPS поставляется с большим количеством примеров, которые расширяют функциональность языка Java. Следует отметить, что MPS не зависит от языков программирования.

Другим примером языкового инструментария является проект Nitra [5]. В настоящее время Nitra поддерживает разработку генераторов синтаксических анализаторов (далее — парсеров, от англ. parser — производить грамматический разбор) для .NET Framework. Однако эта система спроектирована таким образом, что позволяет создавать парсеры и для других платформ — потенциально Nitra может использоваться для генерации кода парсеров на языке C или для виртуальных машин, таких как LLVM.

Решение, предлагаемое в данной статье, позволяет создавать предметно-ориентированные языки, и его можно использовать совместно с множеством языков программирования (более пятидесяти), таких как C#, F#, C++/CLI и др. Представленное решение не использует дополнительных зависимостей от других проектов, как, например, Nitra, который построен на базе языка Nemerle.

Конфигурируемый парсер. Для создания предметно-ориентированного языка необходимо средство, позволяющее анализировать исходный код, описываемый разными грамматиками. Таким образом, парсер должен быть гибко конфигурируемым.

Инструменты для построения классических парсеров не предоставляют достаточной свободы при конфигурировании анализатора, так как основаны на порождающих грамматиках (согласно иерархии Хомского), которые приводят к необходимости сложных преобразований формального языка и парсера для него.

Возможным решением проблемы может служить класс грамматик типа PEG (Parsing Expression Grammars) [6], т.е. грамматик, „разбирающих“ выражения. PEG-грамматики состоят из множества правил (множества нетерминалов, согласно Хомскому). Парсер для грамматики легко может быть построен как стековая машина, которую можно конфигурировать динамически. Кроме того, PEG-грамматики не требуют отдельной фазы разбора текста, так как правила грамматики, по сути, описывают сам процесс разбора.

Подход к реализации. В ходе исследования реализован анализатор, который обрабатывает PEG-грамматику, представленную как набор именованных правил, описывающих разбираемые выражения. Данный анализатор предоставляет возможности для описания правил грамматик различной сложности. Правила можно параметризировать при помощи выражений-аргументов.

Также для грамматики в целом возможно устанавливать шаблон пропуска символов. Такой шаблон определяет игнорируемые анализатором фрагменты текста, что позволяет различать эти фрагменты с точки зрения их необходимости для пользователя или модели анализа, представляющей результат разбора. Примером использования шаблона служат документирующие комментарии, которые должны быть особым образом выделены в текстовом редакторе для легкого визуального восприятия.

Результат анализа текста представляет собой дерево элементов, называемое StringTreeNode. Каждый элемент описывает фрагмент разбираемого текста в соответствии с правилами грамматики. По окончании анализа это дерево можно автоматически преобразовать в дерево любых объектов, например в абстрактное синтаксическое дерево компилятора.

Используемый в предлагаемом прототипе язык для описания грамматики предусматривает написание грамматики в интуитивно понятной форме. Поддерживаются регулярные выражения и атрибуты для правил (написанные в квадратных скобках, например: [right]power: expr '^' expr). Приведем список поддерживаемых атрибутов:

left — помечает выражение как левоассоциативное;

right — помечает выражение как правоассоциативное;

OmitPattern — объявляет шаблон для пропуска;

RootRule — объявляет корневое правило;

RewriteRecursion — помечает правило, содержащее альтернативы, как необходимое для автоматического устранения рекурсии.

Альтернативные варианты в процессе разбора могут быть указаны следующим образом:

— как выражение типа „или“ при помощи символа ‘|’: a|b|c;

— как расширяемое правило, которое состоит из нескольких вложенных правил.

Расширяемые правила дают возможность расширить однажды описанную грамматику без необходимости ее изменения.

Отладчик грамматик. Для упрощения процесса разработки грамматик для DSL был создан специальный отладчик, позволяющий детализировать процесс анализа текста в соответствии с грамматикой, а также изучить деревья разбора. Окно приложения отладчика разделено на следующие области:

1) текстовое поле для ввода анализируемого текста;

2) дерево разбора с применением фильтров;

3) полное дерево разбора;

4) текстовое поле, содержащее объявление грамматики;

5) протокол процесса разбора;

6) использованные в процессе разборе правила (в виде дерева).

Дерево правил грамматики способствует поиску структурных ошибок в выражениях, описывающих правила, а подробный журнал процесса разбора текста — поиску логических ошибок. Применение полного дерева разбора очень полезно при анализе поведения парсера.

Генерация кода. Один из важных аспектов использования предметно-ориентированных языков — кодогенерация, за счет которой возможно взаимодействие кода, написанного вручную, и кода, полученного в результате генерации. При этом целесообразно избегать генерации промежуточного кода, а DSL интегрировать в язык программирования общего назначения.

При генерации используются разные подходы, специфические для языков программирования, например:

— в языке C возможно использование текстовых макросов [7], с помощью которых могут быть описаны термины предметной области; затем в процессе компиляции (точнее, работы транслятора макросов) таким образом описанные термины будут заменены на конструкции на языке C;

— во многих языках программирования присутствуют элементы „синтаксического сахара“, позволяющие скрыть сложные конструкции при помощи простых выражений, которые заменяются компилятором на полноценные языковые конструкции.

Кодогенерация — важная проблема в процессе интеграции разрабатываемого решения в IDE, поскольку генератор необходимо формировать по результатам анализа. В настоящее время существует возможность генерировать исходя из описания грамматик текстовую модель на языке C#. Исходный код может быть скомпилирован при помощи компилятора C# для дальнейшего использования с любыми инструментами и языками, поддерживаемыми CLI (Common Language Infrastructure — общеязыковая инфраструктура, широко распространенная в мире разработки программного обеспечения).

Поддержка интегрированной среды разработки. Инструментарий для поддержки процессов создания и использования DSL средой разработки должен предоставлять:

— текстовый редактор для создания DSL и редактирования текста с его использованием;

— способ интеграции созданного DSL в проект, для которого этот язык предназначен.

В качестве решения предлагается расширение для IDE Microsoft Visual Studio, которое предоставляет возможности для выделения синтаксиса грамматики, описывающей язык, а также для выделения фрагментов текста на созданном DSL. В текстовом редакторе возмож-

но установить произвольную схему выделения (подсветки) — набор стилей визуального представления текста для правил и терминальных символов; также существует возможность переключения между несколькими схемами при помощи выпадающего списка над окном текстового редактора.

Заключение. Разработанный прототип языкового инструментария предоставляет синтаксис, понятный пользователю на интуитивном уровне. В настоящий момент существует поддержка этого инструментария в Microsoft Visual Studio, также разработан специальный инструмент для отладки грамматик. В дальнейшем планируется существенно расширить функциональность этого прототипа, в частности, добавив возможности автоматического завершения текста в соответствии с разработанной грамматикой.

СПИСОК ЛИТЕРАТУРЫ

1. *Martin Fowler Website*: Domain Specific Language [Электронный ресурс]: <http://martinfowler.com/bliki/DomainSpecificLanguage.html>.
2. Pro LINQ: Language Integrated Query in C# 2010. M.: Williams, 2011. 656 p.
3. *Martin Fowler Website*: Language Workbenches: The Killer-App for Domain Specific Languages? [Электронный ресурс]: <http://martinfowler.com/bliki/LanguageWorkbench.html>.
4. *Martin Fowler Website*: A Language Workbench in Action — MPS [Электронный ресурс]: <http://martinfowler.com/articles/mpsAgree.html>.
5. An Introduction to Nitra [Электронный ресурс]: <http://blog.jetbrains.com/blog/2013/11/12/an-introduction-to-nitra/>.
6. *Ford B.* Parsing Expression Grammars: A Recognition-Based Syntactic Foundation. Cambridge, MA: Massachusetts Inst. of Technology, 2004.
7. International Standard ISO/IEC 9899:201x “Programming Languages — C”. 2000. 166 p.

Сведения об авторах

Юрий Дмитриевич Кореньков — Университет ИТМО; кафедра информатики и прикладной математики-1; ассистент

Иван Павлович Логинов — магистрант; Университет ИТМО; кафедра информатики и прикладной математики-1; E-mail: ivan.p.loginov@gmail.com

Рекомендована кафедрой информатики и прикладной математики-1

Поступила в редакцию 31.08.15 г.

Ссылка для цитирования: *Кореньков Ю. Д., Логинов И. П.* Исследование и разработка языкового инструментария на основе PEG-грамматики // Изв. вузов. Приборостроение. 2015. Т. 58, № 11. С. 934—938.

PEG-BASED LANGUAGE WORKBENCH RESEARCH AND DEVELOPMENT

Yu. D. Korenkov, I. P. Loginov

ITMO University, 197101, St. Petersburg, Russia
E-mail: ivan.p.loginov@gmail.com

Tools for creation of a domain-specific language (called the language workbench) and their features are considered. A new solution based on the PEG-grammars is proposed – a prototype of language workbench that provides an intuitive way for description of a domain specific language grammar. A comparison of key features of the developed prototype with the opportunities provided by existing solutions is presented.

Keywords: language workbench, domain-specific language, language-oriented programming.

Data on authors

Yury D. Korenkov — ITMO University, Department of Computer Science and Applied Mathematics-1; Assistant;

Ivan P. Loginov — Master Sci.; ITMO University, Department of Computer Science and Applied Mathematics-1; E-mail: ivan.p.loginov@gmail.com

For citation: *Korenkov Yu. D., Loginov I. P.* PEG-based language workbench research and development // Izvestiya Vysshikh Uchebnykh Zavedeniy. Priborostroenie. 2015. Vol. 58, N 11. P. 934—938 (in Russian).

DOI: 10.17586/0021-3454-2015-58-11-934-938