Динамически формируемый код: синтаксический анализ контекстно-свободной аппроксимации

Ковалев Д. А., dmitry.kovalev-m@ya.ru
Григорьев С. В., semen.grigorev@jetbrains.com
Санкт-Петербургский государственный университет,
Россия, 199034, Санкт-Петербург, Университетская наб. 7/9;
Лаборатория языковых инструментов JetBrains

Аннотация

Многие программы в процессе работы формируют из строк исходный код на некотором языке программирования и передают его для исполнения в соответствующее окружение (пример — dynamic SQL). Для статической проверки корректности динамически формируемого выражения используются различные методы, одним из которых является синтаксический анализ регулярной аппроксимации множества значений такого выражения. Аппроксимация может содержать строки, не принадлежащие исходному множеству значений, в том числе синтаксически некорректные. Анализатор в данном случае сообщит об ошибках, которые на самом деле отсутствуют в выражении, генерируемом программой. В докладе будет описан алгоритм синтаксического анализа более точной, чем регулярная, контекстно-свободной аппроксимации динамически формируемого выражения.

Ключевые слова: синтаксический анализ, динамически формируемый код, контекстно-свободные грамматики, GLL, GFG, dynamic SQL

Современные языки программирования общего назначения поддерживают возможность работы со строковыми литералами, позволяя формировать из них выражения при помощи строковых операций. Строковые выражения могут создаваться динамически, с использованием таких конструкций языка, как циклы и условные операторы. Данный подход широко используется, например, при формировании SQL-запросов к базам данных из программ, написанных на Java, C# и других высокоуровневых языках.

Недостаток такого метода генерации кода заключается в том, что формируемые выражения не проходят статические проверки на корректность и безопасность, что приводит к ошибкам времени исполнения и усложняет сопровождение системы. Включение обработки динамически формируемых строковых выражений в фазу статического анализа осложняется тем, что такие выражения, в общем случае, невозможно представить в виде линейного потока, который принимают на вход традиционные алгоритмы лексического/синтаксического анализа.

Для решения данной проблемы были разработаны специальные методы статического анализа множества значений формируемого выражения. Как

правило, язык, на котором написана исходная программа, тьюринг-полон, что делает невозможным проведение точного анализа. Ряд существующих методов использует для анализа *регулярную аппроксимацию* — множество строк, генерируемых программой, аппроксимируется сверху регулярным языком, и анализатор работает с его компактным представлением, таким как регулярное выражение или конечный автомат.

В магистерской диссертации [5] был описан алгоритм, позволяющий проводить синтаксический анализ регулярной аппроксимации (детерминированного конечного автомата) множества значений динамически формируемого выражения. Основой для данного алгоритма служит алгоритм обобщенного синтаксического анализа Generalized LL (GLL, [7]). Такой подход позволяет получать конечное представление множества деревьев вывода (SPPF, [6]) корректных строк, содержащихся в аппроксимации. Это представление может быть использовано для проведения более сложных видов статического анализа и для целей реинжиниринга.

GLL позволяет работать с произвольными КС-грамматиками, в том числе неоднозначными. Такая возможность достигается путем использования специальной структуры стека (GSS) и механизма *дескрипторов*. Дескриптор хранит в себе информацию о состоянии анализатора в определенный момент времени, достаточную для продолжения процесса анализа с этого момента. Оригинальный GLL-алгоритм был модифицирован для работы с нелинейным входом (конечный автомат представляется в виде графа). Дескрипторы нового алгоритма хранят номер вершины входного графа вместо позиции в линейном потоке. Также, на шаге исполнения просматривается не единственный входной символ, а все ребра, исходящие из текущей вершины.

Мы расширили описанный подход, изменив алгоритм таким образом, чтобы он мог обрабатывать более точную, чем регулярная, *контекстно-свободную аппроксимацию* значений динамически формируемого выражения. Использование более точной аппроксимации позволяет снизить количество ложных синтаксических ошибок, возникающих в результате того, что аппроксимирующее множество содержит строки, осутствующие среди значений искомого выражения. Под контекстно-свободной аппроксимацией здесь подразумевается грамматика, описывающая контекстно-свободный язык, который содержит в качестве подмножества возможные значения выражения. Идея использования такой аппроксимации была заимствована из существующих в данной области работ [1; 2].

Наш алгоритм принимает на вход графовое представление аппроксимирующей КС-грамматики. В качестве такого представления был выбран Grammar Flow Graph (GFG, [4]). Алгоритм последовательно обходит узлы GFG, производя синтаксический анализ порождаемых им строк. Для правильного построения таких строк алгоритм должен манипулировать дополнительным стеком. Информация о текущей вершине данного стека записывается в дескрипторы. Другой особенностью работы с GFG является то, что он, в отличие от регулярной аппроксимации — детерминированного конечного автомата, допускает возможность неоднозначного выбора пути обхода. Подобная ситуация возникает при наличии в исходной грамматике нескольких продукций, содержащих в левой части одинаковый нетерминал. Механизм дескрипторов позволяет решать проблему недетерминированного выбора пути — для каждого из возможных вариантов создается отдель-

ный дескриптор, который добавляется в очередь исполнения.

Алгоритм был реализован на языке программирования F# в рамках проекта YaccConstructor. Исходный код доступен по ссылке: `https://github.com/YaccConstructor/YaccConstructor`. Результаты проведенных синтетических тестов показали снижение количества ложных синтаксических ошибок при анализе динамически формируемого кода. В дальнейшем планируется провести аппробацию на реальных данных. Задача синтаксического анализа графов также возникает в области биоинформатики [5]. Предполагается, что наш алгоритм может увеличить производительность анализа метагеномных сборок за счет использования более компактного представления входных данных — из конечного автомата, описывающего сборку, может быть извлечена КС-грамматика [3], графовое представление которой содержит меньше состояний.

Список литературы

1. *Doh K.-G.*, *Kim H.*, *Schmidt D. A.* Abstract LR-Parsing // Formal Modeling: Actors, Open Systems, Biological Systems: Essays Dedicated to Carolyn Talcott on the Occasion of Her 70th Birthday / под ред. G. Agha, O. Danvy, J. Meseguer. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2011. — С. 90—109. — ISBN 978-3-642-24933-4. — DOI: 10.1007/978-3-642-24933-4_6. — URL: http://dx.doi.org/10.1007/978-3-642-24933-4_6.
2. *Minamide Y.* Static approximation of dynamically generated web pages //. — In Proceedings of the 14th International Conference on World Wide Web, WWW '05. ACM, 2005. — С. 432—441.
3. *Nevill-Manning C. G.*, *Witten I. H.* Identifying Hierarchical Structure in Sequences: A Linear-time Algorithm // J. Artif. Int. Res. — USA, 1997. — Сент. — Т. 7, № 1. — С. 67—82. — ISSN 1076-9757. — URL: http://dl.acm.org/citation.cfm?id=1622776.1622780.
4. *Pingali K.*, *Bilardi G.* A Graphical Model for Context-Free Grammar 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 / под ред. B. Franke. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2015. — С. 3—27. — ISBN 978-3-662-46663-6. — DOI: 10.1007/978-3-662-46663-6_1. — URL: http://dx.doi.org/10.1007/978-3-662-46663-6_1.
5. *Ragozina A.* GLL-based relaxed parsing of dynamically generated code: Master's Thesis / Ragozina Anastasiya. — SPbU, 2016.
6. *Rekers J. G.* Parser generation for interactive environments: PhD Thesis / Rekers Joan Gerard. — Citeseer, 1992.
7. *Scott E.*, *Johnstone A.* GLL parsing // Electronic Notes in Theoretical Computer Science. — 2009. — Т. 253, № 7. — ISSN 1571-0661. — DOI: 10.1016/j.entcs.2010.08.041.