### **УДК 519.685.3**

### **Гумин Е.Д. (3 курс, каф. Системного программирования, СПбГУ),**

### **Григорьев С.В. (магистр ИТ, ст.пр., каф. Системного программирования СПбГУ)**

### **ИСПОЛЬЗОВАНИЕ СИМВОЛЬНЫХ КОНЕЧНЫХ ПРЕОБРАЗОВАТЕЛЕЙ ДЛЯ ЛЕКСИЧЕСКОГО АНАЛИЗА ДИНАМИЧЕСКИ ФОРМИРУЕМОГО КОДА**

При разработке программного обеспечения может возникать необходимость использования динамически формируемого кода – кода, который формируется во время выполнения другой программы, с помощью строковых операций, условных операторов, циклов. Как пример можно рассмотреть генерацию HTML внутри JavaScript-кода, динамическое создание SQL запросов в программах на C++ или C#. Подсветка синтаксиса и диагностика ошибок в динамически формируемом коде может существенно упростить его разработку и сопровождение. Для предоставления подобной функциональности необходим лексический анализ динамически формируемого кода [1], задача которого ‒ выделить лексемы во входном потоке, сохранив привязку к исходному коду. Однако в отличие от анализа обычного кода, динамически формируемый код не представим в виде линейного потока. Одно из возможных решений – построить два конечных преобразователя [2], один из них строится по регулярной аппроксимации множества возможных значений строкового выражения, а другой является лексическим анализатором языка, и применить к ним операцию композиции. Этот подход используется в проекте YaccConstructor [3, 4], в рамках которого выполняется данная работа, но производительность алгоритма композиции оказалось недостаточной.

Цель данной работы – исследовать способы увеличения производительности лексического анализа. Одной из причин проблем в скорости работы является быстрое разрастание конечных преобразователей на больших алфавитах (на каждый символ необходимо добавлять по переходу). Этого недостатка лишены символьные конечные преобразователи (Symbolic State Finite Transducer, SFT) [5] — преобразователи, на рёбрах которых могут быть не одиночные символы, а выражения над ними. Данный формализм реализован в библиотеке Automata [6], которая также содержит реализацию символьных конечных автоматов и ряда других формализмов. Кроме того, в ней реализованы вспомогательные средства для использования библиотеки с другими инструментами, такими как языки Bek [7] и Bex [8], SMT-решателем Z3 [9]).

Таким образом, в рамках работы необходимо оценить применимость библиотеки к задачам, решаемым в проекте YaccConstructor и сравнить результаты работы алгоритма композиции на символьном и классическом конечном преобразователе. Тестирование библиотеки показало, что она на несколько порядков проигрывает в скорости существующему алгоритму, что связано с использованием очень общего подхода, в результате чего выполняется большое количество действий, не нужных при решении частной задачи. Тем не менее, предполагается, что использование языка Bex может оказаться удобным для спецификации лексических анализаторов. Таким образом, применение символьных преобразователей для лексического анализа динамически формируемого кода выглядит перспективным, однако может потребоваться существенный пересмотр реализации.

ЛИТЕРАТУРА:

[1] Semen Grigorev, Ekaterina Verbitskaia, Andrei Ivanov, Marina Polubelova, and Ekaterina Mavchun. 2014. String-embedded language support in integrated development environment. In Proceedings of the 10th Central and Eastern European Software Engineering Conference in Russia (CEE-SECR '14). ACM, New York, NY, USA, Article 21, 11 pages.

[2] Полубелова М.И. Лексический анализ динамически формируемых строковых выражений // Дипломная работа кафедры системного программирования СПбГУ. –– 2015. –– URL:<http://se.math.spbu.ru/SE/diploma/2015/bmo/444-Polubelova-report.pdf>.

[3] Сайт проекта YaccConstructor. –– URL:<http://yaccconstructor.github.io>.

[4] Кириленко Я.А. Григорьев С. В. Авдюхин Д. А. Разработка синтаксических анализаторов в проектах по автоматизированному реинжинирингу информационных систем. –– 2013. –– С. 94–98.

[5] Margus Veanes, Pieter Hooimeijer, Benjamin Livshits, David Molnar, and Nikolaj Bjorner. 2012. Symbolic finite state transducers: algorithms and applications. In Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '12). ACM, New York, NY, USA, 137-150.

[6] Репозиторий библиотеки Automata. –– URL: <http://github.com/AutomataDotNet/Automata>.

[7] Сайт проекта Bek. –– URL: <http://research.microsoft.com/en-us/projects/bek>.

[8] Сайт проекта Bex. –– URL: <http://research.microsoft.com/en-us/projects/bex>.

[9] Репозиторий проекта Z3. –– URL: <http://github.com/Z3Prover/z3>.