



Использование символьных конечных преобразователей для лексического анализа динамически формируемого кода

Автор: Егор Гумин

Научный руководитель: ст.пр. С.В. Григорьев

Санкт-Петербургский государственный университет
Кафедра системного программирования

26 апреля 2016г.

Предметная область

При обработке динамически формируемого кода возникает задача обеспечения подсветки синтаксиса и анализа ошибок

```
private void Go(int cond){  
    string columnName = cond > 3 ? "X" : (cond < 0 ? "Y" : "Z");  
    string queryString = "SELECT name" + columnName + " FROM table";  
    Program.ExecuteImmediate(queryString);  
}
```

- Сложности: условия, циклы, строковые операции (replace, concat)
- Примеры: HTML в JavaScript, SQL в C#
- Необходимость: поддержка и рефакторинг существующего кода

Задача лексического анализа — выделить лексемы во входном потоке, сохранив привязку к исходному коду

Композиция двух конечных преобразователей — основная операция при лексическом анализе:

$$\text{ComposeFST} \langle \text{symbol} * \text{position}, \text{token} \rangle = \\ \text{CodeFST} \langle \text{symbol} * \text{position}, \text{symbol} \rangle \circ \text{LexerFST} \langle \text{symbol}, \text{token} \rangle$$

Результат композиции — FST, который можно применить к входному потоку и получить расположение в нем лексем

Finite State Transducer (FST)

- Быстро разрастается (по количеству дуг) при большом алфавите
- Чаще всего не поддерживает бесконечные алфавиты

Symbolic Transducer (ST)

- Каждому переходу можно сопоставить формулу (например, регулярное выражение)
- Возможна работа с бесконечным алфавитом
- Однозначность — каждому входу соответствует только один выход

- YaccConstructor — исследовательский проект в области лексического и синтаксического анализа
- В YaccConstructor используется FST, а не ST
- В библиотеке Microsoft.Automata, разработанной в Microsoft Research, реализован ST, некоторые другие формализмы и операции над ними

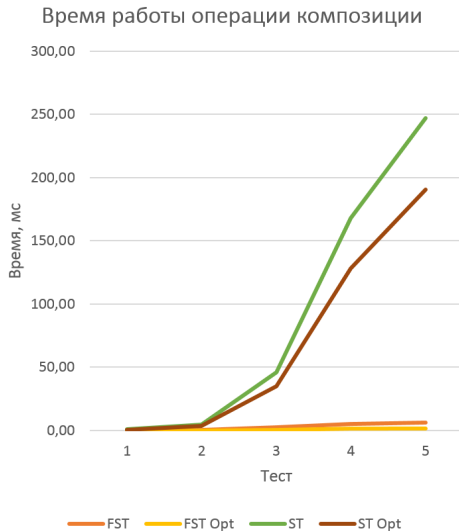
Цель: исследовать возможность применения ST для лексического анализа

Задачи:

- Сравнить время работы операции композиции
- Написать набор тестов для FST и ST
- Реализовать лексический анализатор языка арифметических выражений на языке Bек, эквивалентный используемому в YaccConstructor

Результаты эксперимента

| Тест | Ребра | Вершины |
|------|-------|---------|
| 1 | 1 | 2 |
| 2 | 6 | 5 |
| 3 | 25 | 20 |
| 4 | 50 | 5 |
| 5 | 60 | 50 |



- ST применимы для лексического анализа
- Библиотека Microsoft.Automata разрабатывалась для решения задач формальной верификации
- Язык Vex удобен для описания лексических анализаторов
- Необходима реализация ST, адаптированную под задачи проекта YaccConstructor

- Microsoft.Automata — .NET библиотека, содержащая алгоритмы для композиции и анализа регулярных выражений, автоматов и преобразователей
- C#, F# — языки семейства .NET
- Bek, Bex (Microsoft Research) — предметно-ориентированные языки для написания и анализа строковых преобразователей
- Intel VTune Amplifier — профилировщик производительности

- Исследованы возможности библиотеки Microsoft.Automata
- Проведено сравнение производительности алгоритма операции композиции на ST в библиотеке с производительностью операции композиции над FST в проекте YaccConstructor
- На основании полученных результатов принято решение о написании собственной реализации ST