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 мая 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

Лексический анализ

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

Композиция двух конечных преобразователей — основная операция при лексическом анализе:

$$ComposeFST\langle symbol * position, token\rangle =$$

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

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

Реализация лексических анализаторов

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

Обзор

- 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

Постановка задачи

Цель: исследовать возможность применения ST для лексического анализа

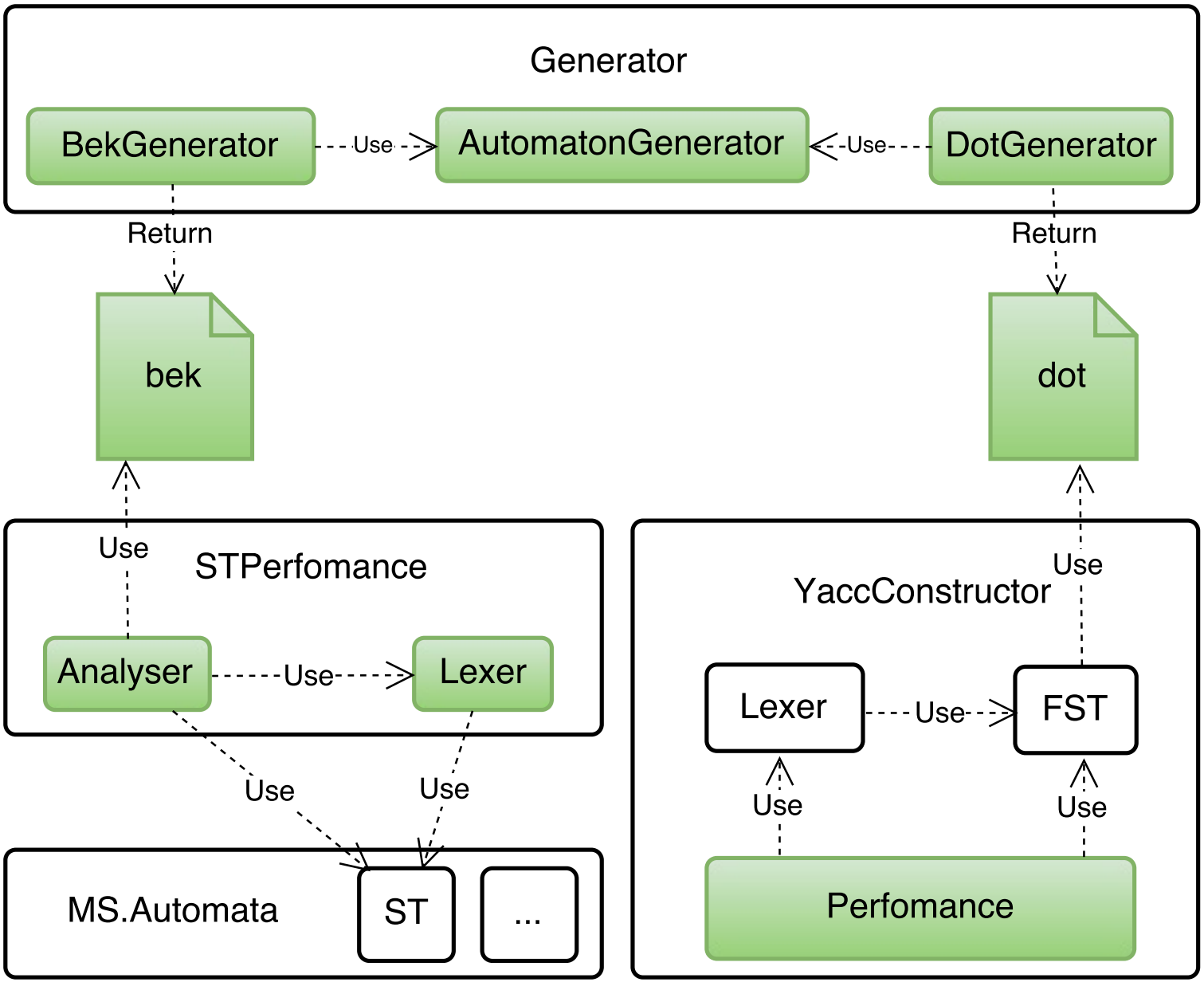
Задачи:

- Изучить возможности библиотеки Microsoft.Automata
- Провести сравнение производительности алгоритма операции композиции на ST в библиотеке с производительностью операции композиции над FST в проекте YaccConstructor
- На основании полученных результатов сделать выводы о применимости библиотеки в проекте YaccConstructor

Изучение библиотеки

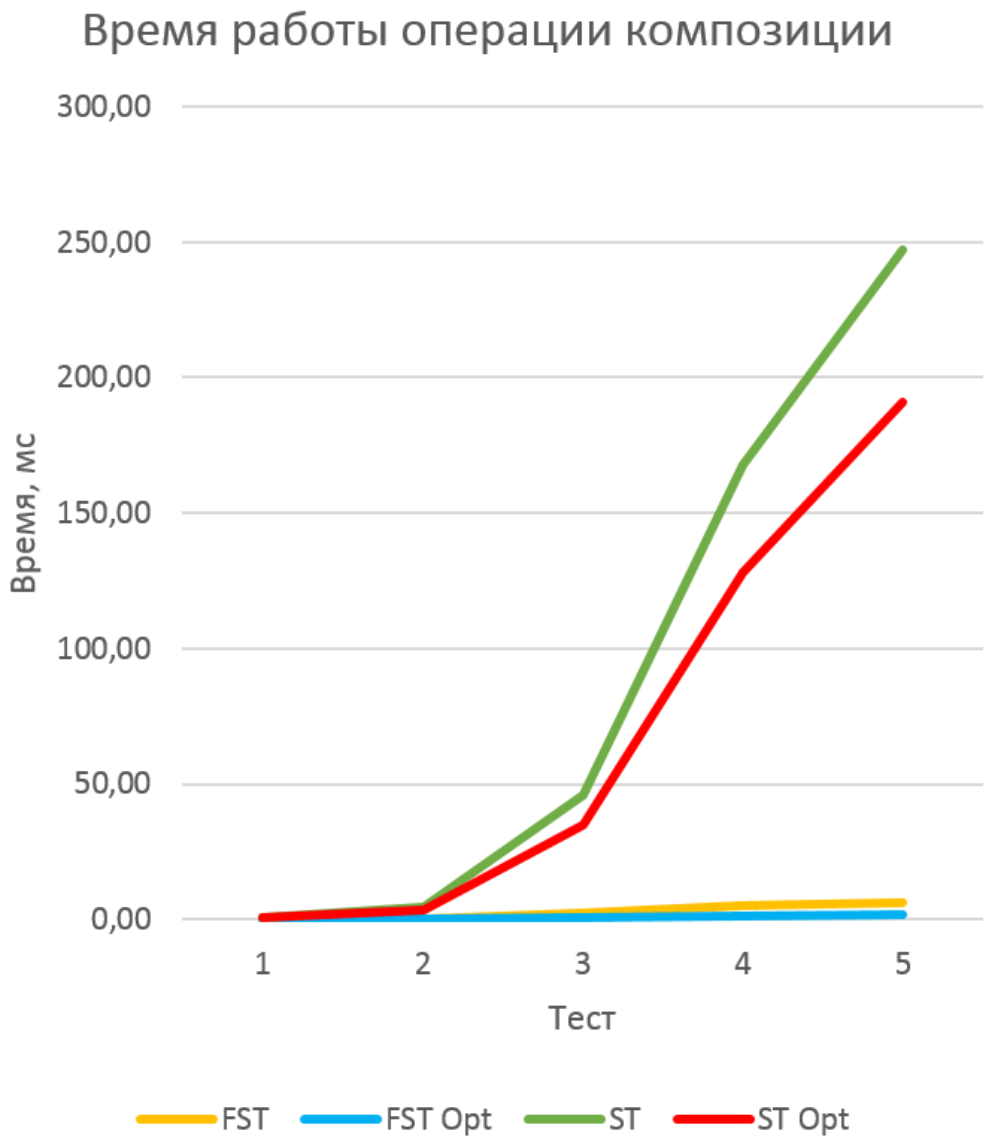
- Разрабатывалась для решения задач формальной верификации
- Активно использует SMT-решатель Z3
- Содержит реализацию множества формализмов (символьные автоматы, преобразователи)
- Для описания преобразователей широко используются языки Bek, Bex

Диаграмма решения



Результаты эксперимента

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



Выводы

- ST применимы для лексического анализа
- Библиотека Microsoft.Automata разрабатывалась для решения задач формальной верификации и не обладает необходимой производительностью
- Необходима реализация ST, адаптированная под задачи проекта YaccConstructor

Результаты

- 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 сделаны выводы о необходимости написания собственной реализации ST
- Результаты работы представлены на конференции «Современные технологии в теории и практике программирования», тезисы опубликованы в сборнике материалов конференции