!´

Статический анализ динамически формируемых строковых выражений

Автор: Григорьев Семён

Лаборатория JetBrains на Математико-Механическом факультете
Санкт-Петербургского государственного университета

1 апреля 2015г

Встроенные языки

- Встроенный SQL

```
let p cond fldLst =
    let mutable flds = "id"
    for fld in fldLst do
        flds <- flds + ", " + fld
    let tbl = if cond then "table1" else "table2"
    execute ("SELECT" + flds + "FROM" + tbl)
```

- JavaScript в Java

```
String script =
    "function hello(name)  print('Hello, ' + name); ";
engine.eval(script);
Invocable inv = (Invocable) engine;
inv.invokeFunction("hello", "Scripting!!!" );
```

Проблемы

- Динамически формируемые выражения — код на некотором языке и его нужно соответствующим образом поддерживать и обрабатывать
- Однако для стандартных инструментов это просто строки
  - Ошибки в динамически формируемых выражениях обнаруживаются лишь во время выполнения
  - Нет поддержки в IDE
  - Затруднён реинжиниринг ПО, разработанного с использованием встроенных языков
  - Увеличивается уязвимость систем (SQL-инъекции)

Встроенный SQL

```
let f () =
    let p cond fldLst =
        let mutable flds = "id"
        for fld in fldLst do
            flds <- flds + ", " + fld
        let tbl = if cond then "table1" else "table2"
        execute ("SELECT" + flds + "FROM" + tbl)
    p false ["x";"y"] |> print
    p true ["x";"z"] |> print
```

Вложенность языков

- $L_1 = L(f)$ — язык, задаваемый программой
- $L_2 = L(G)$ — “эталонный” язык
- L_2 — КС (почти)
- L_1 — КС ?
- Вопрос: $L_1 \subseteq L_2$?
- Разрешим если L_1 — регулярный, L_2 — детрминированный КС

Построение регулярной аппроксимации

- L_1 — не регулярный
- Построим L_R — регулярный: $L_1 \subseteq L_R$
  - Такой L_R можно построить: Fang Yu, Muath Alkhalaf, Tevfik Bultan, and Oscar H. Ibarra. 2014. Automata-based symbolic string analysis for vulnerability detection
    - ★ Учитываются строковые опреации
- $L_R = L_1 + L_\Delta$

Лексический анализ

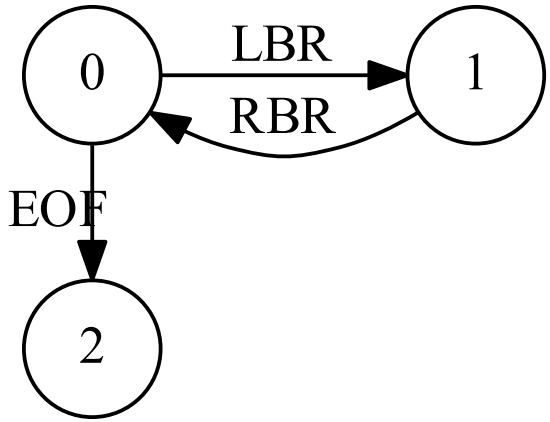
Входной граф	Результат токенизации

Обобщённый синтаксический анализ

- Generalized LR parsing (GLR)
- Предназначен для работы с произвольными КС грамматиками
  - Shift-Reduce и Reduce-Reduce конфликты
- Использует организованный в виде графа стек (GSS)
- Использует компактное представление леса вывода (SPPF)
  - Переиспользование общих узлов
- Мы используем RNGLR — поддерживает произвольные КС грамматики

Синтаксический анализ строковых выражений

Входной граф



Грамматика: $s : LBR\ s\ RBR\ s \mid \varepsilon$

Результат разбора

Синтаксический анализ регулярного множества

- Входной граф
- Грамматика: $expr : NUM\ PLUS\ NUM$
- GSS

Синтаксический анализ регулярного множества

- Завершаемость
  - Поиск неподвижной точки: для каждой вершины исходного графа вычисляем множество всех возможных LR-состояний
  - Состояний в каждой вершине не больше чем всего состояний LR-автомата для данной граматики
  - Все рёбра в GSS уникальны — не более чем полный граф

Синтаксический анализ регулярного множества

- Корректность
  - Предполагаем, что RNGLR корректен
  - Ситуации, не попадающие в классический RNGLR — “замыкание” цикла: можем потерять свёртки
  - Чтобы этого не произошло, нужно запоминать свёртки, в которых участвовало состояние: проходящие свёртки

Результаты

- Предложен алгоритм синтаксического анализа регулярных множеств
  - Строится структура данных, содержащая деревья вывода всех корректных цепочек анализируемого множества
- На основе алгоритма реализован генератор синтаксических анализаторов
- Реализован генератор лексических анализаторов для динамически формируемых строковых выражений
- Генераторы и набор вспомогательных библиотек объединены в пакет для разработки инструментов статического анализа строковых выражений
- С использованием пакета создан плагин для ReSharper, поддерживающий T-SQL встроенный в C#

Дальнейшее развитие

- Диагностика ошибок
  - Применение GLL
- Семантические действия над SPPF
- Трансформации встроенных языков

Основные существующие решения

- Java String Analyzer — статический анализатор динамических выражений для Java (Регулярный в КС)
- Alvor — синтаксический анализ регулярной аппроксимации (Регулярный в КС)
- PHP String Analyzer — статический анализатор динамических выражений для PHP (КС в КС)
- Kyung-Goo Doh, Hyunha Kim, David A. Schmidt — комбинация LR-анализа и анализа потока данных для обработки встроенных языков (КС в КС)

Информация о проекте

- Контакты
  - Григорьев Семён: Semen.Grigorev@jetbrains.com
- Сообщество GitHub: https://github.com/YaccConstructor

Лексический анализ: пример

- Входной граф
- Спецификация

```
PLUS : "+"
POW : "**"
MULT: "*"
```

- Результат токенизации

Регулярная аппроксимация: строковые операции

- `result = replace(src_str, old, new)`
- `src_str:`
- `  old:     new:`
- `result:`

Литература

- RNGLR: *Scott E., Johnstone A.* Right Nulled GLR Parsers.
- Alvor: *Aivar Annamaa, Andrey Breslav, Jevgeni Kabanov, and Varmo Vene*. 2010. An Interactive Tool for Analyzing Embedded SQL Queries.
- JSA: *Aske Simon Christensen, Anders Møller, and Michael I. Schwartzbach*. Precise Analysis of String Expressions.
- PHPSA: *Yasuhiko Minamide*. 2005. Static approximation of dynamically generated Web pages.
- Abstract parsing: *Kyung-Goo Doh, Hyunha Kim, and David A. Schmidt.* 2011. Abstract LR-parsing.
- Построение регулярной апроксимации: *Fang Yu, Muath Alkhalaf, Tevfik Bultan, and Oscar H. Ibarra.* 2014. Automata-based symbolic string analysis for vulnerability detection.

Литература

- О инструментах для работы со встроенными языками
  - *Hung Viet Nguyen,Christian Kästner, Tien N. Nguyen.*Varis: IDE Support for Embedded Client Code in PHP Web Applications.
  - *Zachary Smith.* 2011. Development of tools to manage embedded SQL.
- Зачем могут быть нужны трансформации
  - *Martin Mariusz Lester.* 2013. Position paper: the science of boxing.
  - *Martin Lester, Luke Ong, Max Schaefer.* Information Flow Analysis for a Dynamically Typed Functional Language with Staged Metaprogramming.