Динамически формируемый код: синтаксический анализ контекстно-свободной аппроксимации

Автор: Дмитрий Ковалев

Санкт-Петербургский государственный университет
Лаборатория языковых инструментов JetBrains

4 апреля 2017г.

Динамически формируемый код

- SQL-запросы в C#

```
private void Example (bool cond) {
    string columnName = cond ? "name" : "address";
    string queryString =
        "SELECT id, " + columnName + " FROM users";
    Program.ExecuteImmediate(queryString);
}
```

Динамически формируемый код

- SQL-запросы в C#

```
private void Example (bool cond) {
    string columnName = cond ? "name" : "address";
    string queryString =
        "SELECT id, " + columnName + " FROM users";
    Program.ExecuteImmediate(queryString);
}
```

- Генерация HTML-страниц в PHP

```
<?php
    $name = 'your name';
    echo '<table>
        <tr><th>Name</th></tr>
        <tr><td>'.$name.'</td></tr>
        </table>';
?>
```

Статический анализ динамически формируемого кода

- Необходимо анализировать каждое возможное значение выражения
- В общем случае задача неразрешима
- Для анализа используется аппроксимация множества значений

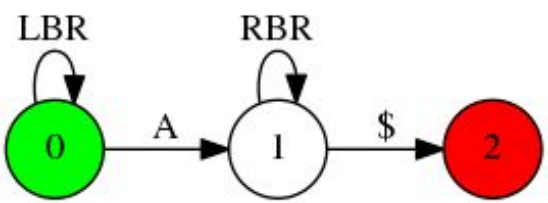
Статический анализ динамически формируемого кода

- Необходимо анализировать каждое возможное значение выражения
- В общем случае задача неразрешима
- Для анализа используется аппроксимация множества значений
- Общая схема:
  - построение аппроксимации
  - лексический анализ
  - **синтаксический анализ**
  - семантический анализ

Синтаксический анализ регулярной аппроксимации

```
expr = "a"
while ...
  expr = "(" + expr + ")"
print expr
```

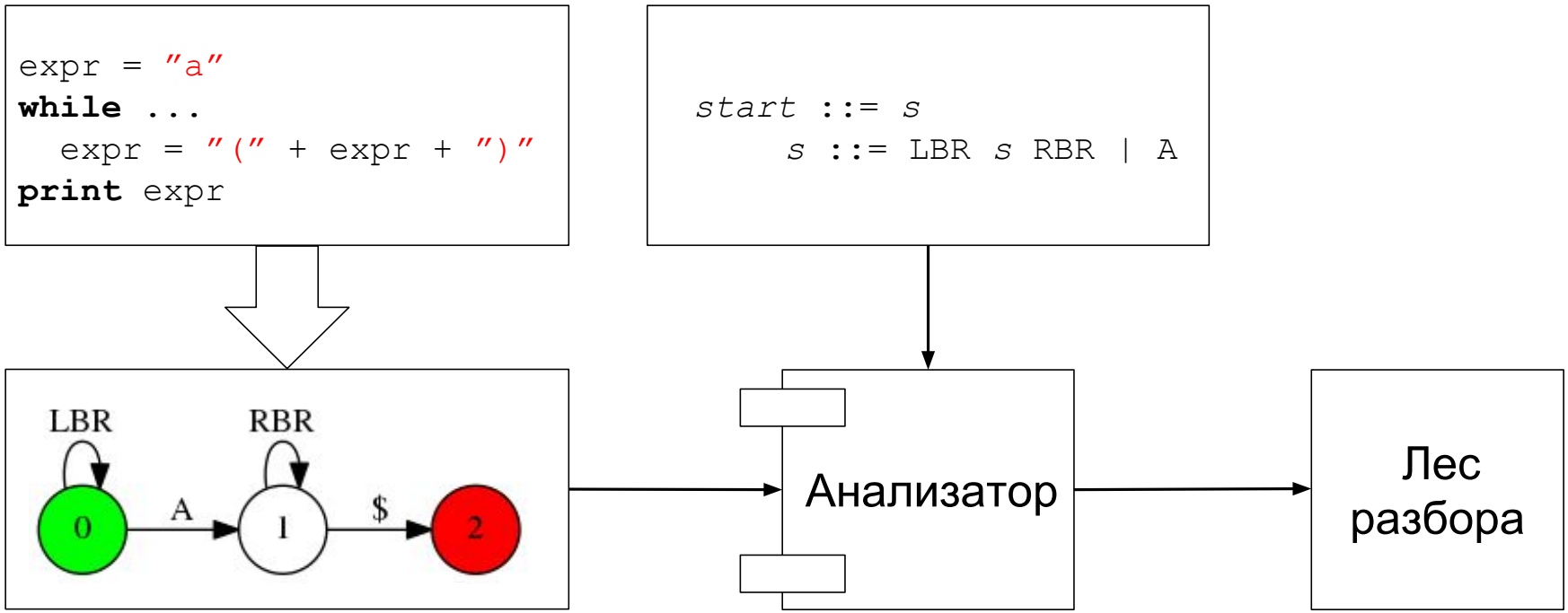
```
start ::= s
    s ::= LBR s RBR | A
```



Анализатор

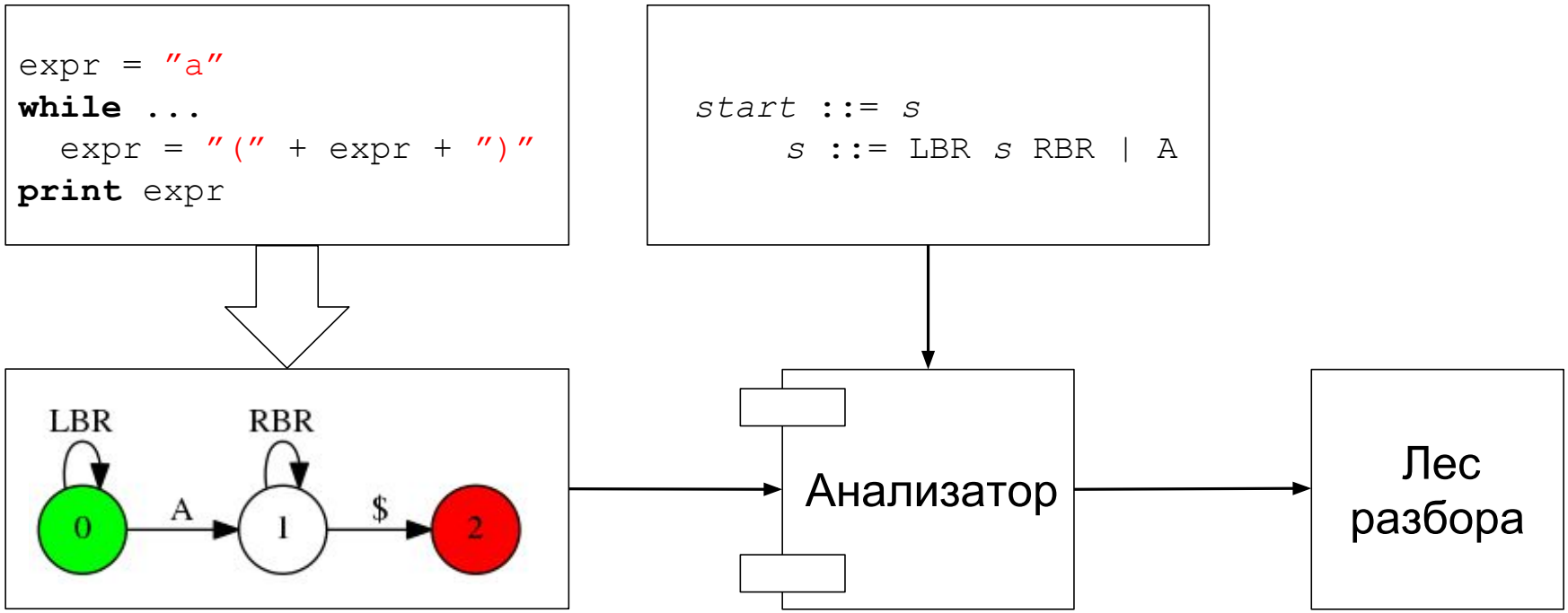
Лес разбора

Синтаксический анализ регулярной аппроксимации



- Проверка включения регулярного языка в LL(k)
- Лес разбора — множество деревьев разбора корректных строк

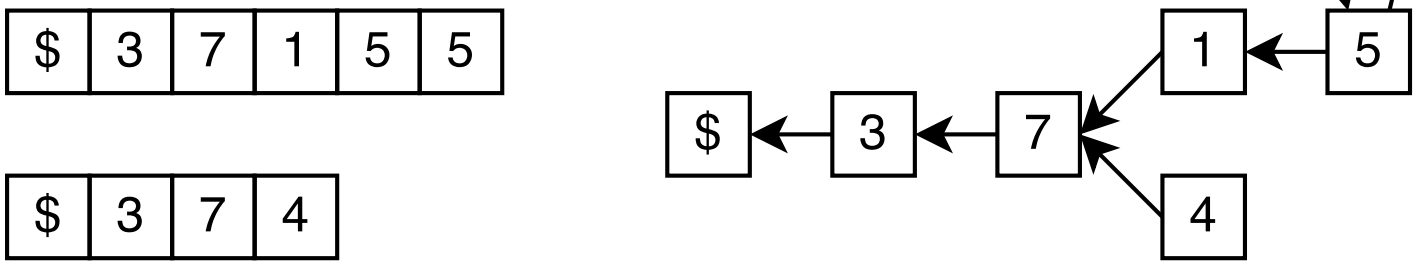
Синтаксический анализ регулярной аппроксимации



- Существующие решения основаны на ...
  - GLR-алгоритме (Alvor, 2010)
  - RNGLR (Е. Вербицкая, 2015)
  - **GLL** (А. Рагозина, 2016)

Generalized LL

- Работает с произвольными KC-грамматиками
- Результат — Shared Packed Parse Forest (SPPF)
- Структурированный в виде графа стек (GSS)
  - Экономия памяти
  - Гарантирует завершаемость алгоритма



GLL и нелинейный вход

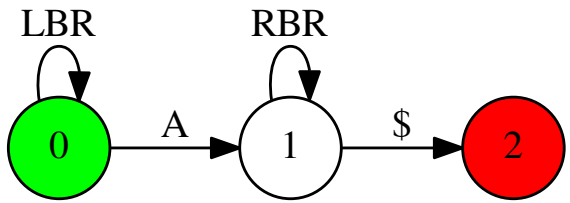
- Механизм дескрипторов — (L, u, i, w), где
  - L — позиция в грамматике вида $A \to \alpha \cdot X \beta$
  - u — вершина GSS
  - i — позиция во входной строке
  - w — узел SPPF

GLL и нелинейный вход

- Механизм дескрипторов — (L, u, i, w), где
  - L — позиция в грамматике вида $A \to \alpha \cdot X \beta$
  - u — вершина GSS
  - i — ~~позиция во входной строке~~ номер вершины автомата
  - w — узел SPPF
- Вместо следующего символа просматриваются метки всех исходящих ребер

Неточность регулярной аппроксимации

```
expr = "a"
while ...
    expr = "(" + expr + ")"
print expr
```

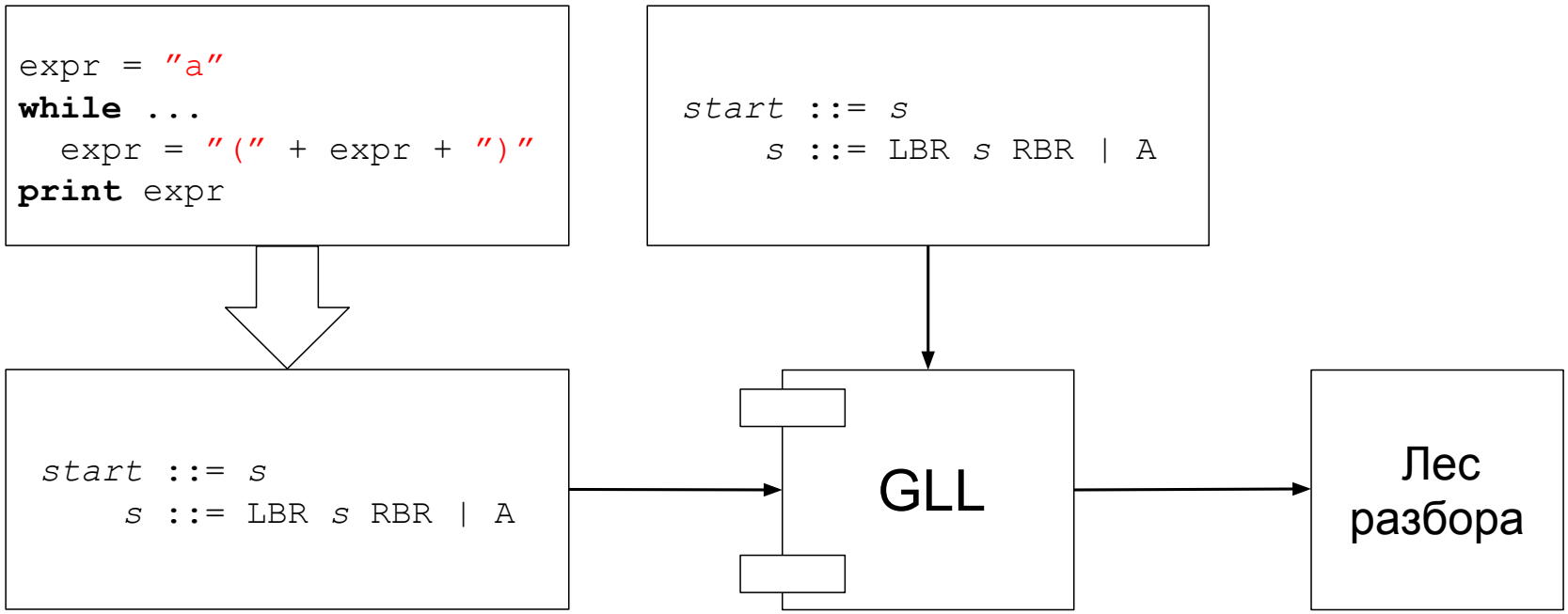


Порождаемый язык: $(^n\ a\ )^n$

Порождаемый язык: $(^*\ a\ )^*$

- Сообщения об ошибках, отсутствующих в строках из исходного множества значений

Синтаксический анализ КС-аппроксимации



- Получение КС-аппроксимации множества значений выражения (Y. Minamide, 2005)

Разрешимость

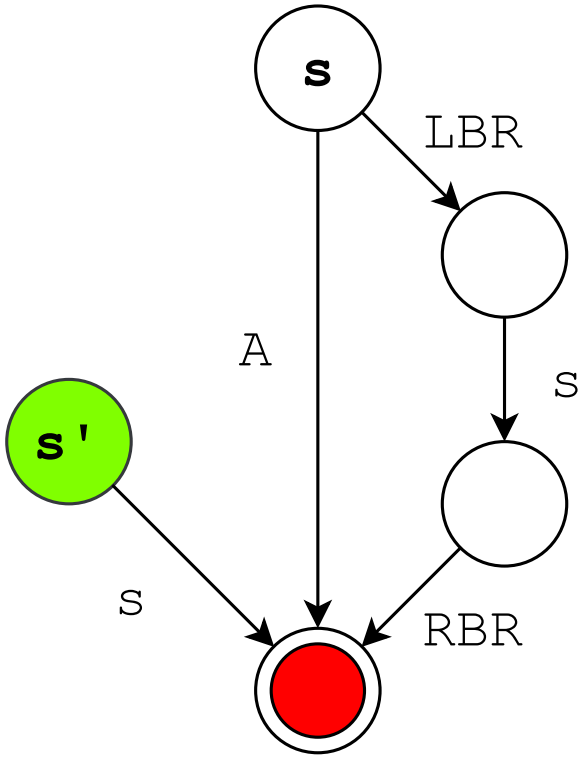
- Проверка включения КС-языков — неразрешимая задача
- Класс КС-языков не замкнут относительно пересечения

Разрешимость

- Проверка включения КС-языков — неразрешимая задача
- Класс КС-языков не замкнут относительно пересечения
- Для некоторых подклассов задача разрешима
- Приближенные решения:
  - Doh, K. G. Abstract Parsing: Static Analysis of Dynamically Generated String Output Using LR-Parsing Technology

Графовое представление грамматики

```
s' ::= s
s  ::= LBR s RBR | A
```



- Ребро с нетерминалом — переход в стартовую вершину данного нетерминала

Модификация алгоритма

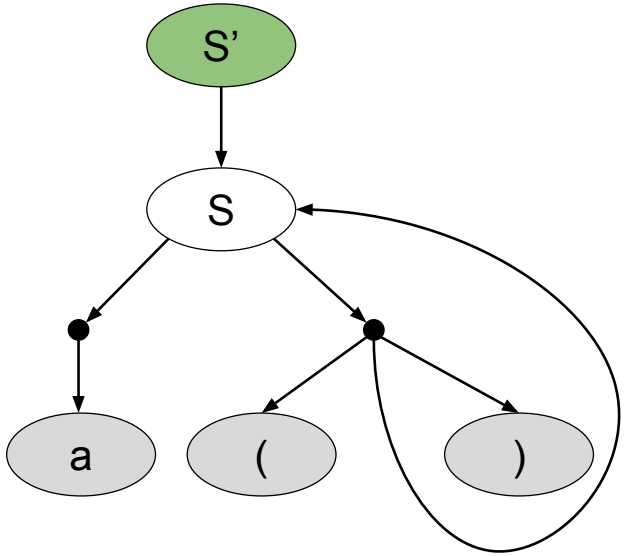
- Необходимо поддерживать два стека (GSS)

Модификация алгоритма

- Необходимо поддерживать два стека (GSS)
- Расширенные дескрипторы — (L, u_1, i, u_2, w), где
  - L — позиция в грамматике вида $A \to \alpha \cdot X \beta$
  - u_1 — вершина первого GSS
  - i — номер вершины автомата (позиция во второй грамматике)
  - u_2 — вершина второго GSS
  - w — узел SPPF

Результат работы алгоритма

```
expr = "a"
while ...
    expr = "(" + expr + ")"
print expr
```



Аппроксимация	Кол-во ошибок
Регулярная	7
КС	0

Заключение

- Представлен алгоритм анализа KC-аппроксимации динамически формируемого кода на основе GLL
- Алгоритм реализован в рамках проекта YaccConstructor