
Десятая независимая научно-практическая конференция
"Разработка ПО 2014"
23-25 октября, Москва


# Инструментальная поддержка встроенных языков в интегрированных средах разработки

**Автор:** Григорьев Семён

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

24 октября 2014г.

# Встроенные языки

- Динамический SQL

```
IF @X = @Y
    SET @TABLE = '#table1'
ELSE
    SET @TABLE = 'table2'
EXECUTE
    ('SELECT x FROM' + @TABLE + ' WHERE ISNULL(n,0) > 1')
```

- 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, который нужно соответствующим образом поддерживать и обрабатывать
  - Ошибки в динамически формируемых выражениях обнаруживаются лишь во время выполнения
  - Поддержка в IDE
  - Реинжиниринг ПО, разработанного с использованием встроенных языков

# Актуальность

- Современные технологии: ORM, LINQ и т.д.
- **Но**
  - Многое уже написано и оно требует поддержки, сопровождения
  - Альтернатив динамическому SQL пока мало

# Предлагаемое решение

- Статическая обработка встроенных языков
  - Поддержка в IDE
    - ★ Многие ошибки можно искать без запуска программы
    - ★ Автодополнение, рефакторинги
  - Реинжиниринг
    - ★ Статический анализ
    - ★ Трансформация (трансляция)

# Существующие решения

- Alvor – плагин для Eclipse для статической проверки встроенного в Java SQL
- Java String Analyzer – статический анализатор динамических выражеинй для Java
- PHP String Analyzer – статический анализатор динамических выражеинй для PHP
- PHPStorm – IDE для PHP с поддержкой HTML, CSS, JavaScript
- IntelliLang – плагин к PHPStorm и IDEA, осуществляющий поддержку различных языков

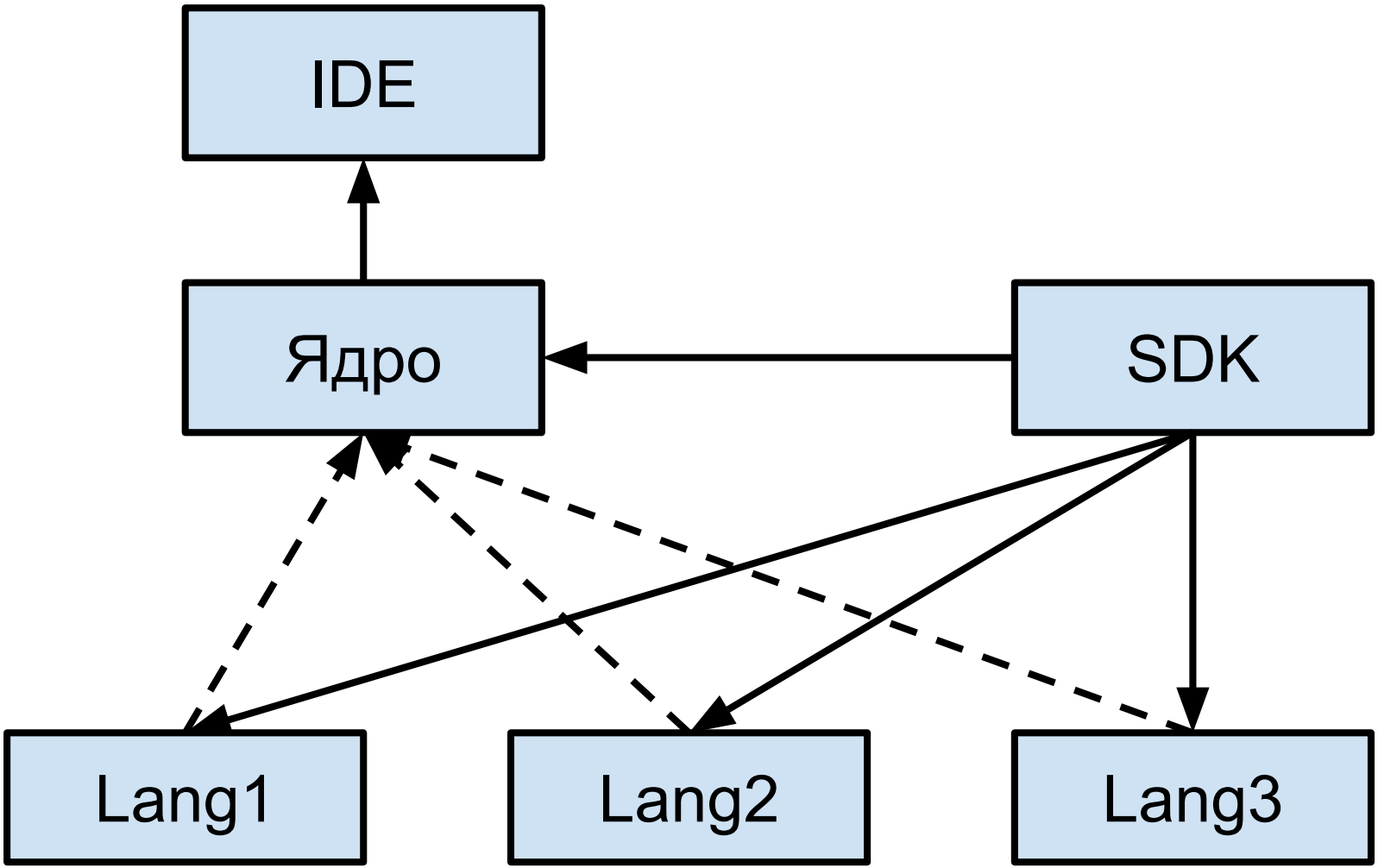
- Предоставляемая функциональность часто скудна
- Часто поддержка других языков возможна только путём изменения исходного кода инструмента

# Цели

- Платформа для создания инструментов анализа встроенных языков
  - Расширяемость в смысле поддержки других языков
  - Расширяемость в смысле предоставляемой функциональности
- Плагин для MS Visual Studio на основе ReSharper
  - Демонстрация возможностей платформы
  - Поддержка встроенных языков в MS Visual Studio

# Языковые расширения

- Поддержка нового языка – создание плагина на основе общей функциональности

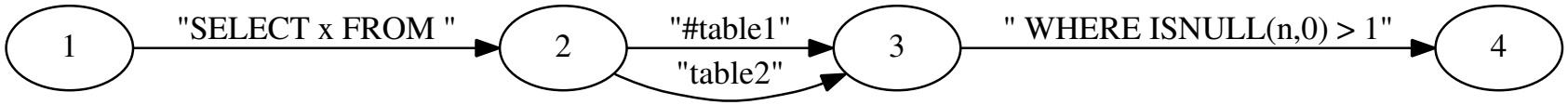


# Как это работает: абстрактный анализ

- Kyung-Goo Doh, Hyunha Kim, David A. Schmidt
  - Комбинация LR-анализа и анализа потока данных для обработки встроенных языков
- Для каждого выражения строится конструкция, аппроксимирующая множество его возможных значений
  - Data-flow уравнение
  - **Конечный автомат**
  - Регулярное выражение
- Выполнение лексического, синтаксического анализа над конечным автоматом (графом)

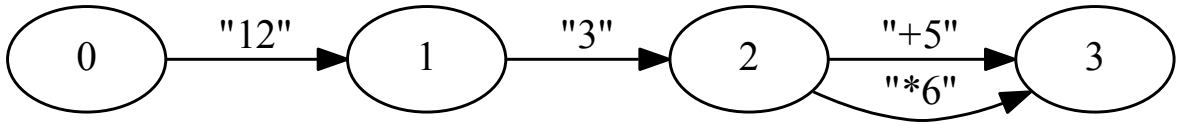
# Пример

- ```
  IF @X = @Y
      SET @TABLE = '#table1'
  ELSE
      SET @TABLE = 'table2'
  EXECUTE
      ('SELECT x FROM ' + @TABLE + ' WHERE ISNULL(n,0) > 1')
  ```
- Множество значений:
  {'SELECT x FROM #table1 WHERE ISNULL(n,0) > 1' ;
  'SELECT x FROM table2 WHERE ISNULL(n,0) > 1'}
- Аппроксимация:

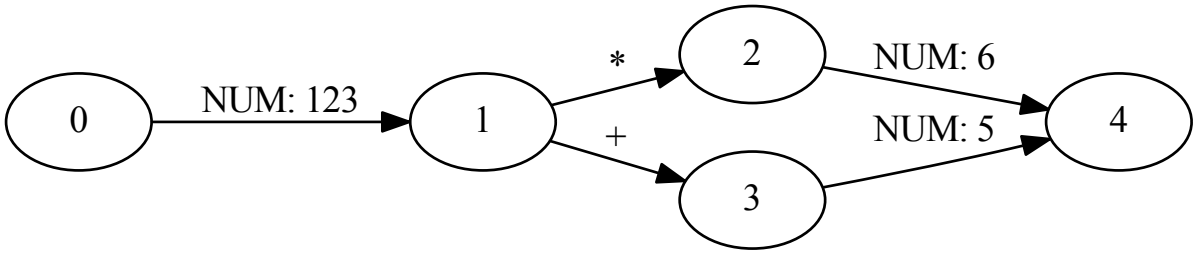

  ```

# Абстрактный лексический анализ

- Аппроксикация (граф со строками на рёбрах) $\rightarrow$ граф с токенами на рёбрах
  - Привязка к литералу в исходном коде
  - Точная привязка внутри литерала
  - Обработка рваных токенов
- Например:
  - Входной граф




  - Результат лексического анализа

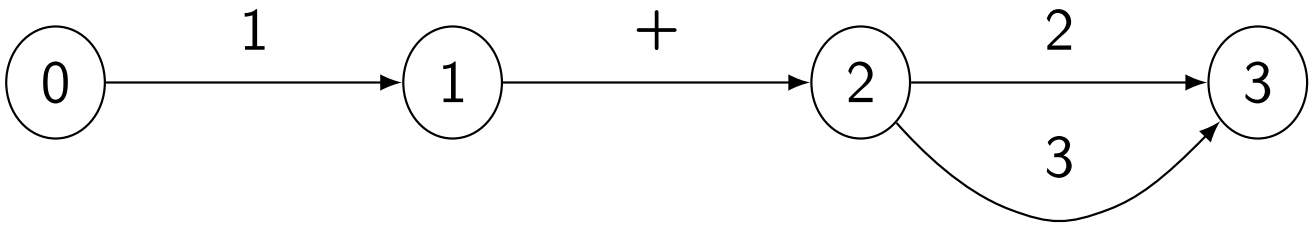


# Обобщённый синтаксический анализ

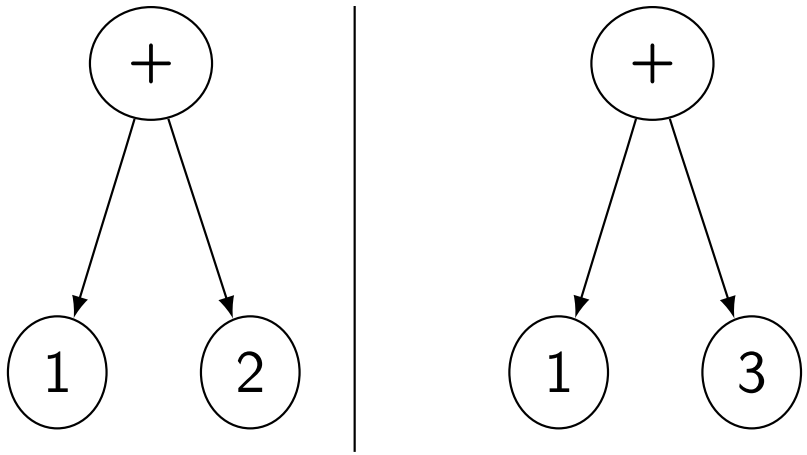
- Generalized LR parsing (GLR)
- Предназначен для работы с произвольными КС грамматиками
  - Shift-Reduce и Reduce-Reduce конфликты
- Использует организованный в виде графа стек (GSS)
- Использует компактное представление леса вывода (SPPF)
  - Переиспользование общих узлов

# Абстрактный синтаксический анализ

- Добавим Shift-Shift "конфликты" – ситуации, возникающие при ветвлении входного потока
- Получилось расширение GLR
- Вход:

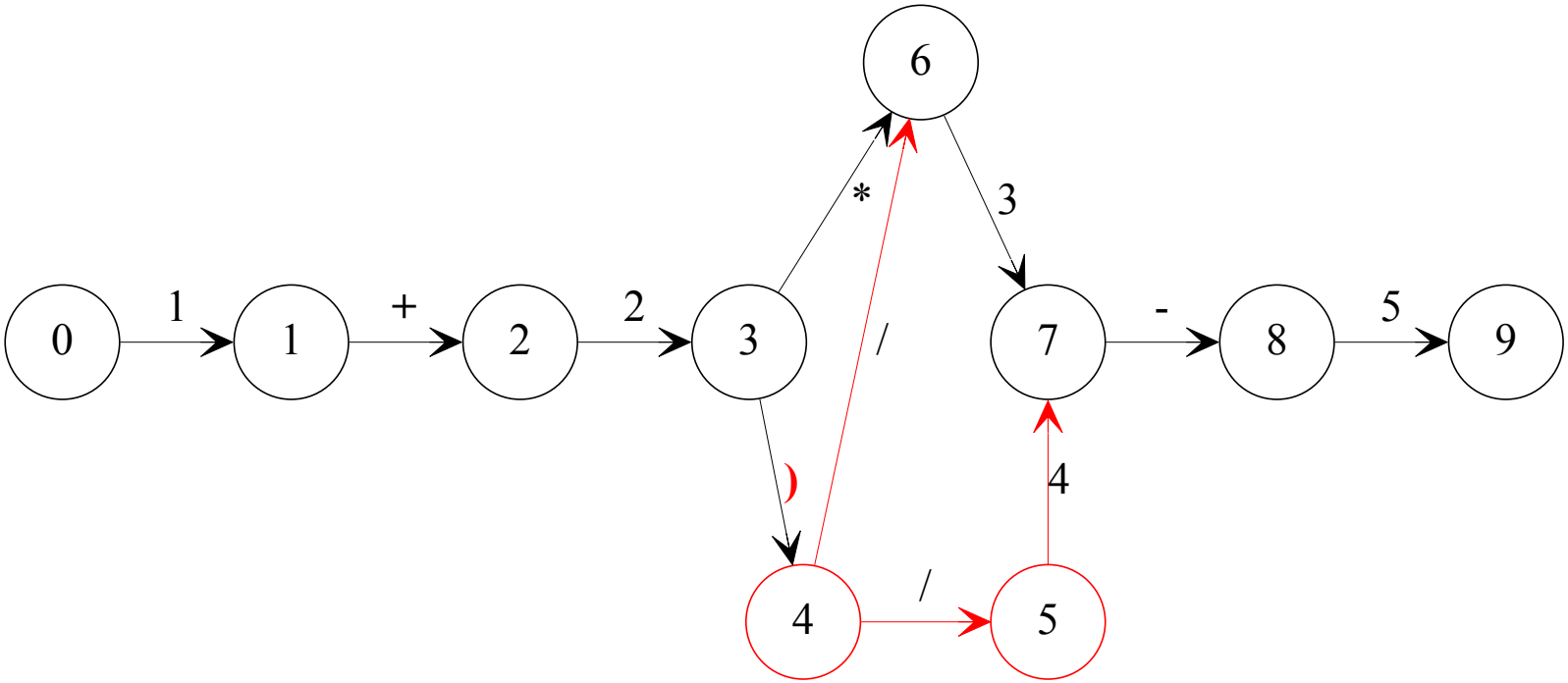



- Результат:



# Диагностика ошибок

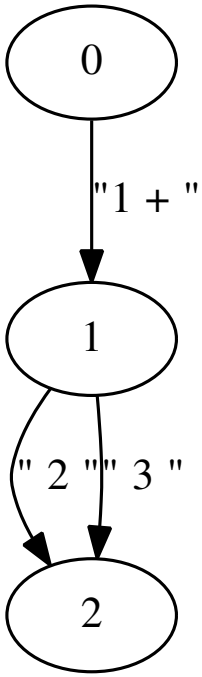
- Нужно возвращать лес разбора для корректных выражений и список ошибок для некорректных
- Для обычного GLR умершая ветка — нормально, для абстрактного не всегда
- Пропускать токены в графе сложнее, чем в линейном потоке

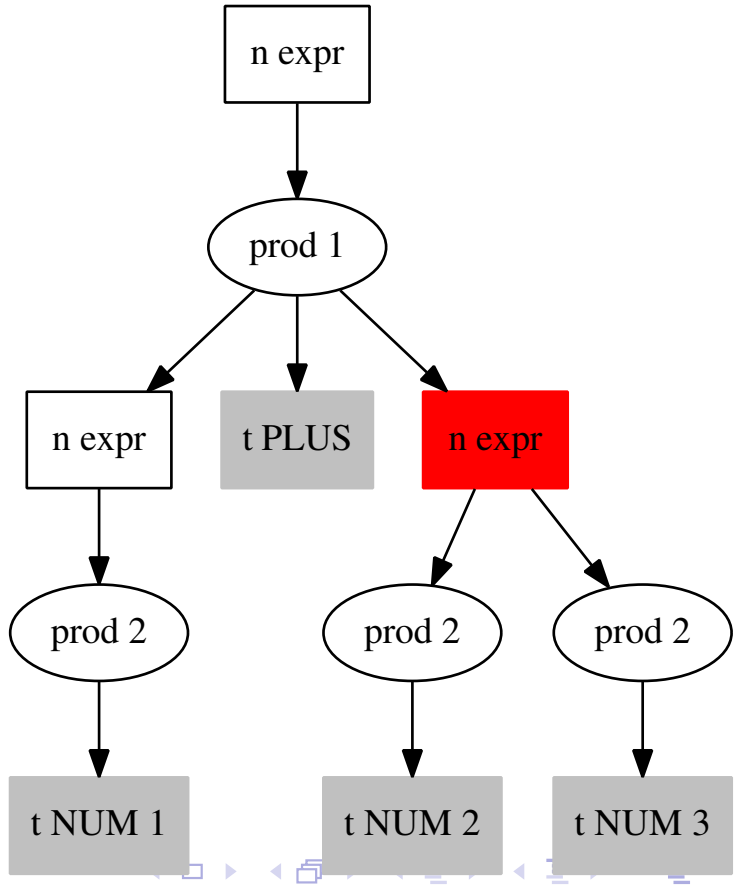



- Существуют проблемы, связанные с особенностями базового (GLR) алгоритма

# Вычисление семантики

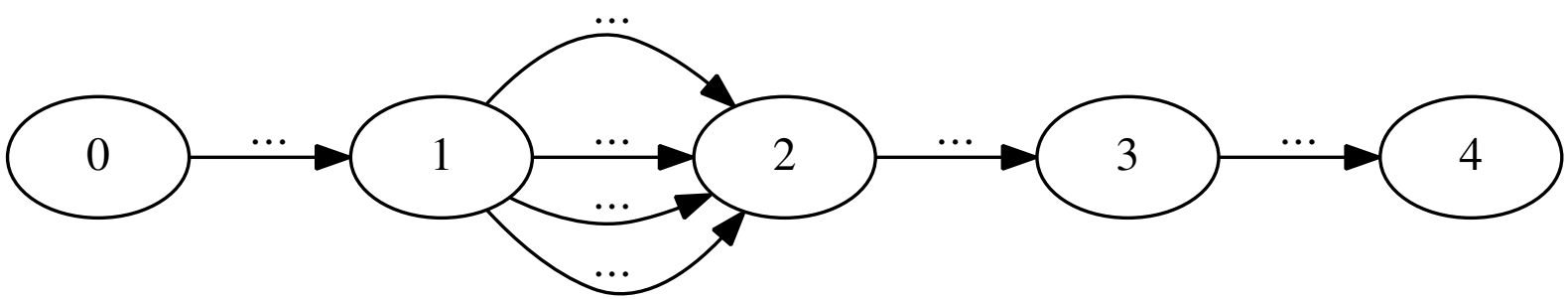
- Результат анализа – минимум одно дерево для пути в графе и весь лес разбора сжат в SPPF
- Что-то можно вычислить прямо на графе, но часто нужно извлекать деревья



# Вычисление семантики

- В худшем случае придётся перебирать все деревья




- Ленивая генерация деревьев

# Демонстрация

Demo

# Результаты

- Ядро
  - Генератор абстрактных лексических анализаторов
    - Привязка к исходному коду
  - Генератор абстрактных синтаксических анализаторов
    - Диагностика ошибок
    - Механизм вычисления семантики
  - Модульная архитектура для языковых расширений
- Плагин для ReSharper
  - Расширяемая архитектура, позволяющая легко поддержать любой встроенный язык
    - Внешний язык должен поддерживаться в ReSharper

# Область применения

- Поддержка встроенных языков в IDE
  - Интерактивная ("на лету")
  - "Офлайновая" проверка (ручной запуск)
- Поддержка, сопровождение кода со встроенными языками
- Автоматизированный реинжиниринг ПО, разработанного с применением встроенных языков

# Информация о проекте

- Контакты:
  - Григорьев Семён: Semen.Grigorev@jetbrains.com
  - Вербицкая Екатерина: kajigor@gmail.com
  - Мавчун Екатерина: emavchun@gmail.com
  - Иванов Андрей: ivanovandrew2004@gmail.com
  - Полубелова Марина: polubelovam@gmail.com
- Исходный код YaccConstructor:
  http://recursive-ascent.googlecode.com
- Google+ сообщество:
  https://plus.google.com/u/0/communities/102842370317111619055
- Сообщество GitHub: https://github.com/YaccConstructor