

Семинар "Наукоемкое  
программное обеспечение"  
PSI-2014  
24-27 июня, Санкт-Петербург



## Инструментальная поддержка встроенных языков в интегрированных средах разработки

**Автор:** Григорьев Семён, Вербицкая Екатерина,  
Иванов Андрей, Мавчун Екатерина,  
Полубелова Марина

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

# 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 и его нужно соответствующим образом поддерживать и обрабатывать
  - ▶ Ошибки в динамически формируемых выражениях обнаруживаются лишь во время выполнения
  - ▶ Поддержка в IDE
  - ▶ Рейнжиниринг ПО, разработанного с использованием встроенных языков
- Однако для стандартных инструментов это просто строки
  - ▶ Ошибки во время выполнения
  - ▶ Нет поддержки в IDE

# Актуальность

- Да, новый код так почти не пишут. Но:
  - ▶ Многое уже написано и оно требует поддержки, сопровождения
  - ▶ Альтернатив динамическому SQL пока мало
  - ▶ Есть кодогенераторы и они генерируют код в виде текста

# Предлагаемое решение

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

## Существующие решения

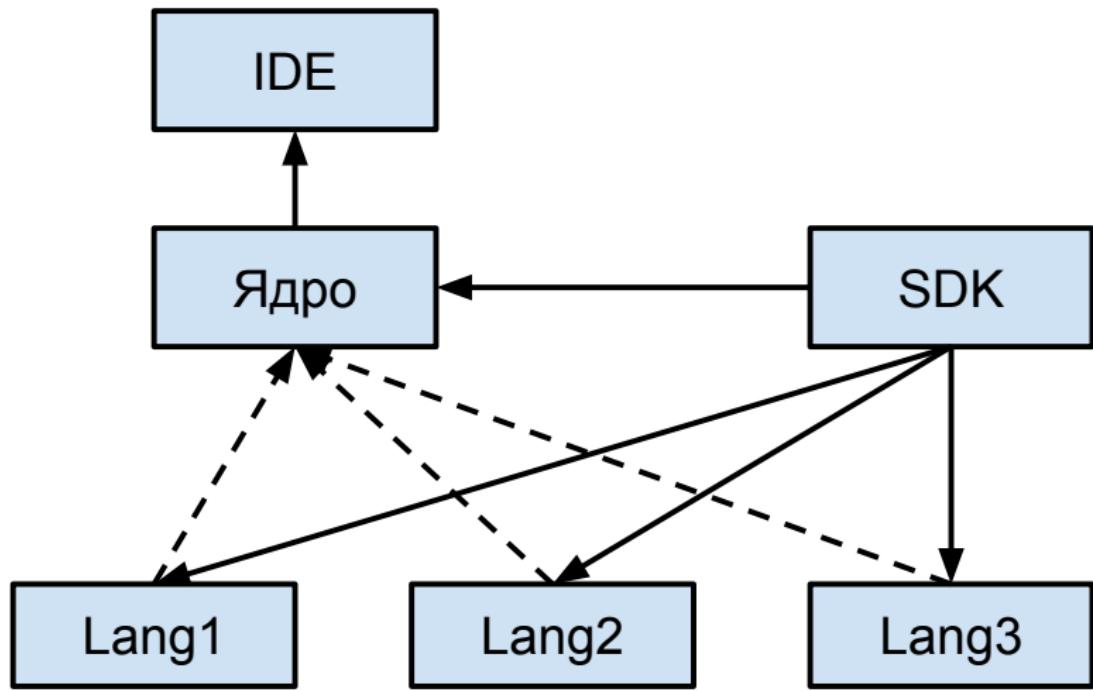
- Alvor – плагин для Eclipse для статической проверки встроенного в Java SQL
- Java String Analyzer – статический анализатор динамических выражений для Java
- PHP String Analyzer – статический анализатор динамических выражений для PHP
- PHPStorm – IDE для PHP с поддержкой HTML, CSS, JavaScript
- IntelliJLang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



# Языковые расширения

- SDK
  - ▶ Генератор абстрактных лексических анализаторов
  - ▶ Генератор абстрактных синтаксических анализаторов
  - ▶ Описание общих интерфейсов
  - ▶ ...
- Плагинная система на основе Mono.Addins
- Механизм разметки кода конечного пользователя

## Языковые расширения: пример

```
var tbl1 = "#tbl1"
```

```
var tbl2 = "tbl2"
```

```
[<InjectedLang("TSQL")>]
```

```
execute ("select x from " + if cond then tbl1 else tbl2)
```

```
[<InjectedLang("Calc")>]
```

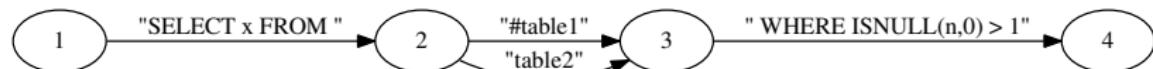
```
eval ("12 + 3 * (4 + 5)")
```

# 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
  - ▶ Граф
  - ▶ Регулярное выражение
- Выполнение лексического, синтаксического анализа над графом

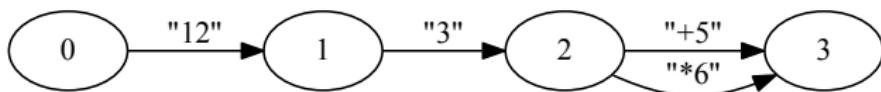
# Пример

- ```
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'}`
- Апроксимация:

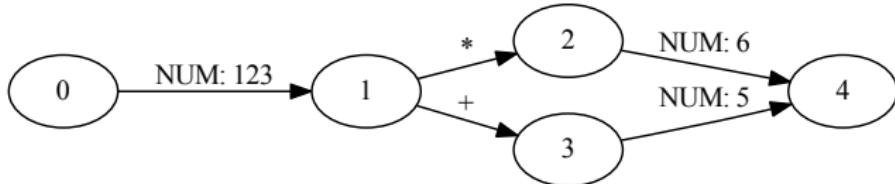


# Абстрактный лексический анализ

- Апроксиация (граф со строками на рёбрах) → граф с токенами на рёбрах
  - ▶ Привязка к литералу в исходном коде
  - ▶ Точная привязка внутри литерала
- Например:
  - ▶ Входной график

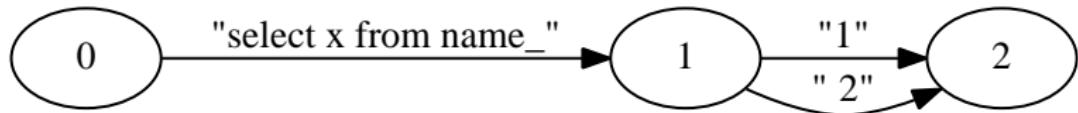


- ▶ Результат лексического анализа



# Абстрактный лексический анализ: рваные токены

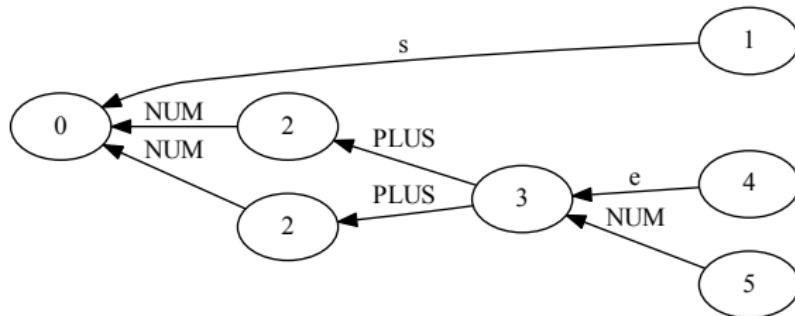
- Токены могут собираться из нескольких частей



- Нужно обрабатывать такие ситуации
- Нужно сохранять привязку каждой части

# Обобщённый синтаксический анализ

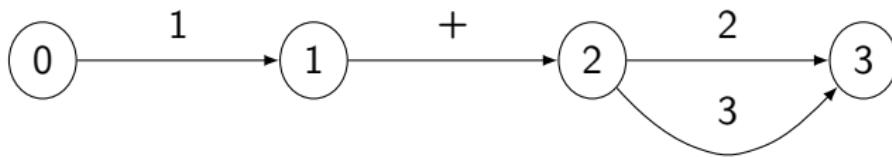
- Generalized LR parsing (GLR)
- Предназначен для работы с произвольными КС грамматиками
  - ▶ Shift-Reduce и Reduce-Reduce конфликты
- Использует организованный в виде графа стек (GSS)



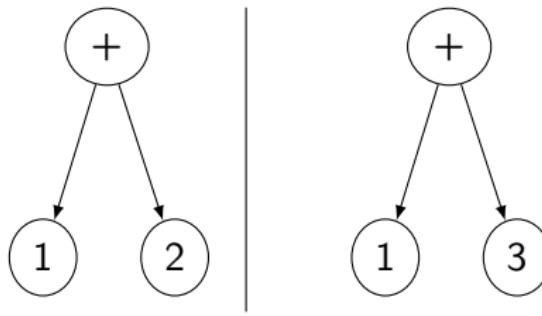
- Использует компактное представление леса вывода (SPPF)
  - ▶ Переиспользование общих узлов

# Абстрактный синтаксический анализ

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

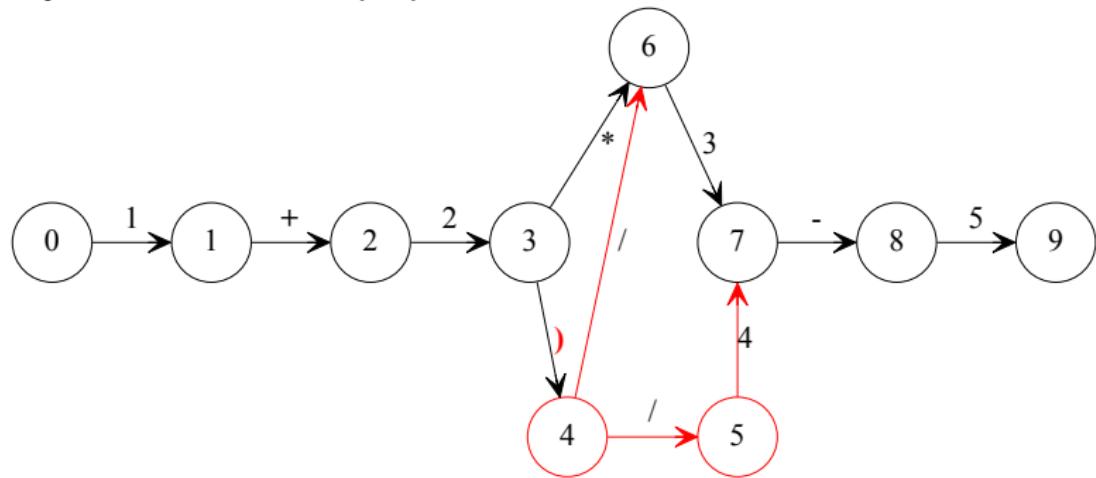


- Результат:



## 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

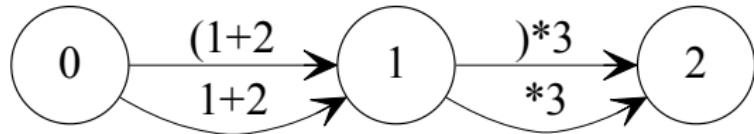
# Восстановление после ошибок

- Отдельный вопрос для исследований

- ▶ В больших выражениях хочется видеть все ошибки
- ▶ В сложных выражениях может быть много ложных ошибок
- ▶ Бывают принципиально некорректные выражения, которые при реальном выполнении не формируются
- ▶ Например

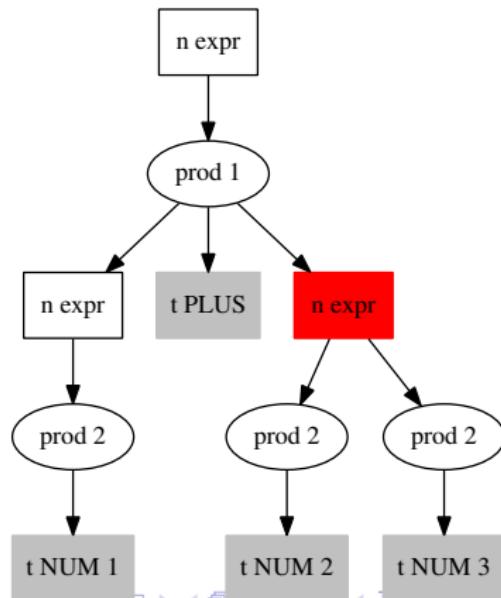
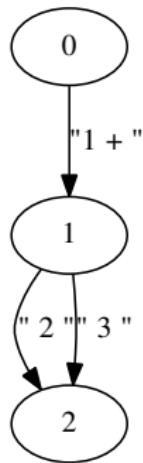
```
x = condition ? “(1+2” : “1+2”;  
y = condition ? “)*3” : “*3”;  
Program.Eval(x + y);
```

2 из 4 путей не могут быть получены, но в графе есть все



# Вычисление семантики

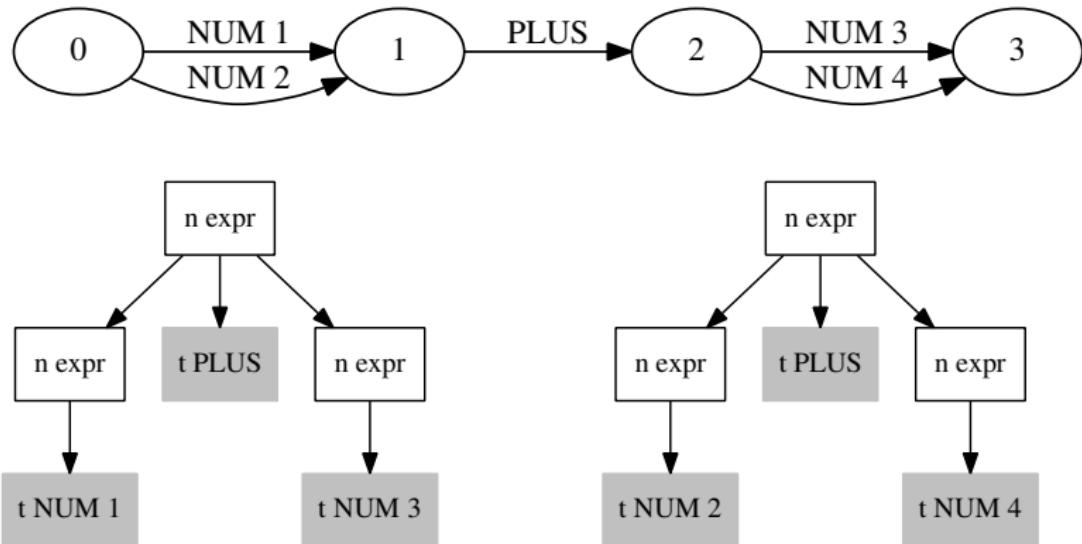
- Результат анализа – минимум одно дерево для пути в графе и весь лес разбора сжат в SPPF
- Что-то можно вычислить прямо на графике, но часто нужно извлекать деревья



# Вычисление семантики: пример

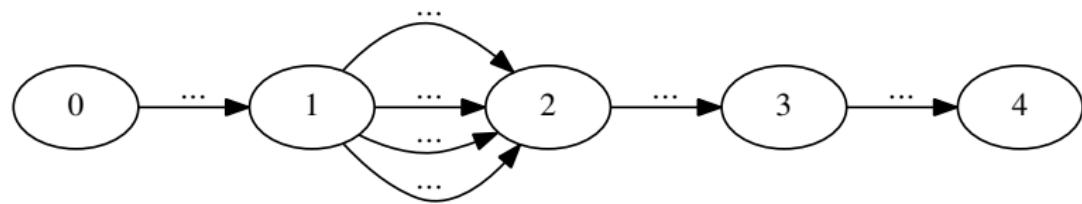
## Подсветка синтаксиса

- Достаточно покрыть все токены
- Можно возвращать не все деревья, а некоторое подмножество



# Вычисление семантики

- В худшем случае придётся перебирать все деревья



- Ленивая генерация деревьев

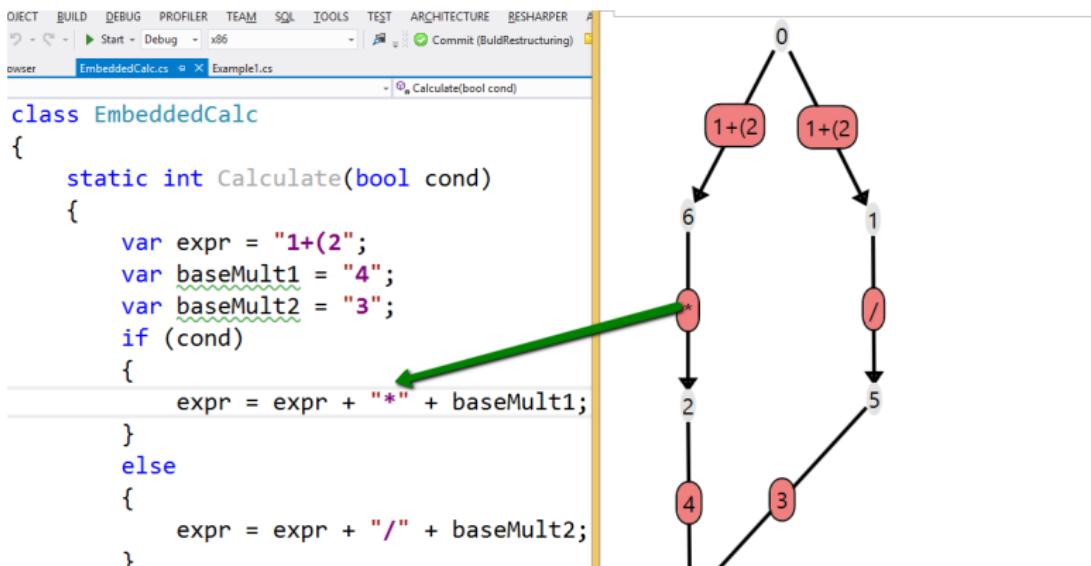
# Демонстрация

# Что дальше

- Большинство задач, применимых к обычному коду, применимо и ко встроенным языкам:
  - ▶ навигация
  - ▶ проверка типов
  - ▶ трансформации
  - ▶ ...

# Навигация по коду

- Визуальное представление динамического выражения в виде графа
  - ▶ Навигация в код
  - ▶ Навигация в график из кода
  - ▶ Информация об ошибках на графике: если ошибка на стыке строковых литералов, то трудно понять, где она реально



# Проверка типов

- Внутри выражения
  - ▶ `int x = eval ("12 * 'string' + 3 ")`
- Согласованность с внешним кодом
  - ▶ `int x = execute ("select 'string' from dual")`

# Трансформации

- Рефакторинг
- Переход с одногоialectа на другой
  - ▶ Для SQL особенно актуально
- Переход на новые технологии
  - ▶ Встроенный SQL → LINQ

Много вопросов

- Возможны ли нетривиальные трансформации?
  - ▶ Как обратно генерировать код?
- Как гарантировано корректность трансформаций?

# 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 поддерживать любой встроенный язык
    - ★ Внешний язык должен поддерживаться в ReSharper

# Реализация

- Открытый код
- Платформа .NET
- Основной язык – F#
- Проект YaccConstructor – платформа для исследований в области синтаксического анализа

# Область применения

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

# 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>