



Разработка средств реинжиниринга

Летняя школа 2012

Руководители: Григорьев Семён Вячеславович,
Либис Илья Романович

Санкт-Петербургский государственный университет
Математико-Механический факультет
Кафедра системного программирования

17 сентября 2012г.

Участники летней школы 2012:

- Байгильдин Ильнур
- Байков Артур
- Бакаева Алиса
- Иконникова Елена
- Шенбин Илья
- Щербаков Олег

Использование Clang для рефакторинга

Елена Иконникова

Clang и LLVM

- LLVM (Low Level Virtual Machine) — система анализа, трансформации и оптимизации программ ("compiler infrastructure"). В основе LLVM лежит промежуточное представление кода (intermediate representation, IR).
- Clang — компилятор для C-подобных языков. Изначально спроектирован для максимального сохранения информации в AST

"Связка": Clang — front-end, LLVM — back-end.

Создание плагина для Visual Studio, позволяющего использовать средства компилятора Clang при рефакторинге.

- Изучить компоненты, входящие в состав Clang
- Реализовать поиск функций и переменных по синтаксическому дереву.

- Реализован обход AST с поиском функций и переменных по их имени.
- При нахождении нужного имени возможна его замена на другое имя либо добавление комментария.

- RecursiveASTVisitor – обходит дерево
- Rewriter – позволяет трансформировать исходный код
- Узлы AST: объявления — Decl (наследники: FunctionDecl, TagDecl, ...), вызовы функций (CallExpr), операторы (BinaryOperator, UnaryOperator) и др.

- Название: YaccConstructor
- Сайт проекта: <http://recursive-ascent.googlecode.com>
- YaccConstructor – это модульный инструмент для разработки парсеров и трансляторов для платформы .NET. Реализован на F#. Основная область применения – реинжиниринг программного обеспечения.

Автоматическое распознавание диалектов

Алиса Бакаева
Ильнур Байгильдин

- Автоматическая обработка диалектов.
 - ▶ Распознавание. Определение диалекта языка программирования.
 - ▶ Оценка соответствия выбранному стилю программирования.
- Реализация автоматической обработки диалектов в рамках инструмента YaccConstructor.

Задача

- Изучить СΥК.
- Расширить СΥК механизмом отслеживания меток.

Реализация. Основа алгоритма.

Существует алгоритм на основе CYK для работы с PCFG.

- PCFG – вероятностная грамматика. Альтернативам приписываются вероятности.
- CYK + PCFG + функция пересчёта вероятностей – строится наиболее вероятный вывод
- Применяется в NLP

Реализация. Обработка меток.

Функция пересчёта вероятностей может быть обобщена.

- В нотации $F\# : curLbl : Option \langle 'lbl \rangle \rightarrow Option \langle 'lbl \rangle \rightarrow Option \langle 'lbl \rangle \rightarrow Option \langle 'lbl \rangle$
- Для простого определения диалектов: функция, проверяющая метки на равенство.
- Эту функцию может определять пользователь.
- В общем случае веса могут интерпретироваться произвольным образом.

X	Y, lbl1	X	X
Z, lbl1		T, lbl2	
S; conflict			

S: ZT
Y: @lbl1(a)
Z: (XY)
T: @lbl2(X X)

- Реализован алгоритм
 - ▶ определяет принадлежность диалекту
 - ▶ печатает координаты характерных конструкций
- Поставлен ряд экспериментов по определению диалектов
 - ▶ Однозначно определён
 - ▶ Конфликт
 - ▶ Не удалось определить

- Интеграция алгоритма СΥΚ, модифицированного, для работы с диалектами, с YaccConstructor
 - ▶ Расширение языка YARD возможностью использования меток в грамматике
 - ▶ Реализация преобразования входной грамматики в эквивалентную ей грамматику в нормальной форме Хомского

Пример:

```
s: X @name(a B c) S;
```

где @name - метка, а в скобках находится последовательность характерных терминалов и нетерминалов.

Преобразование грамматики

- Входная грамматика на языке YARD
 - ▶ Содержит EBNF, метаправила
- Необходимо преобразовать в CNF
 - ▶ Преобразование в BNF
 - ★ существующие преобразования
 - ★ поддержка меток
 - ▶ Удаление ϵ -правил
 - ▶ Удаление цепных правил
 - ▶ Преобразование в CNF

- Грамматика поддерживает метки
- Добавлен алгоритм для преобразования грамматики в нормальную форму Хомского, с поддержкой меток

Использование GPGPU для синтаксического анализа

Илья Шенбин

Эффективная параллельная реализация алгоритма синтаксического анализа на GPU.

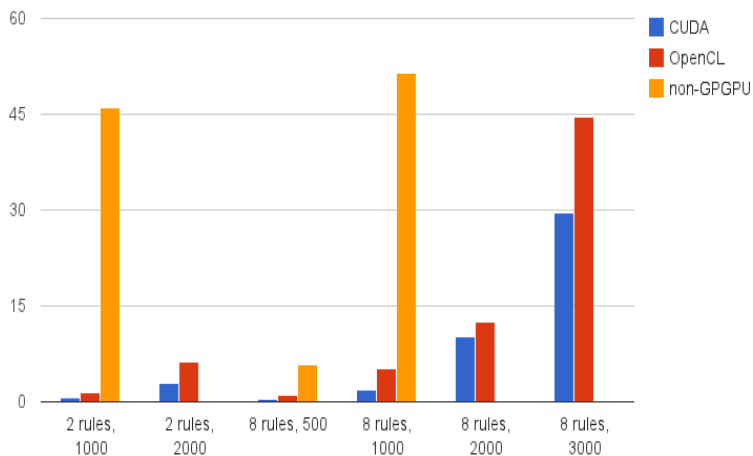
- Асимптотическая сложность парсеров КС-грамматик - $O(n^3)$, при этом размеры анализируемых строк огромны.
- Рост вычислительных мощностей процессоров за счёт увлечения числа ядер.

Актуальность для проекта

- Диалекты
- Быстрый синтаксический анализ

- Изучение технологии CUDA
- Реализация:
 - ▶ Реализация CYK на языке C
 - ▶ Распараллеливание алгоритма (по ячейкам в строке, по правилам...)
 - ▶ Модификация и оптимизация алгоритма с учётом особенностей CUDA (оптимизация памяти, распределения потоков...)

Результаты



Интеграция YARD с Visual Studio

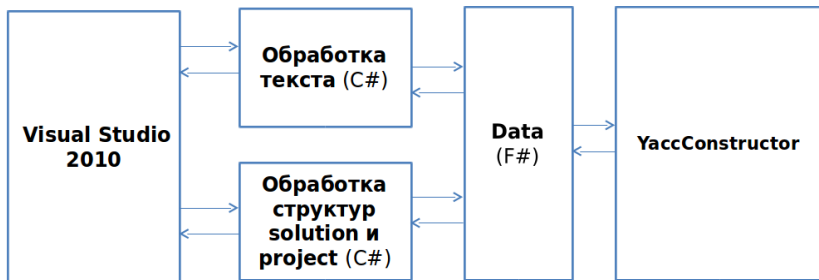
Олег Щербаков
Артур Байков

Реализация поддержки языка YARD в среде Visual Studio.

- Изучение F#;
- Изучение и применение возможностей VS SDK;
- Реализация:
 - ▶ перехода к определению;
 - ▶ подсветки вхождений выделенного нетерминала;
- Улучшение автодополнения.

Реализация

- Обработка текста из активного окна;
- Обработка структуры solution и project в VS
- Модуль Data для работы с парсером и лексером языка YARD, а так же хранение промежуточных результатов и структуры проекта.



Результаты

- Реализованы:

- ▶ переход к определению;
- ▶ подсветка вхождений выделенного нетерминала;

```
proc_body_stmt:
  select_stmt
  |
  | set_stmt
  | execute_stmt
  | sql_expr s
  | KW_DECLARE LOCALVAR KW_AS sql_datatype ( OP_EQ sql_expr)?
  | KW_RETURN sql_expr
  | KW_IF sql_expr KW_THEN ( (KW_BEGIN sql_expr* KW_END) | sql_expr)
  ;
```

- Улучшение автодополнения.

```
proc_body_stmt:
  select_stmt
  |
  | set_stmt
  | execute_stmt
  | sql_expr s
  | KW_DECLARE LOCALVAR KW_AS sql_datatype ( OP_EQ sql_expr)?
  | KW_RETURN sql_expr
  | KW_IF sql_expr KW_THEN ( (KW_BEGIN sql_expr* KW_END) | sql_expr)
  ;
(* http://msd
select_stmt:
(* list<<(KW
  query_exp
  (KW_INT
  (KW_INT
  start
query_expression:
  ( query_specification | (LPAREN query_expression RPAREN) )
  (
    (KW_UNION KW_ALL? | KW_EXCEPT | KW_INTERSECT) query_specification
    | LPAREN query_expression RPAREN
  )?
  ;
```

- Сайт проекта: <http://recursive-ascent.googlecode.com>
- Вопросы, пожелания, предложения:
 - ▶ Semen.Grigorev@lanit-tercom.com
 - ▶ Iliia.Libis@lanit-tercom.com