Разработка средств реинжиниринга

Летняя школа 2012

Руководители: Григорьев Семён Вячеславович,
Либис Илья Батькович

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

17 сентября 2012г.

Участники

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 infrastucture”). 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.

Цель

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

Задачи

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

Результаты

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

Детали реализации

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

YaccConstructor

- Название: YaccConctructor
- 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 распознание диалектов

Алиса Бакаева
Ильнур Байгильдин

Цель

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

Задача

- Изучить CYK.
- Расширить CYK механизмом отслеживания меток.

Реализация. Основа алгоритма.

Существует алгоритм на основе CYK для работы с PCFG.

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

Реализация. Обработка меток.

Функция перещёта вероятностей может быть обобщена.

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

Результаты

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

Задача

- Интеграция алгоритма CYK, 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

Метки

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

Результаты

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

Использование GPGPU для синтаксичаского анализа

Илья Шенбин

Цель

Эффективная параллельная реализация алгоритма синтаксического анализа на GPU.

Актуальность

- Асимптотическая сложность парсеров KC-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

Задача

- Изучение архитектуры 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

<диаграмма, сравнение с непараллельной версией, с другими алгоритмами>

Интеграция YARD с Visual Studio

Олег Щербаков
Артур Байков

Цель

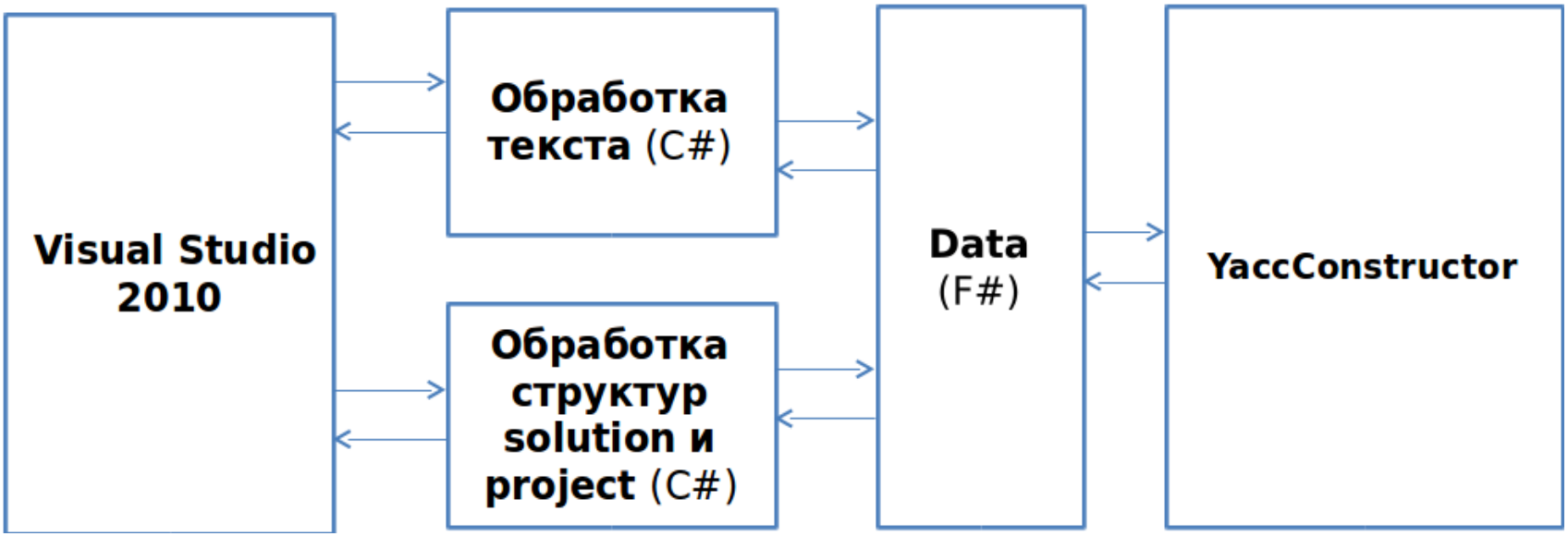
Реализация поддержки языка YARD в среде Visual Studio.

Задача

- 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_stmnt:
    select_stmnt
|   set_stmnt
|   execute_stmnt
|   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.

Заключение

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