Язык описания трансляций для средств реинжиниринга информационных систем

Авдюхин Дмитрий Алексеевич
Научный руководитель: ст. преп. Я.А. Кириленко
Рецензент: С.Д. Шкредов

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

11 июня 2013 г.

Реинжиниринг программного обеспечения

- Синтаксический анализ
- Генераторы парсеров
- Отсутствие подходящей документации языка
  - Синтаксические диаграммы
  - Неполнота и неоднозначность грамматики
  - Множество диалектов языка
    - ★ Ручные изменения
- Итеративная разработка грамматики

Требования к языку спецификации трансляций

- Возможность быстрого изменения грамматики
- Наличие средств переиспользования кода
- Наличие средств для работы с диалектами
- Быстрота разработки прототипа синтаксического анализатора

Большинство инструментов не удовлетворяет всем требованиям

Постановка задачи

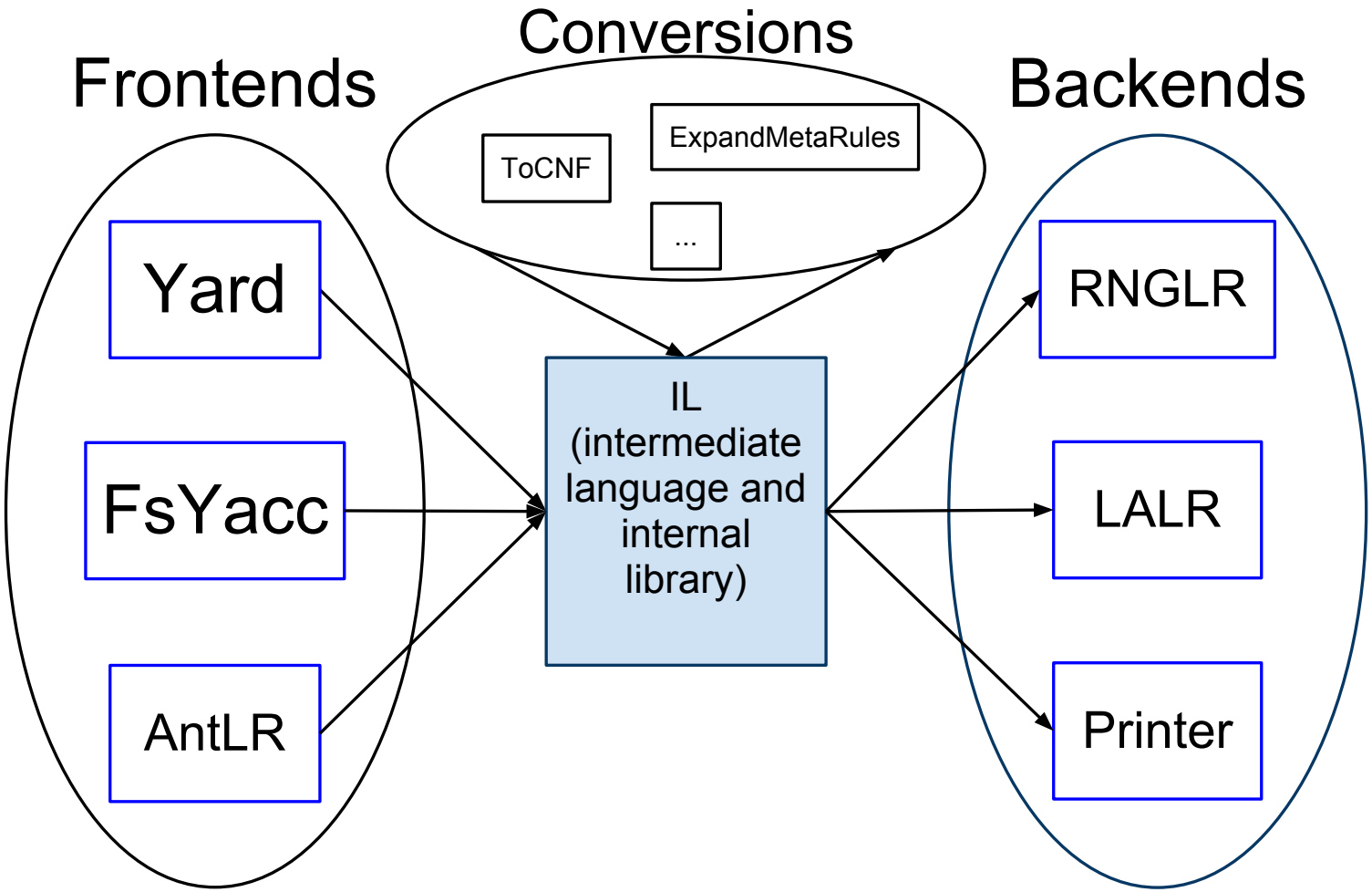
- Провести анализ существующих генераторов парсеров, оценить их средства описания трансляций
- Описать язык спецификации трансляций, сочетающий преимущества рассмотренных инструментов
- Реализовать этот язык
- Создать практически применимый генератор трансляторов на основе этого языка

Анализ инструментов

- Рассмотрены более 50 инструментов
- Нет языка, сочетающего все выявленные возможности
- Серьезные ограничения на конструкции
- Отсутствует поддержка диалектов
- Слабые алгоритмы разбора

YaccConstructor

Модульный инструмент для создания генераторов парсеров



Yard – язык описания трансляций

- Расширенная форма Бэкуса – Наура

```
a: b (c|d)* e
```

- Параметризованные правила

```
list<elem sep>: elem (sep elem)*
```

- Наследуемые атрибуты

```
rule<<arg>>: x=e {x * arg}
```

- Предикаты

```
rule: a=e b=e =>{a < b}=> {a+b}
```

Поддержка диалектов в Yard

- Условная компиляция

```
exec_literal:
  #if ms
    ('EXEC' | 'EXECUTE')
  #elif pl
    'EXECUTE' 'IMMEDIATE'
  #endif
    '(' LITERAL ')';
```

- Метки

```
s: x (@l1(y z)| @l2(a b)) g
```

Модульность

- Разбиение грамматики
- Явное указание зависимостей между модулями

```
module A
    public a: X
    b: Y

module B
    open A
    c: a // OK
    d: b // Ошибка
```

Опции генератора

```
FsYard.exe -i mssql.yrd -o MsParser.fs -translate false ^
     -module Yard.Examples.MSParser -token SourceText ^
     -infEpsPath epsilons -pos uint64 > log.txt

    options {
        translate = false
        token = SourceText
        pos = uint64
        module = "Yard.Examples.MSParser"
        infEpsPath = epsilons
    }
```

Типы терминалов

- NUMBER может хранить соотвествующее число
- Во многих случаях достаточно, чтобы все токены имели один и тот же тип
- Для некоторых терминалов тип указывается особо

```
tokens {
    | NUMBER of double
    | SELECT
    | _ of string
}
```

Разрешение неоднозначностей

- Естественный приоритет правил

```
expr:
    expr '+' expr
  | expr '*' expr
  | ...
```

- Указание терминалов, ожидаемых на данной позиции

```
if_stmt:
    'if' expr 'then' stmt /^'else'
  | 'if' expr 'then' stmt 'else' stmt
```

Приведение типов

```
sql_stmt:
    s=execute_stmt { s :> Statement }
  | s=set_stmt  { s :> Statement }
  | s=writetext_stmt { s :> Statement }
  | ...

sql_stmt: {:> Statement}
    execute_stmt
  | set_stmt
  | writetext_stmt
  | ...
```

Инструментальная поддержка

- Генератор GLR трансляторов
  - Был ранее реализован как модуль YaccConstructor
  - Большие расход памяти и время работы
  - Проведена доработка
    - В 3 раза меньше затрачиваемая память и время работы
    - Поддержаны новые конструкции языка
- Преобразования
  - Представление грамматики в виде списка правил
  - Раскрытие EBNF
  - Раскрытие параметризованных правил

Результаты

- Изучены существующие языки описания трансляций, выделены основные средства разработки грамматик
- Описан язык спецификации трансляций, предоставляющий выявленные возможности
- Доработан фронтенд языка Yard до состояния, соответствующего спецификации
- Получен генератор RNGLR трансляторов на основе этого языка за счет доработки одного из модулей YaccConstructor
- Доработаны преобразования грамматики, приводящие ее к виду, принимаемому генератором

https://code.google.com/p/recursive-ascent/