Форматирование программ

Принтер-комбинаторы и сопоставление с образцом

Подкопаев Антон, podkoav239@gmail.com

Лаборатория JetBrains

27 марта 2014

Контекст задачи

Языковые процессоры

- Синтаксический анализ
- Преобразование
- Представление результата
  - Код программы
  - ...

Контекст задачи

Языковые процессоры

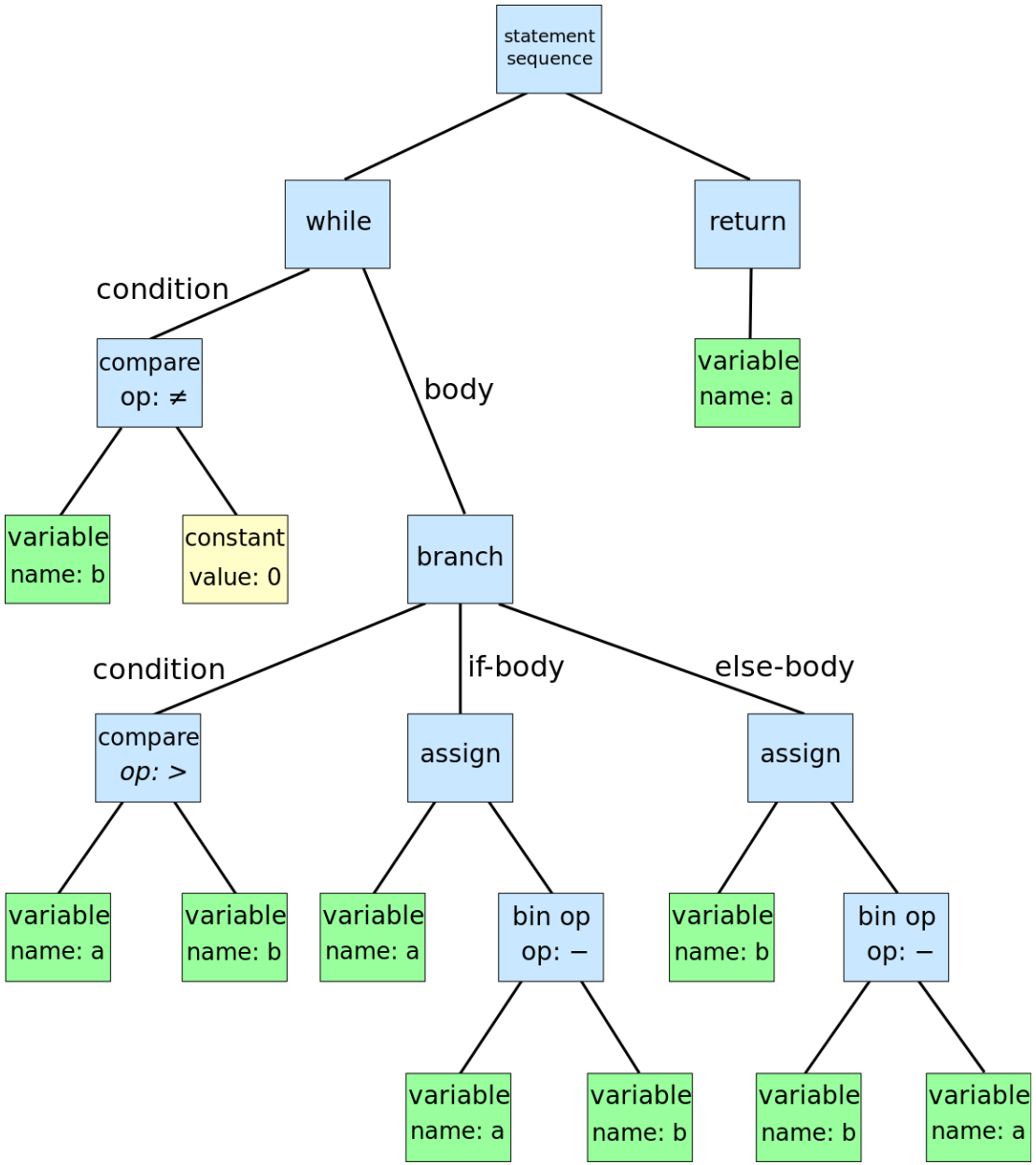
- Синтаксический анализ
- Преобразование
- Представление результата
  - Код программы
  - ...

Форматирование кода в IDE

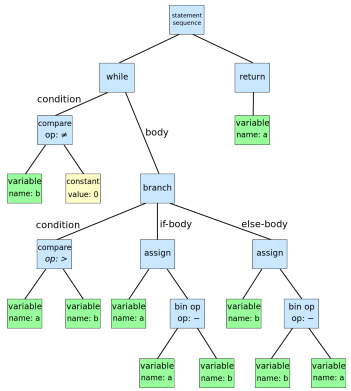
Принтеры (1)

- Pretty-printing
- Цель: форматированный вывод текста
- Вход: синтаксическое дерево
  - Абстрактное
  - Конкретное
- Выход: текст

Множественность представлений (1)



Множественность представлений (2)




```
while (b != 0) {
  if (a > b) {
    a = a - b;
  } else {
    b = b - a;
  }
}
return a;
```

```
while (b != 0) {
  if (a > b) { a = a - b; }
  else       { b = b - a; }
}
return a;
```

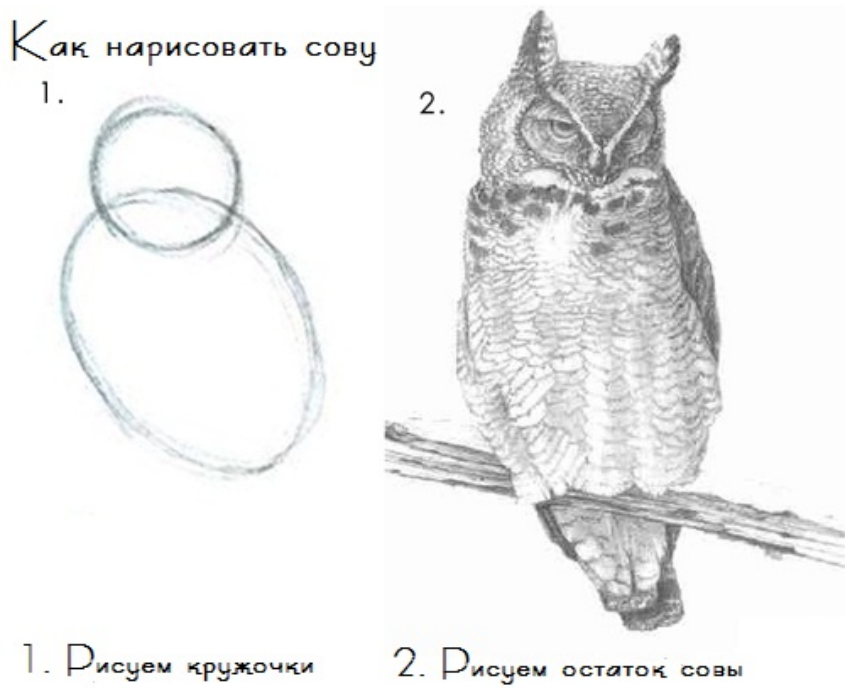
Принтеры (2)

- Ограничение - ширина вывода
- Оценка
  - Стиль (style guide)
  - Наглядность структуры
  - Обозримость (размер текста)
    - Оптимальность

Комбинаторы

— это функции высшего порядка, которые из одних функций строят другие

- Элементарные функции
- Переход от простого к сложному
- Пример: парсер-комбинаторы
  - symbol
  - »=
  - many
  - option
- Характерны для функционального программирования
- См. паттерн ”Компоновщик” и GUI-frameworks



Wadler Doc

```
data Doc = Doc `Beside` Doc
         | Nest Int Doc
         | Text String
         | Line -- '\n' or ' '
         | Doc :<|> Doc

x <> y    = x `Beside` y
nest i x  = Nest i x
text s    = Text s
line      = Line
group  x  = flatten x :<|> x

pretty :: Int -> Doc -> String
```

Wadler TTX

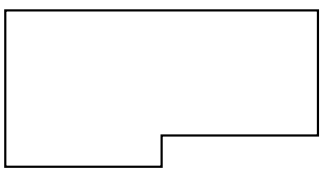
- Быстро работает (есть реализации за $O(n)$)
- Не всегда подходит
  - Язык Python

```
print 5
print 6
```

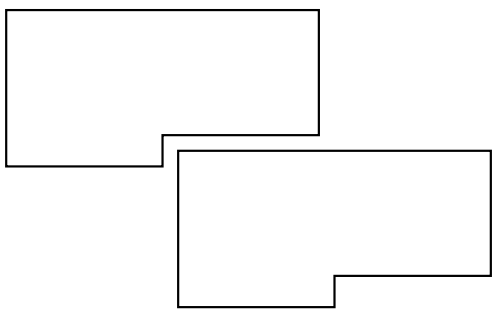
```
print 5 print 6
```

Azero, Swierstra Format

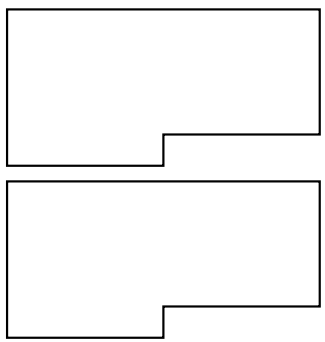
```
data Format = F { height         :: Int
                , totalWidth     :: Int
                , lastLineWidth  :: Int
                , txtstr         :: TextStruct
                }
```



Single



Beside



Above

Azero, Swierstra Doc

```
data Doc = Doc `Beside` Doc
         | Doc `Above`  Doc
         | Nest Int Doc
         | Text String
         | Doc `Choice` Doc

x <> y        = x `Beside` y
x $$ y        = x `Above`  y
nest i x      = Nest i x
text s        = Text s
x `choice` y = x `Choice` y
```

Azero, Swierstra pretty

```
type FormatSet = [Format]

dtf :: Int -> Doc -> FormatSet
dtf w (d1 `Beside` d2) = cross fmtBeside w d1 d2
dtf w (d1 `Above`  d2) = cross fmtAbove  w d1 d2

dtf w (Nest i d) = fmtListNest i $ dtf (w-i) d
dtf w (Text s) | length s < w = [strToFmt s]
               | otherwise    = []

dtf w (Choice d1 d2) = dtf w d1 ++ dtf w d2

pretty :: Int -> Doc -> String
pretty w d = best $ dtf w d
```

Azero, Swierstra TTX

- Работает в худшем случае за $O(2^n)$
- Выразительная способность достаточная для применения с шаблонами
- Выдает оптимальный результат
  - Самый низкий вариант из попадающих в заданную ширину

BURS

Bottom-Up Rewrite Systems

- Динамическое программирование на деревьях (дэгах)
- Правила

$$\begin{array}{lcll} N & : & \alpha & [c] \\ N & : & \alpha\,(K_1, \ldots, K_n) & [c] \end{array}$$

- Нахождение лучшей раскладки за линейное время от размера структуры

Мои комбинаторы

- Тот же самый интерфейс и результат, что и у Azero, Swierstra
- Факторизация по размерам Format

```
type Frame     = F totalWidth lastLineWidth
type FormatSet = HashMap Frame Format
```

- Ключ Map - нетерминал в BURS

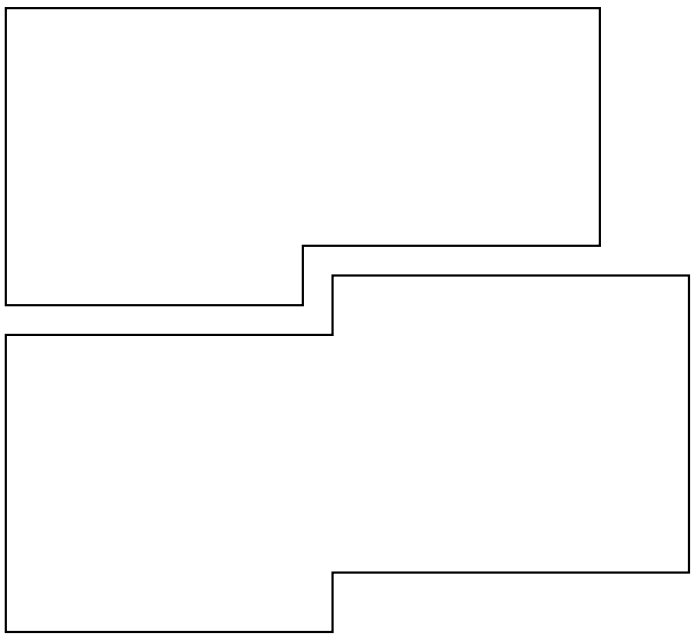
Алгоритмическая сложность

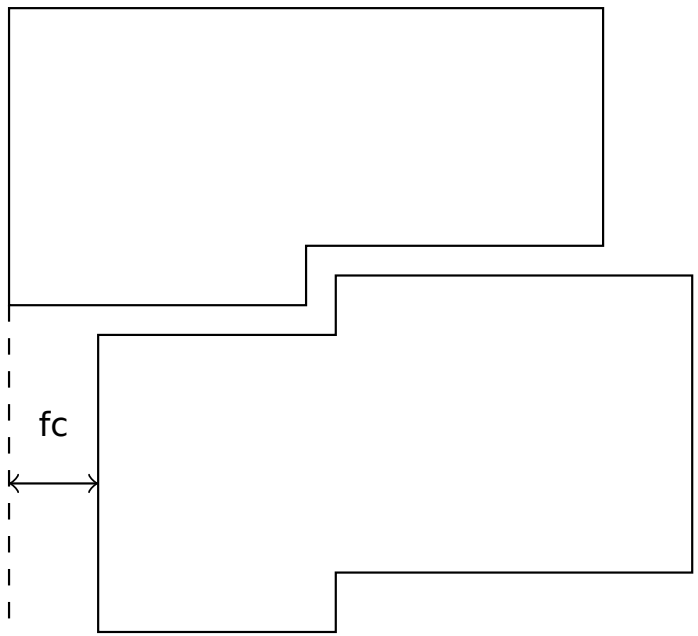
$O(w^{p*k} * n)$

- k - максимальная степень дерева (дэга)
- p - размерность ключа Map
- w - ширина вывода
- В случае комбинаторов Beside, Above, Choice
  - $k = 2, p = 2$
  - Сложность: $O(w^4 * n)$
- $w \in 100 \ldots 200$
- Скоро сложность возрастет до $O(w^6 * n)$

Расширение модели комбинаторов (1)

Комбинатор Fill





fc - Fill Constant

Расширение модели комбинаторов (2)

Факторизация

- Было : width, lastLineWidth
- Стало: width, lastLineWidth, firstLineWidth

Сложность возрасла до $O(w^6 * n)$

В таком виде абсолютно не применимо

Добавление в FormatSet

Вводим отношение частичного порядка на Format

Эвристика:

- Добавить только в случае, если нет варианта лучше
- Убрать варианты, которые хуже добавляемого

На данный момент самое критичное по производительности место

Получение образцов

Реализация

- Форматтер-плагин для IDEA
- Написан на Kotlin

Нерешенные проблемы

- Обработка комментариев
  - Возможно, шаблоны и для них, т.к. структура известна (JavaDoc, Doxygen)
- Списки (структуры с плавающим числом поддеревьев)
  - Сейчас - фиксированное представление
  - Шаблонный переход от одного элемента к другому
- Время работы
  - сейчас - 4с
  - нужно - 0.5с

Направления развития

- Абстрагирование от языка
- Проверка репозитория шаблонов на полноту
- Задание порядка поддеревьев
  - Поля, методы, подклассы
- Фильтрация ’плохих’ шаблонов
- Совместное форматирование поддеревьев