Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

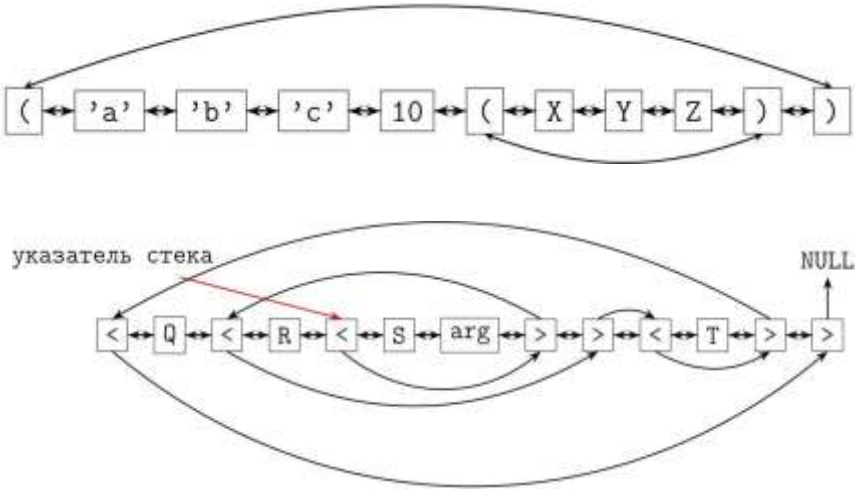
Реализация Рефала-5 с древесным представлением строк

Автор: Киселев К.А.

Списковое представление

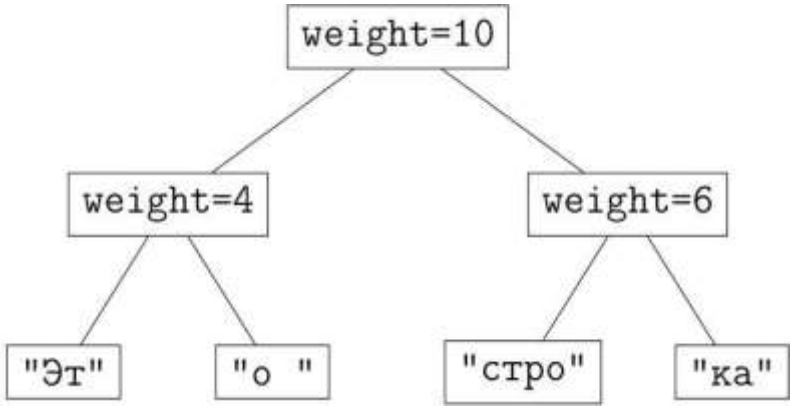
Каждый узел списка содержит:

- Указатели на предыдущий и следующий элементы
- Данные:
  - Символ
  - Структурную скобку
  - Угловую скобку



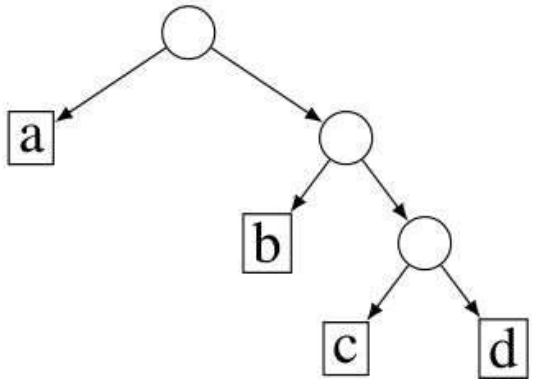
Структура данных Rope

- Бинарное дерево, листьями которого являются массивы символов
- Основные операции
  - Конкатенация
  - Получение символа по индексу
  - Разбиение по индексу
  - Операции взятия подстроки, вставки символа, удаление подстроки редуцируются к комбинации разбиения и конкатенации



Балансировка Rope

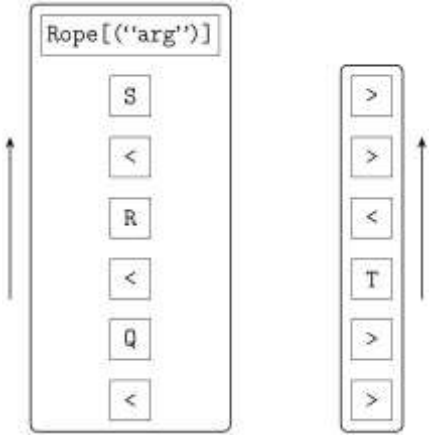
- Для сохранения эффективности операций с Rope необходимо поддерживать баланс дерева
- Можно выделить следующие критерии балансировки:
  - На основе чисел Фибоначчи. Сложность: O(n), где n - количество листьев
  - На основе АВЛ деревьев: Сложность: O(log(n)), где n - количество листьев



Представление поля зрения

- Поле зрения можно представлять с помощью двух стеков
- Главный цикл работает до тех пор, пока второй стек не пуст
- Изначально в правом стеке расположен вызов функции Go

Содержание поля зрения:
<Q <R <S (“arg”) >> <T>>



Синтаксический анализ

- Для реализации синтаксического анализа используется Tree-Sitter
- На основе грамматики описанной в виде JavaScript модуля генерируется код анализатора
- Анализатор подключается в виде отдельного пакета

```
identifier ::= letter (letter | digit | '-' | '_')*;
macrodigit ::= [0-9]+;
letter ::= [A-Za-z];
variable ::= type '.' index;
type ::= 's' | 't' | 'e';
index ::= identifier | macrodigit;
f-call ::= '<' identifier expression '>';
expression      ::= (expression-part)*;
expression-part ::= identifier
              | f-call
              | macrodigit
              | string
              | ( expression );
pattern      ::= (pattern-part)*;
pattern-part ::= identifier
              | macrodigit
              | string
              | ( pattern );
program ::= f-definition
           | f-definition program
           | f-definition ';' program
           | external-decl ';' program
           | program external-decl ';';
f-definition ::= ('$ENTRY')? identifier '{' block '}';
external-decl ::= ('$EXTERNAL' |  '$EXTERN' | '$EXTRN') f-name-list
f-name-list ::= identifier (',' identifier)*;
block ::= (sentence ';')* ( sentence )? ;
sentence ::= pattern (',' condition )* '=' expression
           | pattern (',' condition )* ',' block-ending;
condition ::= expression ':' pattern;
block-ending ::= expression ':' '{' block '}';
comment ::= line-comment | slashed-comment;
line-comment ::= '^*' [^\n]*;
slashed-comment ::= '/*' .* '*/';
```

Генерация кода

- Каждая функция Рефала переводится в функцию языка Golang
- Каждое сопоставление компилируется в цикл for
- Условия и блоки преобразуются в процессе построения AST дерева

```
type R5FunctionPtr func(l, r int, arg Rope, rhsStack *[]ViewFieldNode)

type R5Function struct {
        Name  string
        Entry bool
        Ptr   R5FunctionPtr
}

 for i := 0; i < 1; i++ {
  var p []int = make([]int, k)
        p[0] = l
        p[1] = r
        // Элементарное сопоставление 1
        // ...
        // Элементарное сопоставление n
        // Построение результата
        return
}
```

Стандартная библиотека

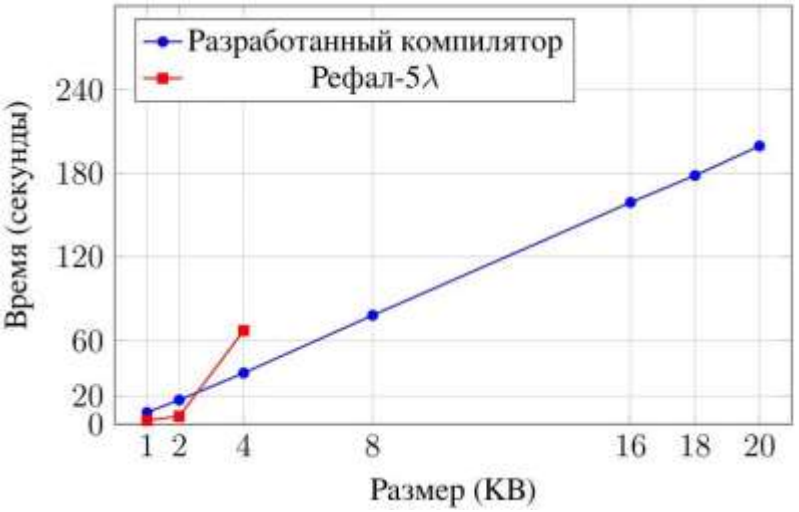
- Метафункции
  - Реализована метафункция Mu, которая генерируется для каждого исходного файла отдельно
- Арифметические функции:
  - Реализована длинная арифметика с использованием math/big
- Функции по работе с вводом/выводом:
  - Реализованы функции по работе с файлами/потоками ввода/вывода
- Функции преобразования типов:
  - Реализованы функции для перевода данных в разные типы

Сравнение со списковым представлением

```
Test {
  (0) e.Other, <Lenw e.Other> : {
    s.Len e.X = <Prout s.Len>;
  };
  (s.N) s.X e.Other = <Test (<Sub s.N 1>) e.Other e.Other e.Other>;
}

$ENTRY GO {
  e.x = <Open 'r' 1 <Arg 1>> <Test (10) <Get 1>>;
}
```

Сравнение со списковым представлением



Размер входных данных (КБ)	Рефал-5 (с древесным представлением строк, время выполнения, с)	Рефал-5λ (время выполнения, с)
1	8.33	2.60
2	17.25	5.52
4	36.60	67.05
8	78.01	-
16	159.01	-
18	178.36	-

Размер входных данных(КБ)	Рефал-5 (с древесным представлением строк), используемая память, МБ)	Рефал-5λ, используемая память, МБ)
1	3.08	1848.87
2	3.20	3695.20
4	3.82	5792.09
8	4.39	-
16	5.82	-
18	6.07	-
20	6.41	-

Размер входных данных (КБ)	Рефал-5 (с древесным представлением строк) время выполнения, с)	Рефал-5λ, время выполнения, с)
1	0.20	4.31
2	0.20	6.86
4	0.20	31.73
8	0.20	-
16	0.20	-
18	0.20	-
20	0.20	-