

1 Теоретическая основа

Для описания введённых структур данных используется *формальная* БНФ-грамматика. Все литералы имеют один из трёх префиксов: `s.`, `t.` или `e.`, соответствующих *символам*, *термам* и *объектным выражениям* в языке Рефал.

Помимо явных символьных слов типа `AreEqual`, `Var` и т.п. используются следующие нетерминалы:

- `s.NUMBER` — любая макроцифра;
- `s.CHAR` — любой символ;
- `s.WORD` — любое (символьное) слово.
- `e.ANY` — произвольное объектное выражение.

Символы `*` и `+` означают повторение литерала ноль и более и один и более раз соответственно.

1.1 Составляющие уравнения

Рассматриваются уравнения в словах с алфавитом переменных Ξ и алфавитом констант Σ . Применение алгоритма Jez'a предполагает хранение дополнительной информации об уравнении: какие константы являются результатом сжатия блоков, какие переменные не могут быть пусты и т.д. Поэтому *уравнение* представляется структурой данных `t.Eq`:

```
t.Eq ::= ((AreEqual (t.Elem*) (t.Elem*)) (t.Constr*) (t.Cond*)),
```

где `(t.Elem*)` есть левая и правая части уравнения, `(t.Constr*)` — ограничения на переменные, а `(t.Cond*)` содержит условия на константы.

Элемент `t.Elem` обобщённо представляет *константу* или *переменную*, соответствующие структурам данных `t.Const` и `t.Var`:

```
t.Elem ::= t.Var | t.Elem,  
t.Var  ::= (Var s.CHAR),  
t.Const ::= (s.CHAR s.NUMBER).
```

1.2 Ограничения на переменные

Ограничение на переменную представляется структурой `t.Constr` и описывается в конъюнктивной нормальной форме. Литералами дизъюнкций являются *рестрикции* (отрицательные условия на переменные) типа `t.Restr`, подразделяющиеся на *краевые* `t.BoundRestr` и рестрикции *на пустоту* `t.EmptyRestr`:

```
t.Restr ::= t.BoundRestr | t.EmptyRestr.
```

Краевые рестрикции бывают *префиксными* `t.PrefixRestr` и *суффиксными* `t.SuffixRestr`. Они указывают, на какие константы не может начинаться (кончатся) данная переменная.

```
t.BoundRestr ::= t.PrefixRestr | t.SuffixRestr,  
t.PrefixRestr ::= (not t.Const starts t.Var),  
t.SuffixRestr ::= (not t.Const ends t.Var).
```

Рестрикция на пустоту `t.EmptyRestr` сообщает, как следует, о невозможности обращения данной переменной в пустое слово ε :

```
t.EmptyRestr ::= (not empty t.Var).
```

Ограничения на переменные `t.Constr` могут содержать одну или две рестрикции, в соответствии с чем называются *тривиальными* `t.TrivialConstr` и *нетривиальными* `t.NonTrivialConstr`. В программе используется только четыре вида ограничений:

```
t.Constr ::= t.TrivialConstr | t.NonTrivialConstr  
t.TrivialConstr ::= (OR t.Restr),  
t.NonTrivialConstr ::= (OR t.SuffixRestr t.PrefixRestr).
```

Говоря далее ограничения мы, как правило, будем подразумевать именно тривиальные, если не оговорено противное, а также будем экстраполировать тип рестрикции на ограничение её содержащее: префиксное ограничение, ограничение на пустоту и т.д.

1.3 Условия на константы

Условие на константу `t.Cond` содержит сжимаемые блоки и соответствующую этому сжатию константу. Вообще, *блок* `t.Block` обозначает степень константы и представляется в виде

$t.\text{Block} ::= (t.\text{Const } t.\text{Exp}^* (\text{const } s.\text{NUMBER})),$

где $t.\text{Const}$ — сжимаемая константа, $(\text{const } s.\text{NUMBER})$ — обязательный константный показатель, а $t.\text{Exp} ::= (s.\text{WORD } s.\text{NUMBER})$ — переменный показатель степени. Таким образом, условие на константу представляется в виде

$t.\text{Cond} ::= (t.\text{Const is } t.\text{Block}^+)$

В программе используется только два типа условий: *парное условие* $t.\text{PairCond}$ и *условие на блок* $t.\text{BlockCond}$:

$t.\text{PairCond} ::= (t.\text{Const is } (t.\text{Const } (\text{const } 1)) (t.\text{Const } (\text{const } 1))),$
 $t.\text{BlockCond} ::= (t.\text{Const is } t.\text{Block}).$

1.4 Краевые элементы

Мы хотим применять *Pair*- и *Block*-сжатие не только к тривиальным константам, но и тем, что уже являются результатом сжатия в блоки. Для корректной обработки ограничений необходимо аккуратно отслеживать, на какие константы не может начинаться (кончатся) та или иная переменная.

Пусть $\alpha, \gamma \in \Sigma$. Скажем, что $\alpha \in \text{First}(\gamma)$, если существует цепочка условий на константы такая, что

$$\gamma = \beta_1^{i_1} B_1, \beta_1 = \beta_2^{i_2} B_2, \dots, \beta_{n-1} = \alpha^{i_n} B_n,$$

где $\beta_i \in \Sigma, i = 1, 2, \dots, n-1$, и $B_j \subset \Sigma, j = 1, 2, \dots, n$. Симметрично определяется множество *Last*-элементов константы.

Иногда удобно использовать обобщение введённых выше понятий: будем говорить, что константа α является **краевым** элементом константы γ , если $\alpha \in \text{First}(\gamma)$ или $\alpha \in \text{Last}(\gamma)$. Обозначение: $\alpha \in \text{Bound}(\gamma)$.

1.5 Слабые и сильные ограничения

В соответствие с введённым выше понятием краевого элемента, для данной переменной X и константы γ мы можем ввести *иерархические отношения* на множествах префиксных и суффиксных ограничений X с константами $\text{First}(\gamma)$ и $\text{Last}(\gamma)$ соответственно.

Пусть даны два таких ограничения одного типа с константами α и β . Если $\alpha \in \text{Bound}(\beta)$, то α -ограничение является **более сильным**. В то же время β -ограничение есть **более слабое** по сравнению с первым.

Имеет смысл хранить в уравнении лишь самые сильные ограничения.

1.6 Избыточные рестрикции и условия

В результате некоторых действий какие-то рестрикции могут стать неактуальными, вследствие чего их желательно удалять из уравнения.

Скажем, что рестрикция **избыточная**, если участвующей в ней константы нет в уравнении, а также в правых частях условий на константы.

Как и рестрикция, условие (на константу) также может стать **избыточным**. Это происходит в двух случаях:

- константа в левой части условия не участвует в уравнении и не является *Bound*-элементом какой-либо другой константы;
- условие имеет парный тип, обеих констант его правой части нет в уравнении и они не являются левыми частями каких-либо других условий.

1.7 Нормальное уравнение

Скажем, что уравнение **несократимо**, если его левая и правая части не имеют совпадающих префиксов и суффиксов.

Будем называть уравнение **нормальным**, если оно несократимо, не содержит слабых (краевых) ограничений и избыточных рестрикций и условий. Также потребуем, чтобы в нормальном уравнении все ограничения были отсортированы, равно как и показатели степеней `t.Exp` условий на блок.

Нормальные уравнения представляют особый интерес для нас. Именно такие уравнения возвращают и ожидают получить на вход функции `Pick`, `SubstIndex`, `PairComp` и `BlockComp`.