1 Теоретическая основа

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 в Рефале уравнение представляется структурой данных `t.Eq`:

```
t.Eq ::= ((AreEqual (e.LHS) (e.RHS)) (t.Constr*) (t.Cond*)),
```

где `e.LHS` и `e.RHS` содержат левую и правую части уравнения, `t.Constr` представляет ограничения на переменные, а `t.Cond` — условия на константы.

Выражения `e.LHS` и `e.RHS` состоят из произвольного числа *переменных* и *констант*, представляющихся структурами `t.Var` и `t.Const` соответственно. Нередко константы и переменные обобщённо называются *элементами* типа `t.Elem`:

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

1.2 Ограничения на переменные

Ограничения на переменные представляются структурой `t.Constr` и описываются 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 (кончаться) 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 соотвествующую этому сжатию константу. Вообще, *блок* `t.Block` представляется в виде

```
t.Block ::= (t.Const t.Exp* (const s.NUMBER)),
```

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

```
t.Cond ::= (t.Const is t.Block+)
```

В программе используется только два типа условий: *парное условие* `t.PairCond` и *условие на блок* `t.BlockCond`:

```
t.PairCond ::= (t.Const is (t.Const (const 1)) (t.Const (const 1))),
t.BlockCond ::= (t.Const is t.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 на переменные необходимо аккуратно отслеживать, на какие константы не может начинаться (кончаться) та или иная переменная.

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

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

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

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

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 $First(\gamma)$ и $Last(\gamma)$ соответственно.

Пусть даны два таких ограничения одного типа с константами α и β. Если $\alpha \in Bound(\beta)$, то α-ограничение (то, что с константой α) 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`.