Распознавание группового экранирования образцов в Рефале-5λ

Яушев Камиль

МГТУ имени Н. Э. Баумана

Совместное Совещание по языку Рефал

МГТУ им. Н.Э. Баумана и ИПС им. А.К. Айламазяна РАН

17 июня 2023 г.

Постановка задачи

```
Screening {
  s.Head e.Tail = False;
  'A' e.Tail = True;
}
```

```
MultiScreening {
    e.x (s.x 'A' e.y) e.z          = 1;
    e.x t.z ( e.y 'A' e.z) e.w     = 2;
    (( ) 'A' e.y) e.z              = 3;
    e.x ((t.q e.w) 'A' e.y) e.z    = 4;
    ('A' e.y) e.z                  = 5;
    e.x ( e.y t.q1 t.q2 'A' e.z) e.w = 6;
    e.x (e.y 'A' e.z) e.w          = 7;
}
```

- Экранирующие предложения такого вида могут появиться в программе как артефакты предыдущих версий, либо быть написаны по ошибке. Причем, зачастую такого рода артефакты или ошибки не являются очевидными.

Рефал-5λ: особенности языка

- Точное надмножество Рефала-5
- Функции высших порядков
- Абстрактные типы данных — «запечатанные скобки»
- Синтаксический сахар

Абстрактные типы данных

```
$ENUM SymTable;

$ENTRY SymTable-Create {
  = [SymTable];
}

$ENTRY SymTable-Lookup {
  [SymTable e.Names-B ((e.Name) e.Value) e.Names-E] e.Name
    = Success e.Value;
  [SymTable e.Names] e.Name = Fails;
}

$ENTRY SymTable-Update {
  [SymTable e.Names-B ((e.Name) e.OldValue) e.Names-E]
  (e.Name) e.NewValue
    = [SymTable e.Names-B ((e.Name) e.NewValue) e.Names-E];

  [SymTable e.Names] (e.Name) e.Value
    = [SymTable e.Names ((e.Name) e.Value)];
}
```

Переменные в Рефале-5λ

- s-переменные: символ
- t-переменные: терм – символ, ‘(‘ … ‘)’ и ‘[‘ … ‘]‘
- e-переменные: 0 и более термов

  - Обозначения:

```
s.X
t.Y
e.Z
```

Функции высших порядков в Рефале-5λ

```
*$FROM LibraryEx
$EXTERN Map;

PrintEachLine {
  (e.Line) = <Prout e.Line>;
}

$ENTRY PrintLines-1 {
  e.Lines = <Map &PrintEachLine e.Lines>;
}

$ENTRY PrintLines-2 {
  e.Lines = <Map { (e.Line) = <Prout e.Line>; } e.Lines>;
}
```

Синтаксический сахар: блоки

Блок можно приписать к любому результатному выражению (в том числе, в условии)

```
Func {
  некоторый образец
    , условие
    : { ...блок 1... }
    : { ...блок 2... }
    : образец условия
    = результатное выражение;
}
```

Синтаксический сахар: блоки

Блок выражается через вложенные функции:

Result : { …A… } : { …B… } : { …C… }

является эквивалентом для

<{ …C… } <{ …B… } <{ …A… } Result>>>

Синтаксический сахар: присваивания

Присваивание — «безоткатное условие»:

```
Length {
  e.Expr
    = <Lenw e.Expr> : s.Len e.Expr1
    = s.Len
}
```

Присваивание выражается через блок:

```
Length {
  e.Expr
    = <Lenw e.Expr>
    : { s.Len e.Expr1 = s.Len }
}
```

Синтаксический сахар: сокрытие переменных

```
/* Procedure → Header Declarations Body. */
ParseProcedure {
  e.Tokens
    = <ParseHeader e.Tokens>
    : (e.ProcedureName) (e.Parameters) (e.HeaderErrors) e.Tokens^

    = <ParseDeclarations e.Tokens>
    : (e.Declarations) (e.DeclErrors) e.Tokens^

    = <ParseBody e.Tokens> : (e.Body) (e.BodyErrors) e.Tokens^

    = ((e.ProcedureName) (e.Parameters) (e.Declarations) e.Body)
      (e.HeaderErrors e.DeclErrors e.BodyErrors)
      e.Tokens;
}
```

Экранирование предложений

Предложение называется *экранируемым*, если множество значений, описываемым его левой частью, является подмножеством множества, описываемого левой частью одного из предшествующих образцов.

```
FirstA {
  s.Head e.Tail = False;
  'A' e.Tail = True;
}
```

Якорная t-переменная

Якорной называется t-переменная t.x в образце P, если:

• либо t.x имеет кратность ≥ 2;

• либо в P существует подслово A, не содержащее e-переменных, такое, что A = B t.x C, причем и B, и C содержат хотя бы один символ или t-переменную, имеющую кратность ≥ 2.

В противном случае назовем t.x плавающей.

Нормальная форма образца

Считаем, что линейный образец P в *нормальной форме*, если

• P не содержит двух и более идущих подряд e-переменных;

• каждая плавающая t-переменная в P предшествует

e-переменной и следует за e-переменной.

Нормальная форма образца

t.y1 t.y2 e.x1 *t.y3 t.y4* **t.y2** e.x2 t.y5

есть

t.y1 t.y2 e.x1 *t.y3* e.x3 *t.y4* e.x4 **t.y2** e.x2 *t.y5* e.x5

Алгоритм обобщенного сопоставления с образцом

- E – образец-шаблон, P – произвольный образец
- Есть прямое решение обобщенного уравнения E : P

  ⇒ E экранирует P
- Ограничение на экранирующий образец:

  отсутствие кратных t- и e-переменных

Алгоритм, основанный на теории языков образцов

- Идея: рассматриваются языки образцов — множества строк, описываемые образцами
- Язык образца P вкладывается в язык образца E

  ⇒ E экранирует P
- Ограничение на экранирующий образец:

  отсутствие кратных e-переменных

Алгоритм, основанный на теории языков образцов

- E – образец-шаблон, P – произвольный образец

  1) Приведение P в нормальную форму

  2) Построение общего формата для E и P

  3) Решение уравнений Ei : Pi соответствующих элементов

  4) Проверка отсутствия противоречий среди решений в случае кратных t-переменных

Особенности реализации

Проблема	**Решение**
Переопределенные переменные	Проход переименования переменных
Связанные переменные	Организация контекста
Предложения в условиях, присваиваниях, замыканиях, блоках	Рекурсивный обход
Условия в экранирующих предложениях	Пропускаются

Групповое экранирование

Предшествующие предложения в совокупности покрывают то же, или большее множество возможных значений, что и само это предложение.

```
Map {
  t.Fn t.Next e.Tail
    = <Apply t.Fn t.Next>
      <Map t.Fn e.Tail>;
  t.Fn /* пусто */
    = /* пусто */;
  t.FUNC e.items
    = <Map@0 t.FUNC e.items>;
}
```

```
MultiScreening {
  e.x 'A' = 1;
  e.x t.1 t.2 t.3 e.y = 2;
  'A' t.1 = 3;
  'A' e.x0 = 4;
}
```

Регулярные выражения и автоматы

- Идея: представить образцы в виде регулярных выражений, проверить вложение распознающих их автоматов (правильная скобочная последовательность, с ограничением на конечную глубину)
- Язык образца P вкладывается в язык мультиобразца E

  ⇒ E экранирует P
- Ограничение на экранирующие образцы:

  отсутствие кратных e-переменных

Регулярные выражения и автоматы

Для построения регулярного выражения введем отображение h , такое что:

- буквы алфавита Σ остаются неподвижно,
- пустые скобки отображаются в ε,
- не пустые в ω ,
- s-переменные в любую букву алфавита,
- t-переменные в s-переменную, или скобочное выражение,
- e-переменные в любую комбинацию термов

Регулярные выражения и автоматы

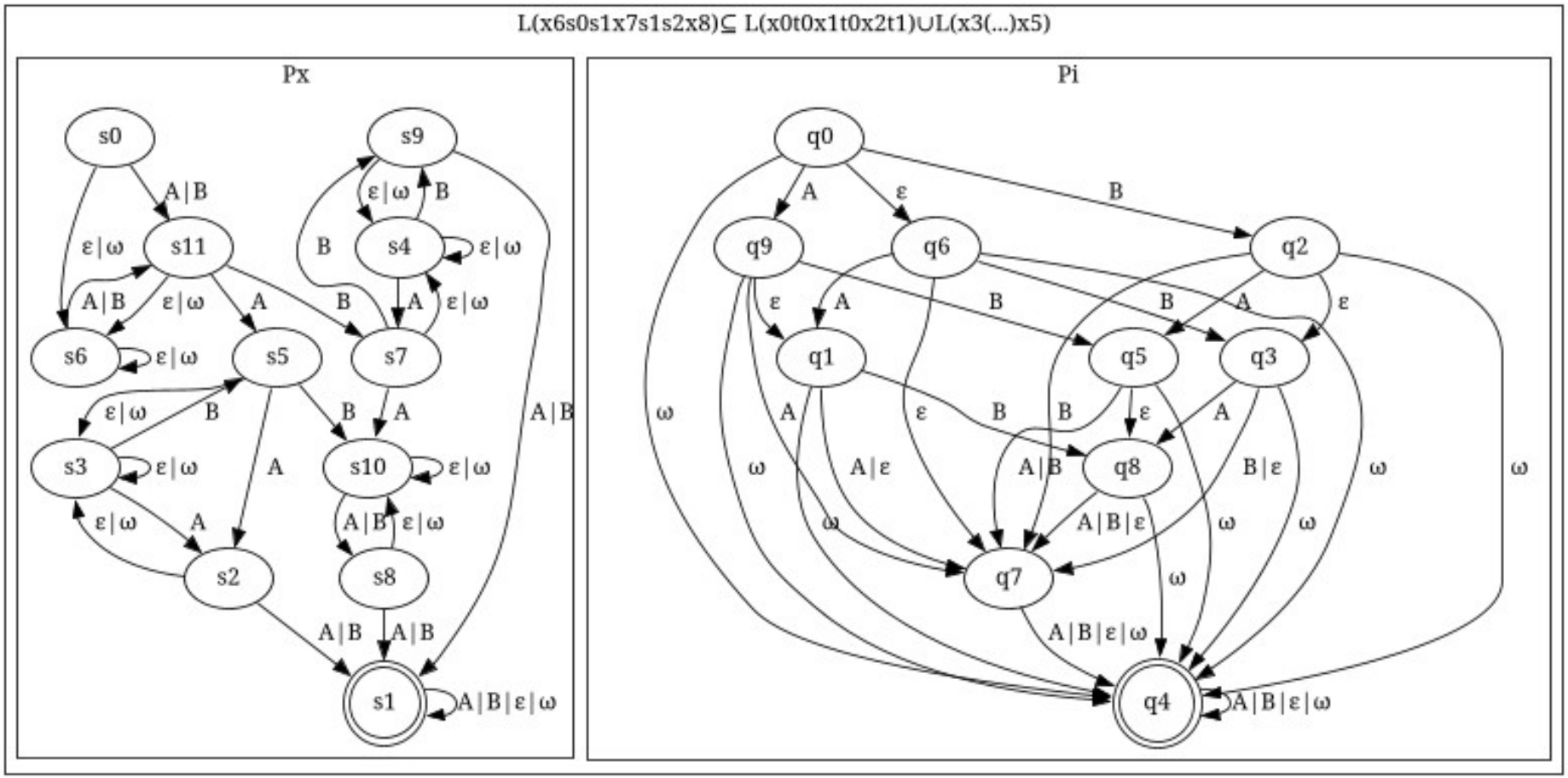
1. Сгенерировать автоматы, для i-го уровня вложения скобочной структуры (скобочное выражение i+1-го уровня заменяется на ω, скобки i-1-го уровня остаются в регулярном выражении);

2. Проверить вложение с учетом повторных переменных;

3. При удачном вложении, вернуться на шаг 1 с i+1;

4. Возвращаем результат вложения.

Регулярные выражения и автоматы



Автомат с совместными состояниями

- Идея: Построения автоматов для правильной скобочной последовательности, с ограничением на конечную глубину, для последовательного сравнения элементов образцов.
- Автомат языка образца P вкладывается в автомат мультиобразца E

  ⇒ E экранирует P
- Ограничение на экранирующие образцы:

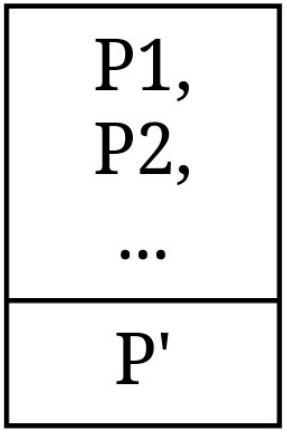
  отсутствие кратных переменных

Автомат с совместными состояниями

Проверка вложения одного регулярного языка в другой сводится к построению декартова произведения состояний двух автоматов. Для всех достижимых состояний этого декартова произведения финальность вкладываемого автомата должна влечь финальность соответствующего состояния второго автомата.

Автомат с совместными состояниями

Введем объединенное состояние, как комбинацию состояния экранирующих предложений и экранируемого. Состояния будут хранить суффиксы образцов.



Автомат с совместными состояниями

Пусть $P' \subseteq P$, где P' — язык экранируемого образца,

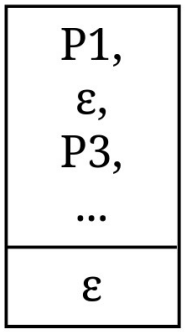
$P = \cup Pi$ — язык мультиобразца.

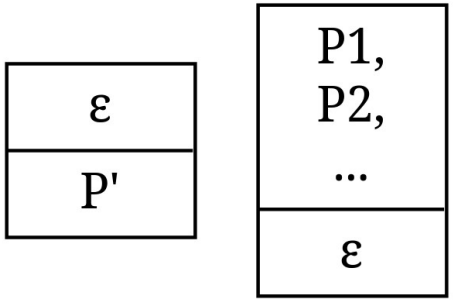
$P' = P1' \cup P2'$ — нормализация образца P'.

Так как

$$P' = P1' \cup P2' \subseteq P \Leftarrow\Rightarrow P1' \subseteq P \wedge P2' \subseteq P ,$$

таким образом, если финальный переход в каком либо состоянии невозможен, то это означает отсутствие группового экранирования предложений.





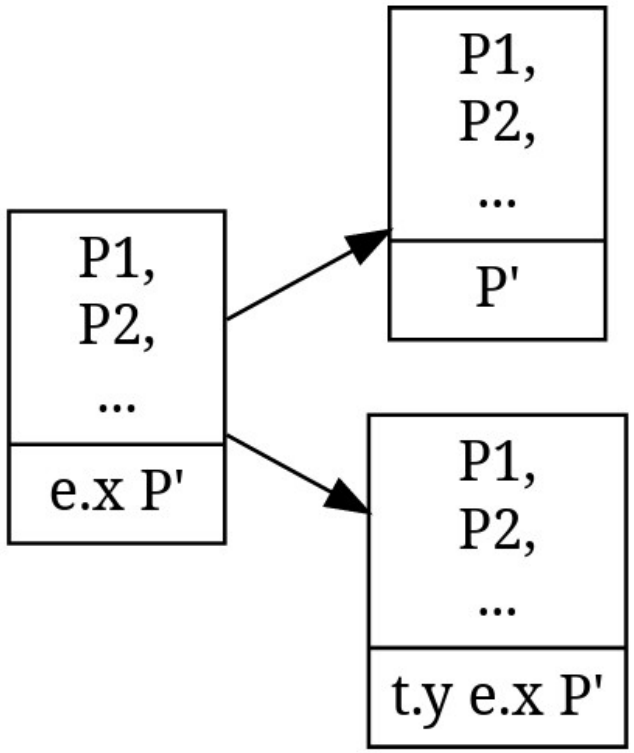
Нормализация e-переменных

Производятся замены:

e.x e.y e.z P ⇒ e.x P

e.x P ⇒ t.eX e.x P

⇒ P



Нормализация t-переменных

- В Рефале-5λ нормализация не требуется (АДТ)
- В Рефале-5 производятся замены, пока текущая глубина скобочного выражения меньше максимального:

t.x P ⇒ s.tX P

⇒ (e.tX) P

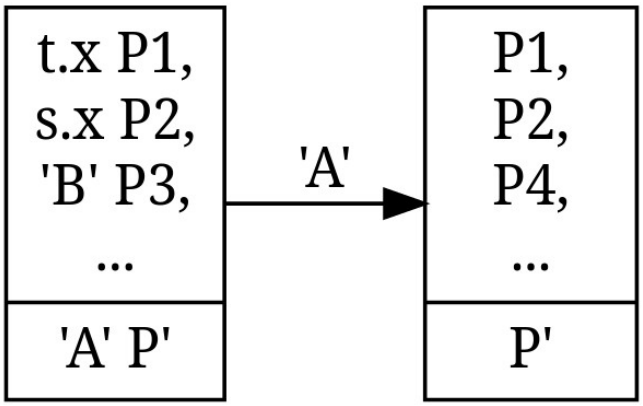
Нормализация t-переменных

В Рефале-5 производятся замены, пока текущая глубина скобочного выражения меньше максимального:

```
Screening1 {
  s.X = False;
  (e.X) = Fails;
  t.X = True;
}
```

```
Screening2 {
  s.X = False;
  (e.X) = Fails;
  ((e.Y)) = True;
}
```

Переходы



- По ’A’ ’A’ P → P
- По ’A’ s.X P → P
- По ’A’ t.X P → P
- По s s.X P → P
- По s t.X P → P
- По t t.X P → P
- По «(» (P) → P)
- По «(» t.X P → e.Y) P
- По «[» [Name P] → Name t.Inner] P
- По «[» t.X P → t.Name t.Inner] P

Переходы

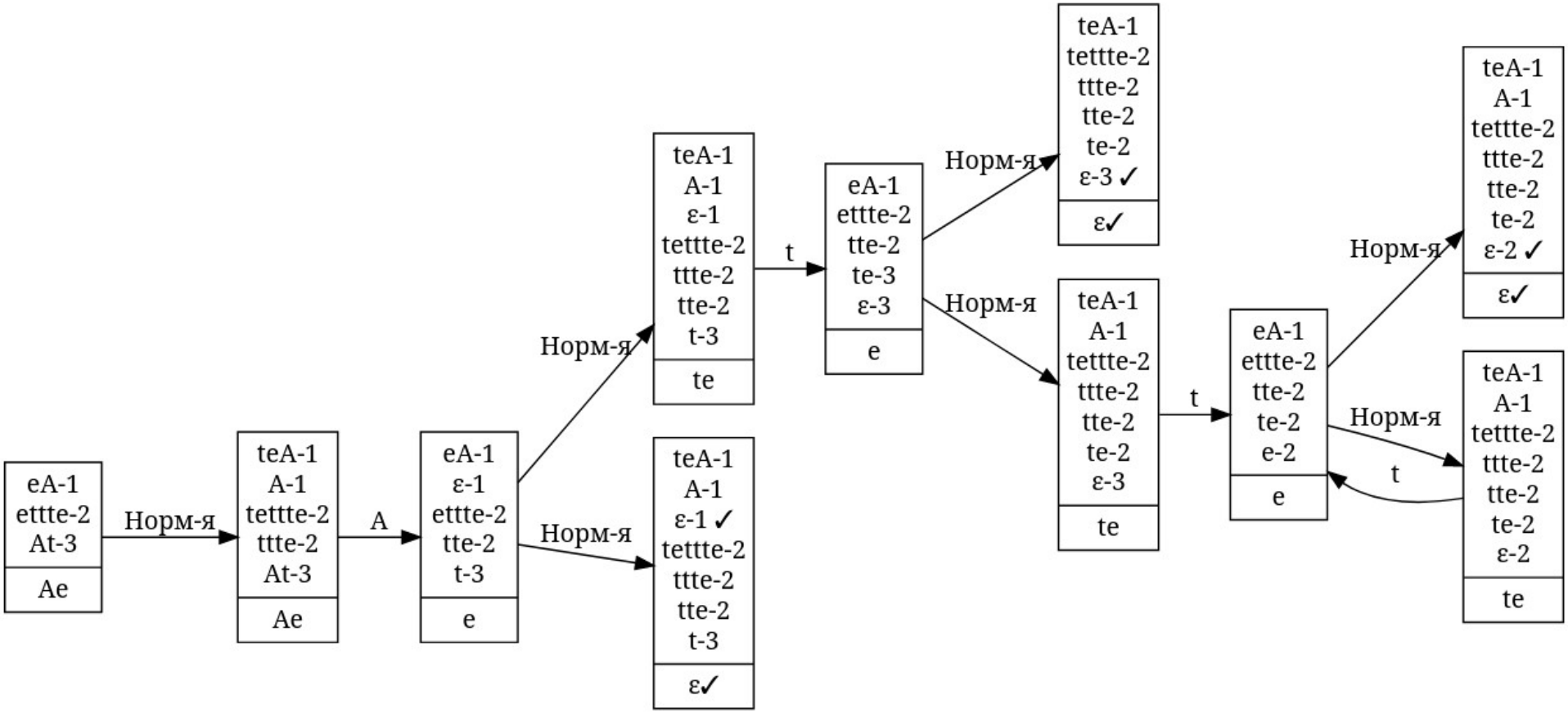
```
Transition {
  (Symbol e.SymInfo)   (Symbol e.SymInfo)   = /*пусто*/;
  (Symbol e.SymInfo)   (Var 's' e.VarName)  = /*пусто*/;
  (Symbol e.SymInfo)   (Var 't' e.VarName)  = /*пусто*/;
  (Var 's' e.VarName1) (Var 's' e.VarName2) = /*пусто*/;
  (Var 's' e.VarName1) (Var 't' e.VarName2) = /*пусто*/;
  (Var 't' e.VarName1) (Var 't' e.VarName2) = /*пусто*/;
  (Bracket_L)          (Bracket_L)          = /*пусто*/;
  (Bracket_L)          (Var 't' e.VarName)
     = (Var 'e_')(Bracket_R);
  (Bracket_R)          (Bracket_R)          = /*пусто*/;
  (ADT-Bracket_L)      (ADT-Bracket_L)      = /*пусто*/;
  (ADT-Bracket_L)      (Var 't' e.VarName)
    = (Var 't_')(Var 'e_')(ADT-Bracket_R);
  (ADT-Bracket_R)      (ADT-Bracket_R)      = /*пусто*/;
  e._ = False;
}
```

Алгоритм

- Нормализация e- и t-переменных
- Проверка наличия финальных состояний
- Шаг по каждому состоянию:

  посещенные состояния пропускаются,

  невозможность перехода — отсутствие экранирования
- Повторяем шаг 1, пока множество состояний в очереди не пусто


eA-1
ettte-2
At-3
Ae
Норм-я
teA-1
A-1
tettte-2
ttte-2
At-3
Ae
A
eA-1
ε-1
ettte-2
tte-2
t-3
e
Норм-я
teA-1
A-1
ε-1
tettte-2
ttte-2
tte-2
t-3
te
Норм-я
teA-1
A-1
ε-1 ✓
tettte-2
ttte-2
tte-2
t-3
ε✓
t
eA-1
ettte-2
tte-2
te-3
ε-3
e
Норм-я
teA-1
tettte-2
ttte-2
tte-2
te-2
ε-3 ✓
ε✓
Норм-я
teA-1
A-1
tettte-2
ttte-2
tte-2
te-2
ε-3
te
t
eA-1
ettte-2
tte-2
te-2
e-2
e
Норм-я
teA-1
A-1
tettte-2
ttte-2
tte-2
te-2
ε-2 ✓
ε✓
Норм-я
t
teA-1
A-1
tettte-2
ttte-2
tte-2
te-2
ε-2
te

Механизм обработки ошибок

Управление предупреждениями производится через флаг -W командной строки.

Доступные опции:

-Wall — включение всех предупреждений компилятора;

-W<name> — включение конкретного предупреждения по имени name;

-W[no-]error[=<name>] — включить/отключить режим, в котором все предупреждения (или конкретное предупреждение) считаются ошибками и прерывают компиляцию.

Ключи для проверки на экранирование имеют формат:

-Wscreening -Wmuiltiscreening, -Wmuiltiscreening5.

Пример компиляции программы:

rlc -Wmultiscreening5 program.ref