Universidade do Minho

Mestrado em Engenharia Informática

UCE30: Engenharia de Linguagens

Módulo: Engenharia Gramatical

Exercício para Avaliação n.º1

Trabalho desenvolvido por:

Bruno Azevedo

Miguel Cosa

Índice

Descrição do problema

Era pretendido que se usasse o processador da Lista de Elementos Mistos (palavras e inteiros), que foi desenvolvido nas aulas, e alterar a sua Gramática de Atributos (GA) de modo a calcular o somatório de cada sequência se inteiros que surjam a seguir à palavra "soma".

Exemplo:

A frase "[a,1,2,b,soma,3,a,4,soma,b,2,7]"

Dá como resultado: [7,9]

Resolução

Gramática Independente do Contexto

Observando o problema formulámos a seguinte Gramática Independente do Contexto (GIC):

GIC = (T, N, S, P)
Símbolos terminais (T): {str, int, '[' , ']', ','}
Símbolos não terminais (N): {Lista, Elementos, Elemento}
Símbolo Inicial (S): Lista
Produções (P):

```
P0: Lista       -> '[' Elementos ']'
P1: Elementos   -> Elemento
P2:             | Elemento ',' Elementos
P3: Elemento    -> int
P4:             | str
```

str = [a-zA-Z]+
int = [0-9]+

Gramática de Atributos

Depois de definida e analisada a GIC, definimos a Gramática de Atributos (GA) como:
GA = (GIC, A, RC, CC, RT)

Para resolver este problema, usamos 3 variáveis:

- Sum
- Sum_flag
- Result

A variável *sum_flag* é inicializada a 0 e quando for encontrada a palavra "soma" fica 1 e coloca a variável *sum* a 0, a partir deste momento quando encontrar um elemento inteiro vai adiciona-lo a *sum*.

Result é um array que vai conter o resultado, ele é alterado quando se encontrada a palavra "soma" e a variável *sum* é maior que 0, vai ficar: result = result.add(sum).

Os símbolos não terminais podem ter atributos sintetizados e herdados, por isso, a forma que encontramos para resolver o problema de saber quando adicionar ao array *result* o *sum* foi dizer que os símbolos não terminais tem:

- Atributos sintetizados:
  - out_sum
  - out_sum_flag
  - out_result
- Atributos herdados:
  - in_sum
  - in_sum_flag
  - in_result

O que é pretendido com esta solução, é que o símbolo não terminal receba a informação do estado atual (atributos in) e depois devolva a informação atualizada (atributos out).

Atributos (A)

Lista	result : ArrayList<Integer>
Elementos	in_result : ArrayList<Integer> out_result : ArrayList<Integer> in_sum : int out_sum : int in_sum_flag : int out_sum_flag :int
Elemento	in_result : ArrayList<Integer> out_result : ArrayList<Integer> in_sum : int out_sum : int in_sum_flag : int out_sum_flag :int

Regra de Cálculo (RC), Condição Contextual (CC) e Regra de Tradução (RT)

```
P0: Lista -> ‘[‘ Elementos ‘]’
      Lista.result = Elementos.result
      Elementos.in_result = new ArrayList<Integer>();
      Elementos.in_sum = 0
      Elementos.in_sum_flag = 0

P1: Elementos -> Elemento
      Elemento.in_result = Elementos.in_result
      Elemento.in_sum = Elementos.in_sum
      Elemento.in_sum_flag = Elementos.in_sum_flag
      Elementos.out_result = Elemento.out_result
      Elementos.out_sum = Elemento.out_sum
      Elementos.out_sum_flag = Elemento.out_sum_flag
```

```
P2: Elementos₀ -> Elemento ‘,’ Elementos₁
      Elementos₀.out_sum = Elementos₁.out_sum
      Elementos₀.out_sum_flag = Elementos₁.out_sum_flag
      Elementos₀.out_result = Elementos₁.out_result
      Elemento.in_sum = Elementos₀.in_sum
      Elemento.in_sum_flag = Elementos₀.in_sum_flag
      Elemento.in_result = Elementos₀.in_result
      Elementos₁.in_sum = Elemento.out_sum
      Elementos₁.in_sum_flag = Elemento.out_sum_flag
      Elementos₁.in_result = Elemento.out_result

P3: Elemento -> int
      Elemento.out_result = Elemento.in_result
      Elemento.out_sum = function refresh_sum
      Elemento.out_sum_flag = Elemento.in_sum_flag

      $1 = Elemento.in_sum, $2 = Elemento.in_sum_flag, $3 = str.value
      int refresh_sum($1,$2,$3){
            if($2==1) return $1+$3; else return $1;
      }

P4: Elemento -> str
      Elemento.out_result = function refresh_result
      Elemento.out_sum = function refresh_sum
      Elemento.out_sum_flag = function refresh_sum_flag

      $1 = Elemento.in_result, $2 = Elemento.in_sum,
      $3 = Elemento.in_sum_flag, $4 = str.value
      ArrayList<Integer> refresh_result($1, $2, $3, $4){
            if($4.equals(“soma”) && $3 == 1 && $2 > 0)
                  return $1.add($2); else return $2;
      }

      $1 = Elemento.in_sum, $2 = str.value
      int refresh_sum($1,$2){
            if($2.equals(“soma”)) return 0; else return $1;
      }

      $1 = Elemento.in_sum_flag, $2 = str.value
      int refresh_sum_flag($1, $2){
            if($2.equals(“soma”)) return 1; else return $1;
      }
```

Resolução utilizando o VisualLisa

Este problema foi também resolvido visualmente com a ajuda da ferramenta VisualLisa.

Produções

As **Produções (P):**

```
P0: Lista       -> '[' Elementos ']'
P1: Elementos   -> Elemento
P2:             | Elemento ',' Elementos
P3: Elemento    -> int
P4:             | str
```

da gramática independente de contexto que já está definida, quando representada visualmente em VisualLisa fica como a Figura 1.

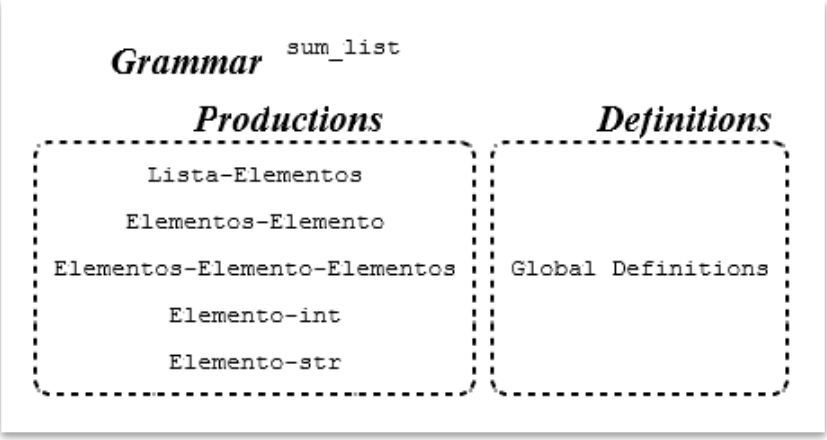

Figura 1 Produções

Lista -> Elementos

A produção Lista -> Elementos visualmente fica como mostra a Figura 2, em que também já aparecem os atributos de cada símbolo.

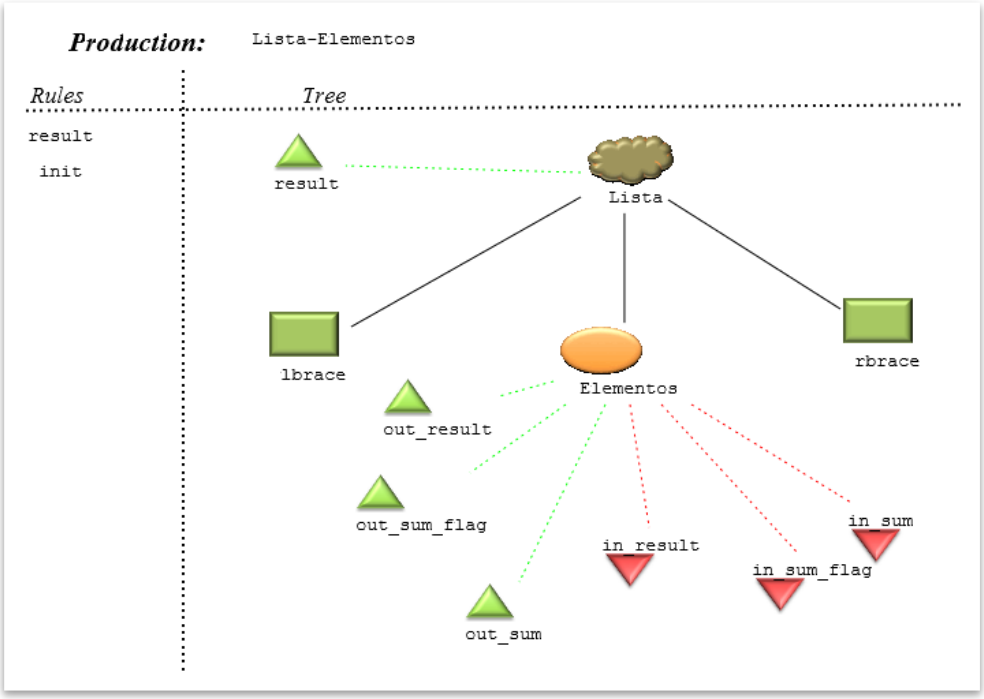

Figura 2 Produção P0

Elementos -> Elemento

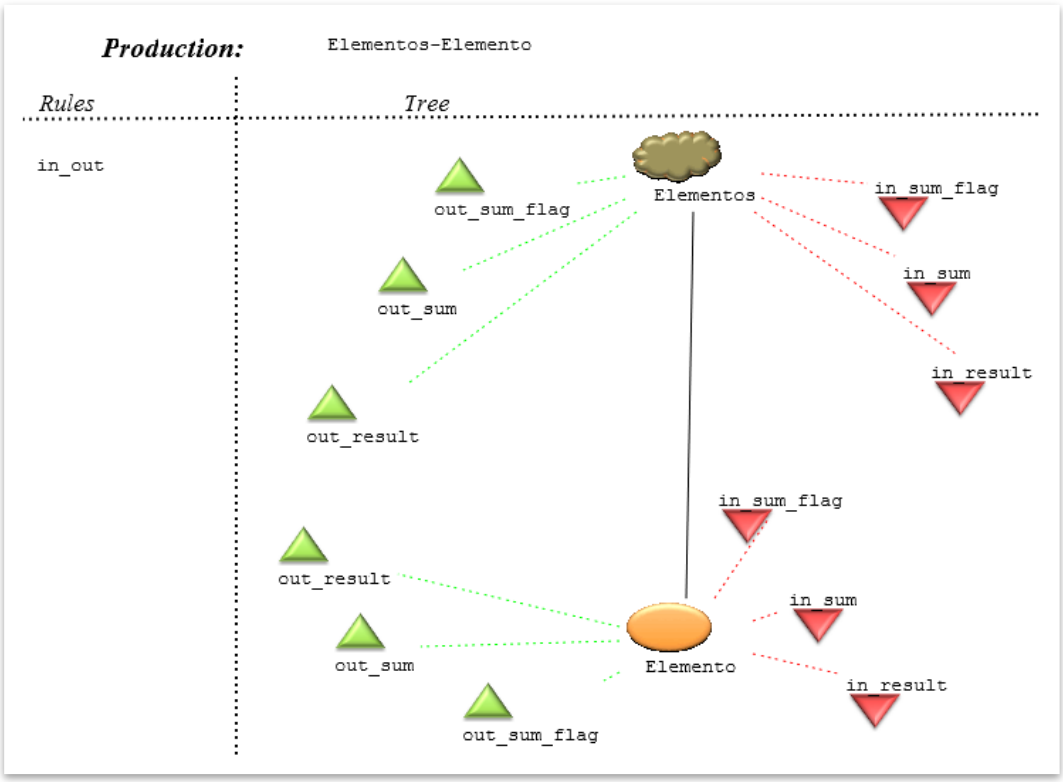

Figura 3 Produção P1

Elementos -> Elemento ',' Elementos

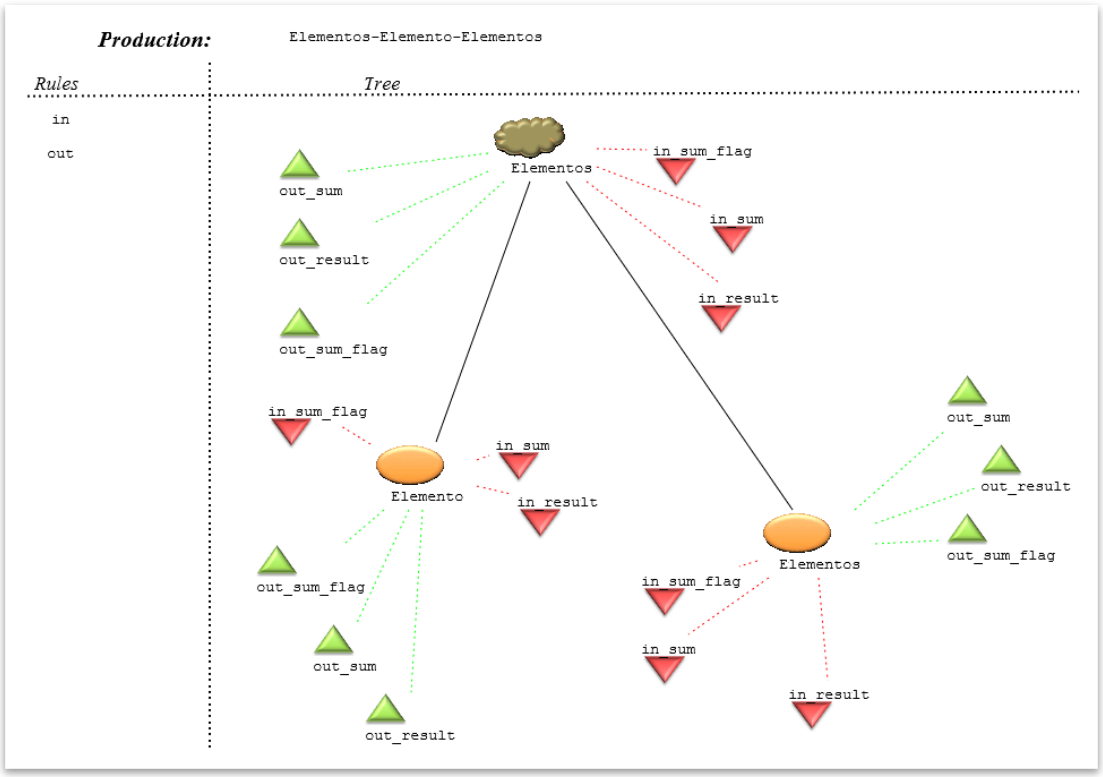

Figura 4 Produção P2

Elemento -> int

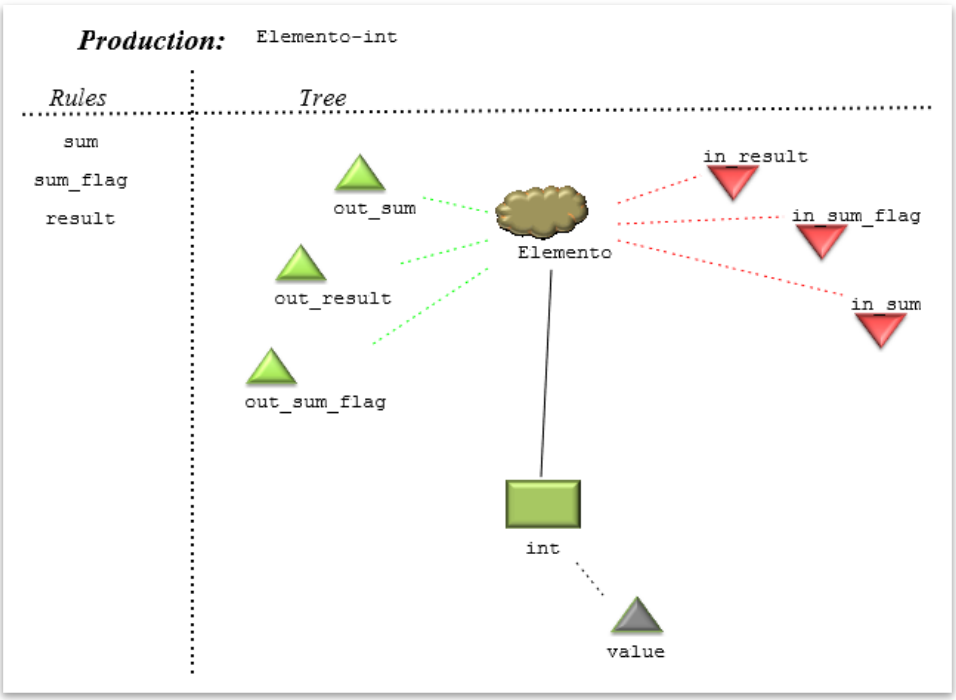

Figura 5 Produção P3

Elemento -> str

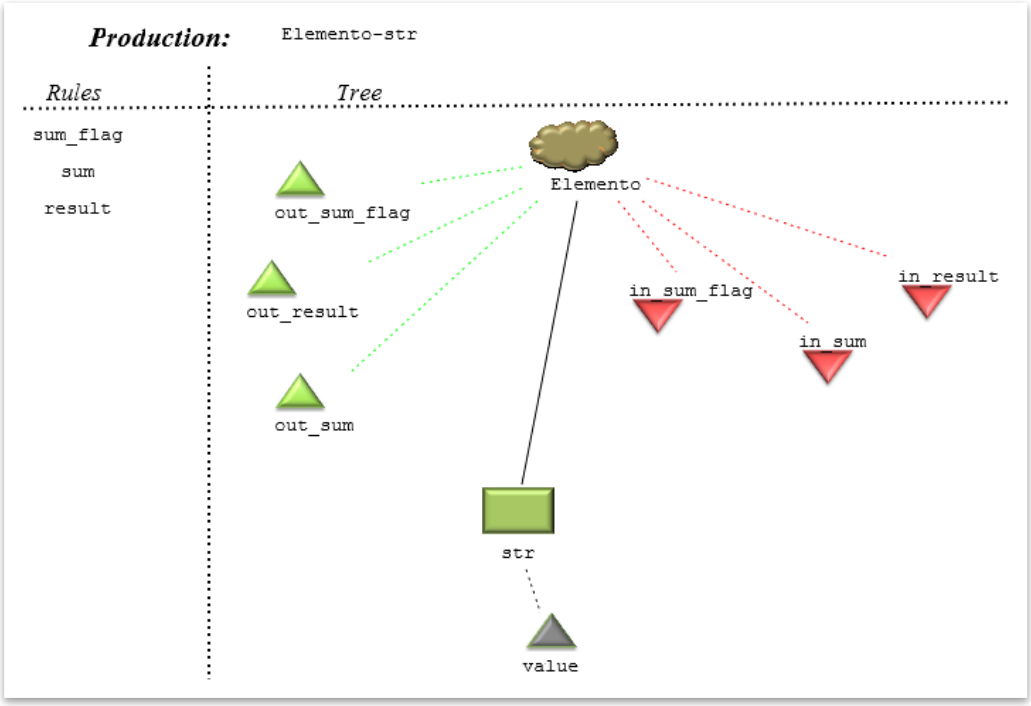

Figura 6 Produção P4

Regras

Lista -> Elementos

result

Esta é a regra que devolve o resultado da frase que for dada para calcular e é calculada por: `Lista.result = Elementos.result`

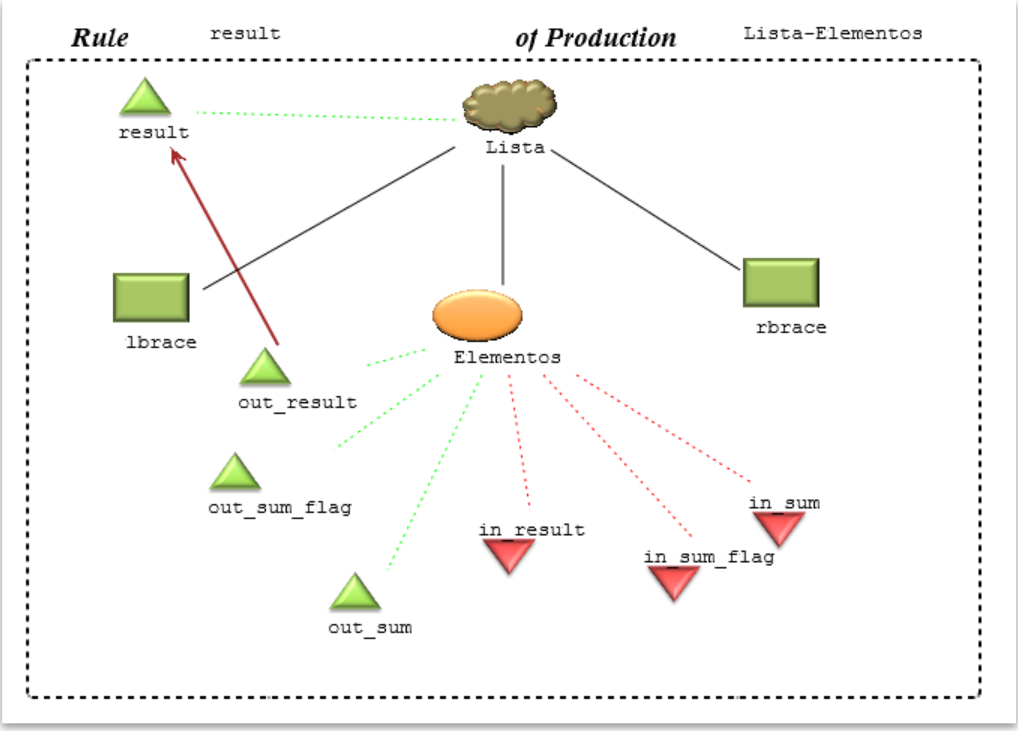

Figura 7 Regra para Lista.result

init

O que é feito nesta regra é inicializar as variáveis *in_sum* e *in_sum_flag* a zero.

```
Elementos.in_sum = 0
Elementos.in_sum_flag = 0
```

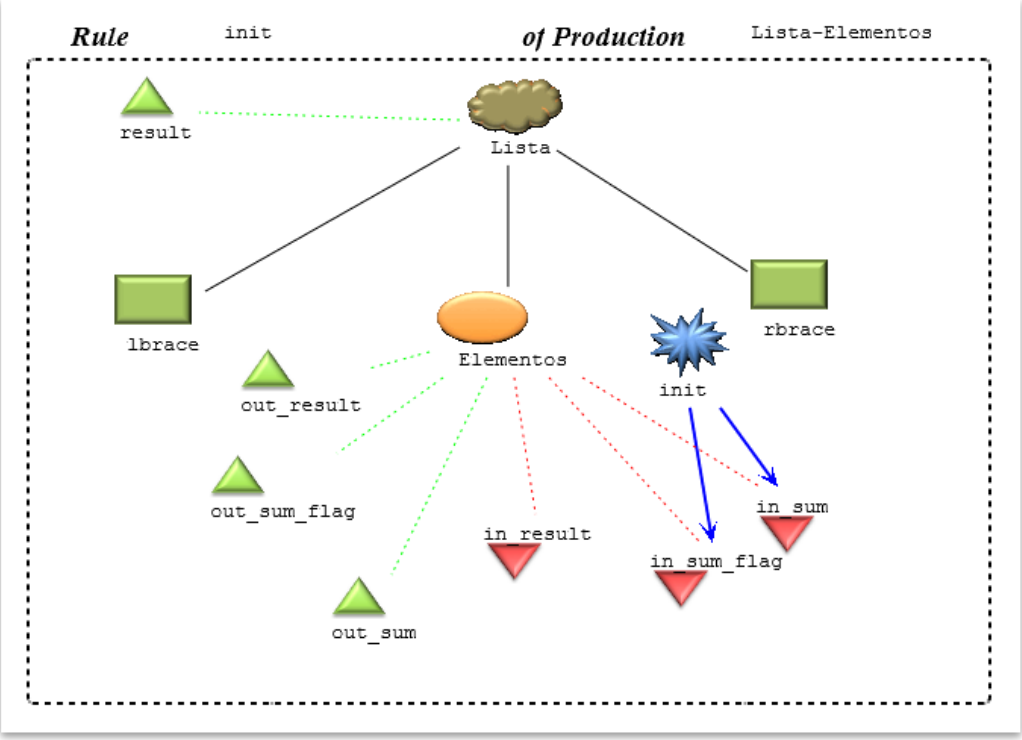

Figura 8 Regra para inicializar variáveis

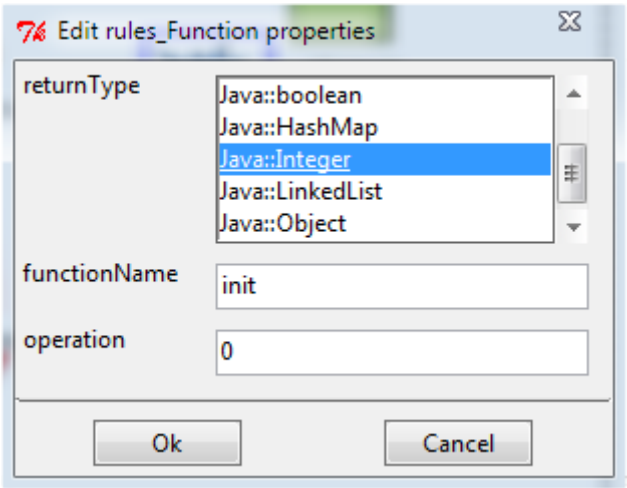

Figura 9 Função init

Elementos -> Elemento

in_out

Aqui estão as regras:

```
Elemento.in_result = Elementos.in_result
Elemento.in_sum = Elementos.in_sum
Elemento.in_sum_flag = Elementos.in_sum_flag
Elementos.out_result = Elemento.out_result
Elementos.out_sum = Elemento.out_sum
Elementos.out_sum_flag = Elemento.out_sum_flag
```

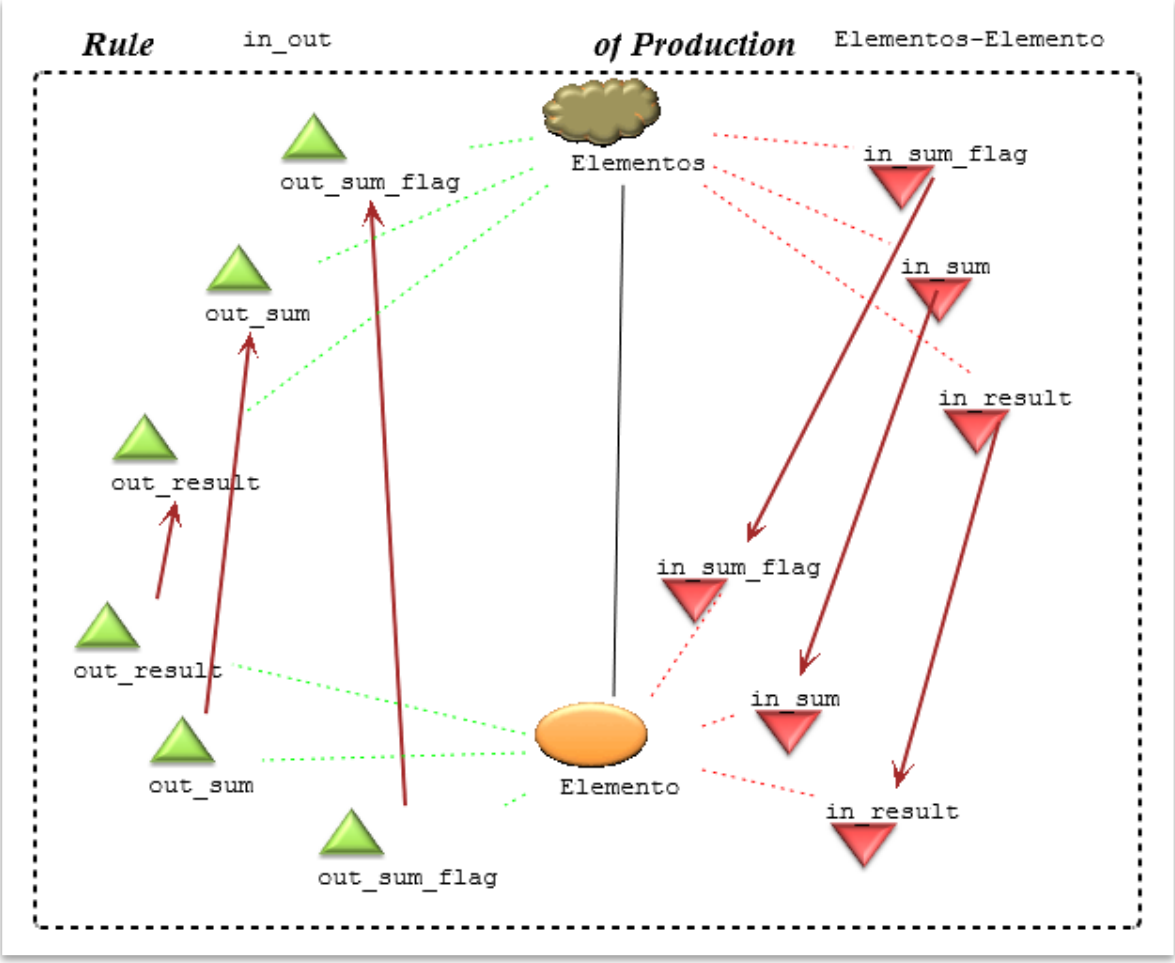

Figura 10 Regras in_out

Elementos -> Elemento ',' Elementos

In

Regras:

$$Elemento.in\_sum = Elementos_0.in\_sum$$
$$Elemento.in\_sum\_flag = Elementos_0.in\_sum\_flag$$
$$Elemento.in\_result = Elementos_0.in\_result$$
$$Elementos_1.in\_sum = Elemento.out\_sum$$
$$Elementos_1.in\_sum\_flag = Elemento.out\_sum\_flag$$
$$Elementos_1.in\_result = Elemento.out\_result$$

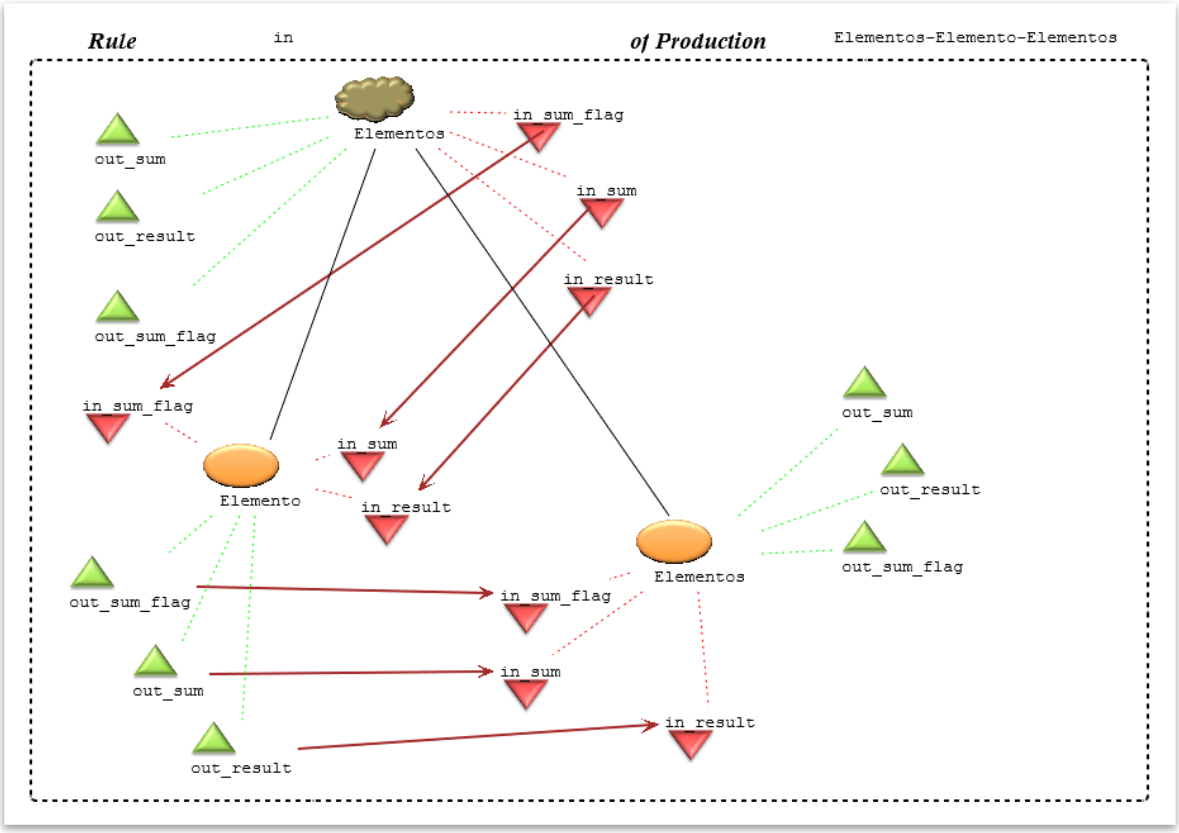

Figura 11 Regras in

out

$Elementos_0.out\_sum = Elementos_1.out\_sum$
$Elementos_0.out\_sum\_flag = Elementos_1.out\_sum\_flag$
$Elementos_0.out\_result = Elementos_1.out\_result$

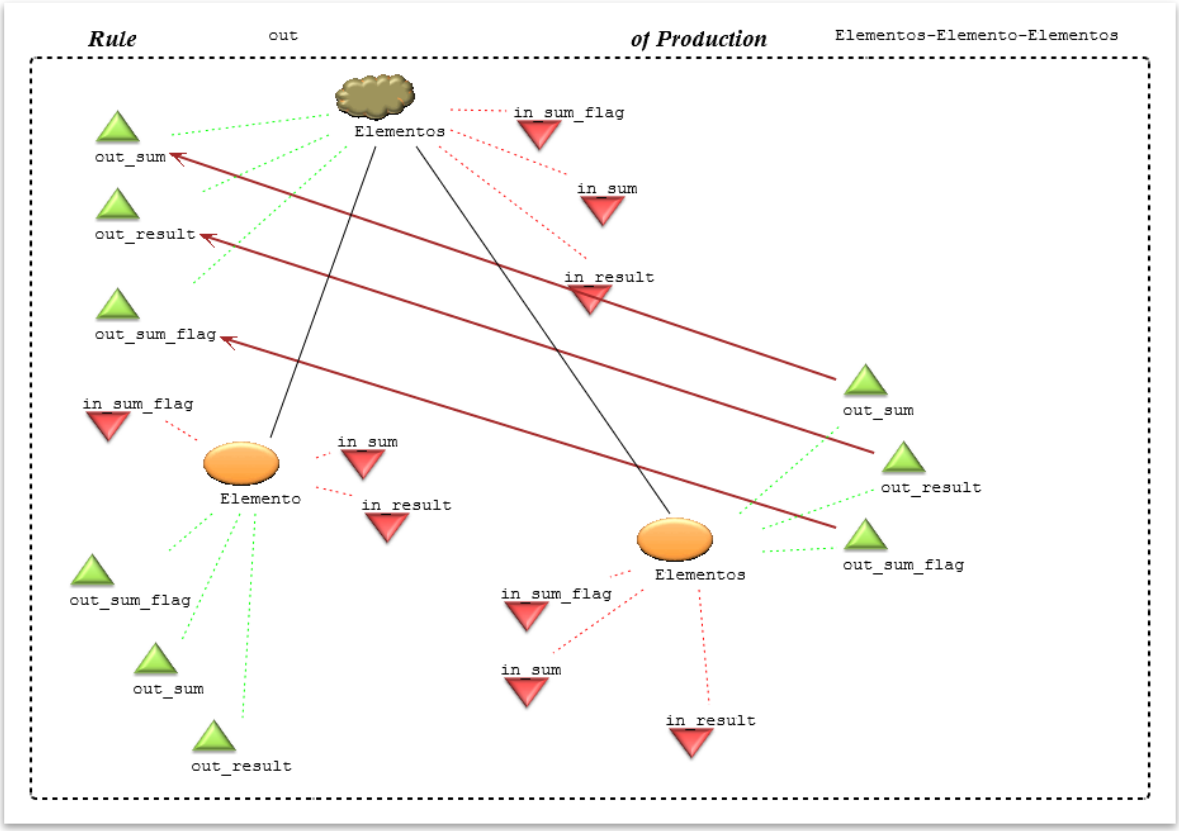

Figura 12 Regras out

Elemento -> int

sum

```
Elemento.out_sum = function refresh_sum
```

Em que a função é definida por:

```
$1 = Elemento.in_sum, $2 = Elemento.in_sum_flag, $3 = str.value
int refresh_sum($1,$2,$3){
      if($2==1) return $1+$3; else return $1;
}
```

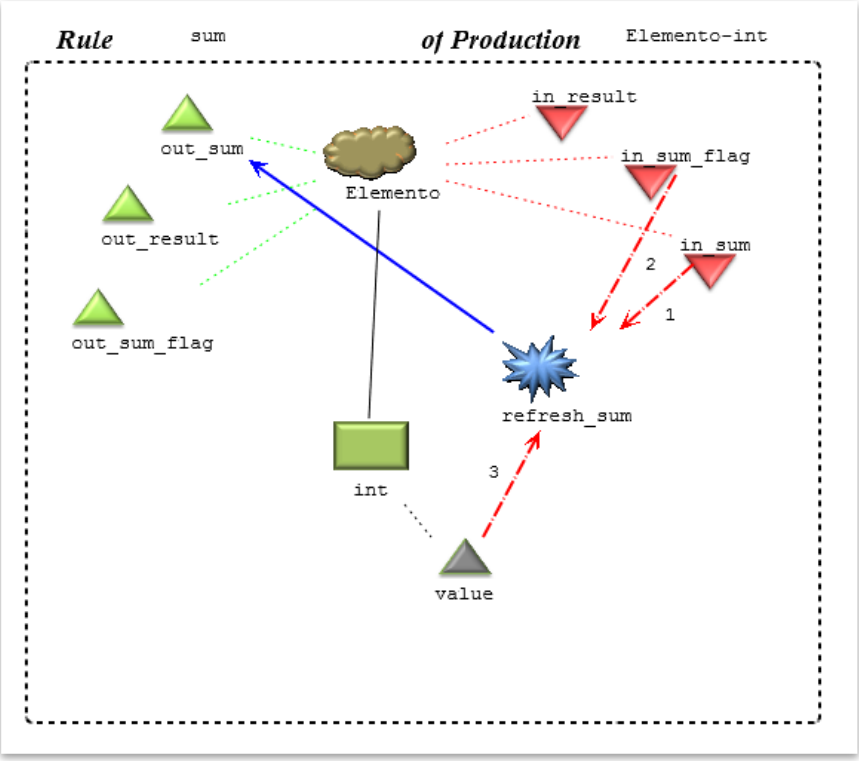

Figura 13 Regra sum

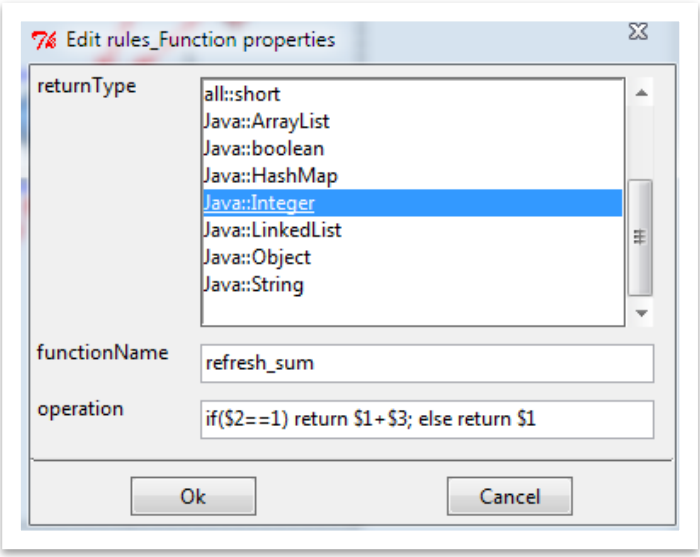

Figura 14 Função refresh_sum

sum_flag

```
Elemento.out_sum_flag = Elemento.in_sum_flag
```

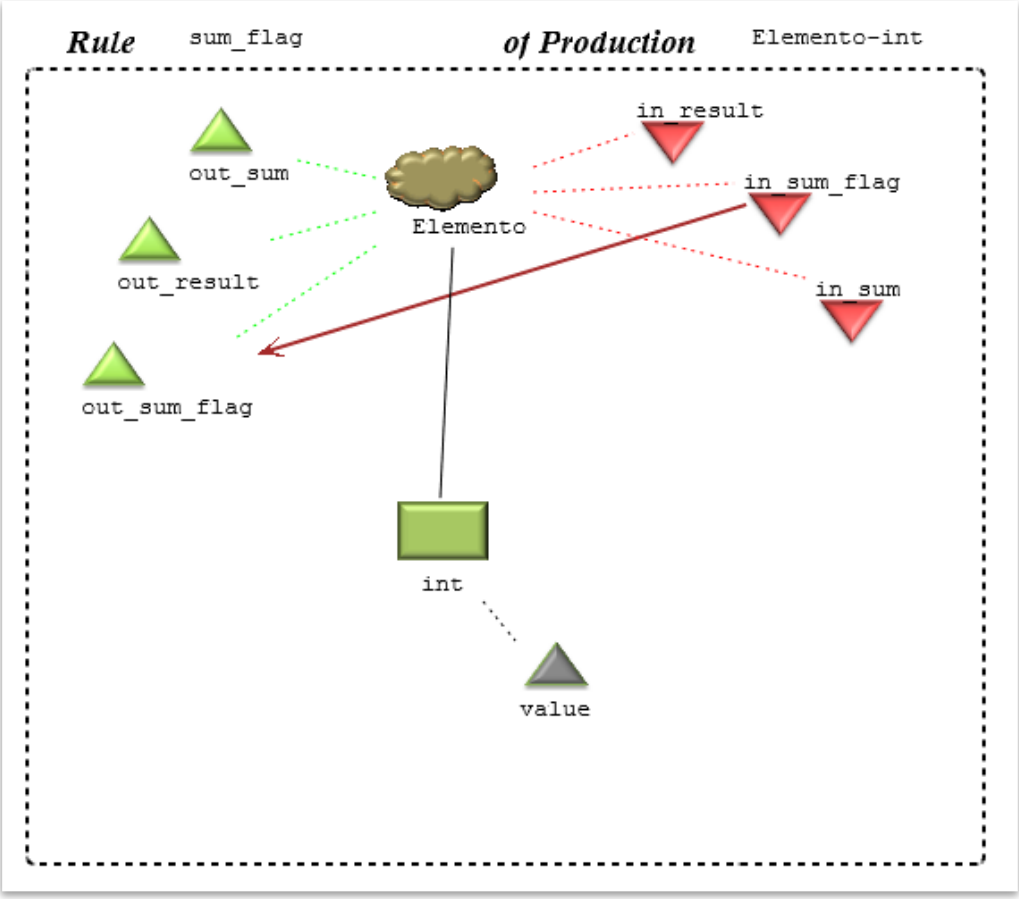

Figura 15 Regra sum_flag

result

```
Elemento.out_result = Elemento.in_result
```

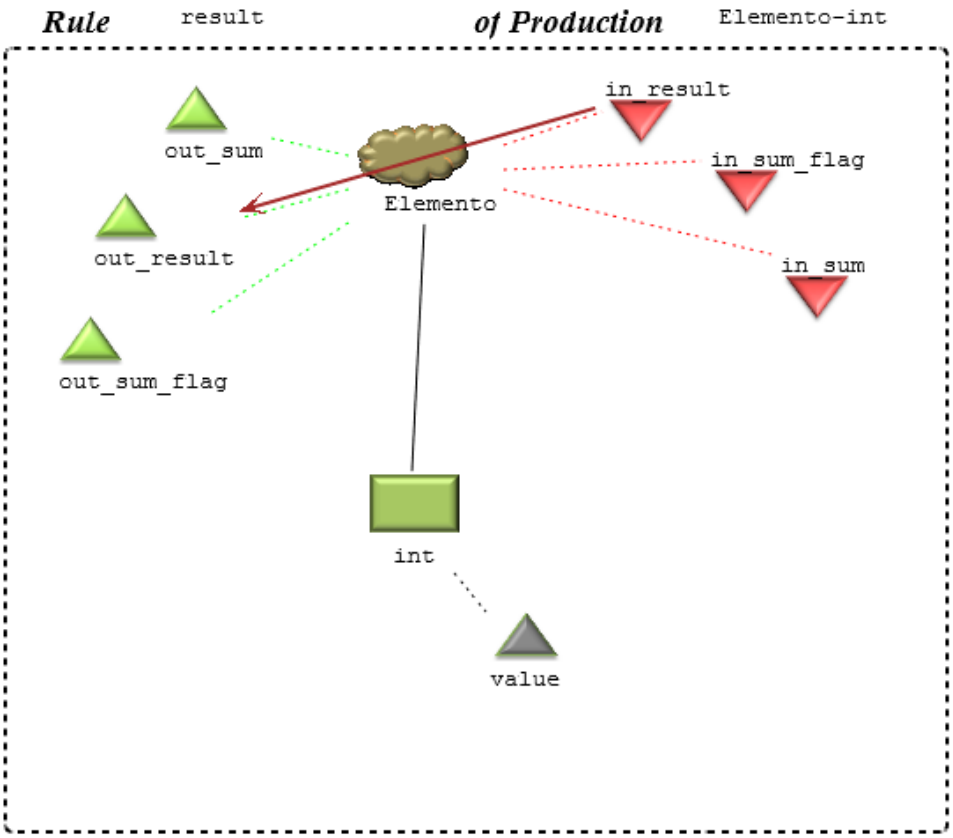

Figura 16 Regra result

Elemento -> str

sum_flag

```
Elemento.out_sum_flag = function refresh_sum_flag
```

Em que a função é definida por:

```
$1 = Elemento.in_sum_flag, $2 = str.value
int refresh_sum_flag($1, $2){
      if($2.equals(“soma”)) return 1; else return $1;
}
```

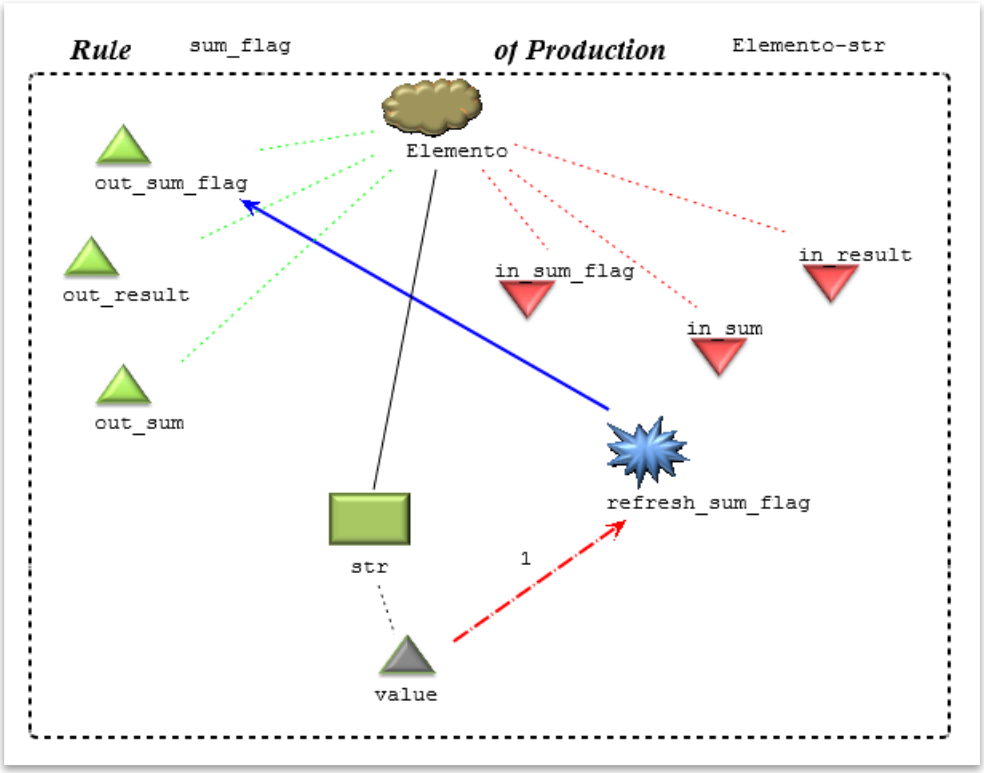

Figura 17 Regra sum_flag

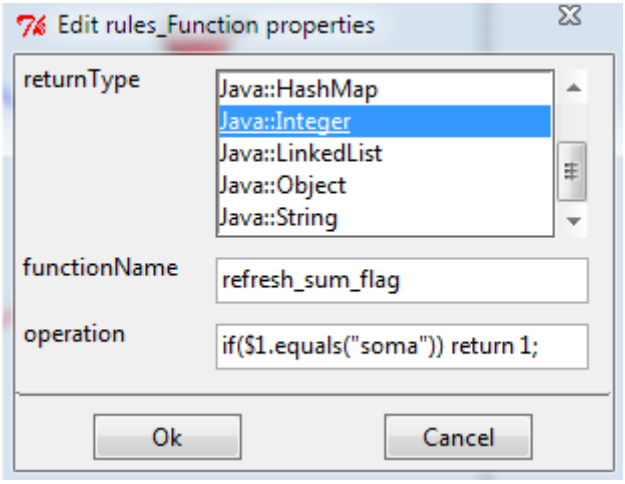

Figura 18 Função refresh_sum_flag

sum

```
Elemento.out_sum = function refresh_sum
```

Em que a função é definida por:

```
$1 = Elemento.in_sum, $2 = str.value
int refresh_sum($1,$2){
      if($2.equals("soma")) return 0; else return $1;
}
```

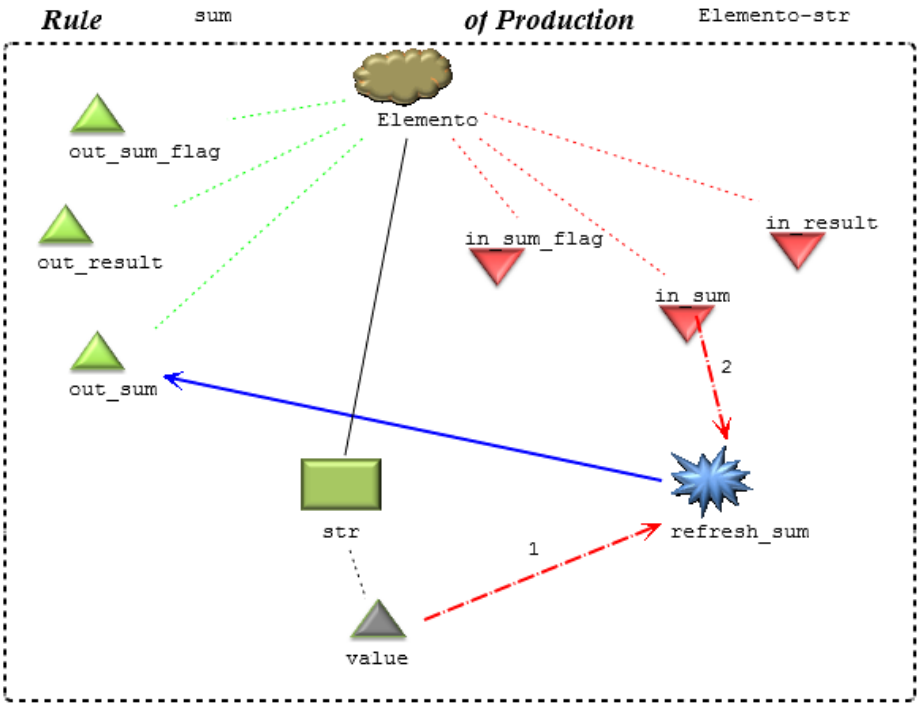

Figura 19 Regra sum

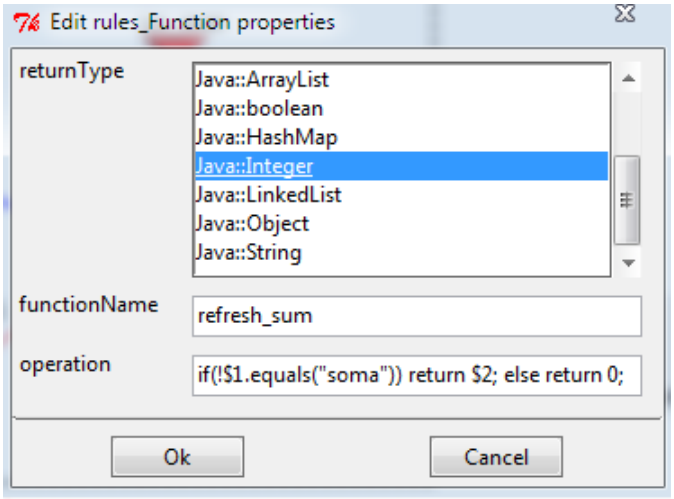

Figura 20 Função refresh_sum

result

```
Elemento.out_result = function refresh_result
```

Em que a função é definida por:

```
$1 = Elemento.in_result, $2 = Elemento.in_sum,
$3 = Elemento.in_sum_flag, $4 = str.value
ArrayList<Integer> refresh_result($1, $2, $3, $4){
      if($4.equals("soma") && $3 == 1 && $2 > 0)
            return $1.add($2); else return $2;
}
```

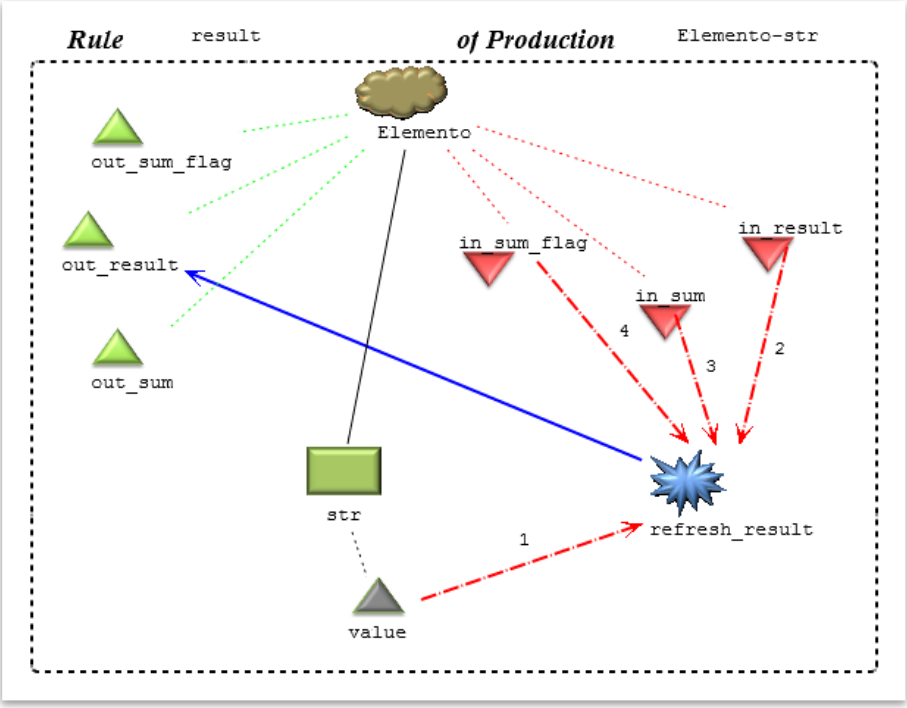

Figura 21 Regra result

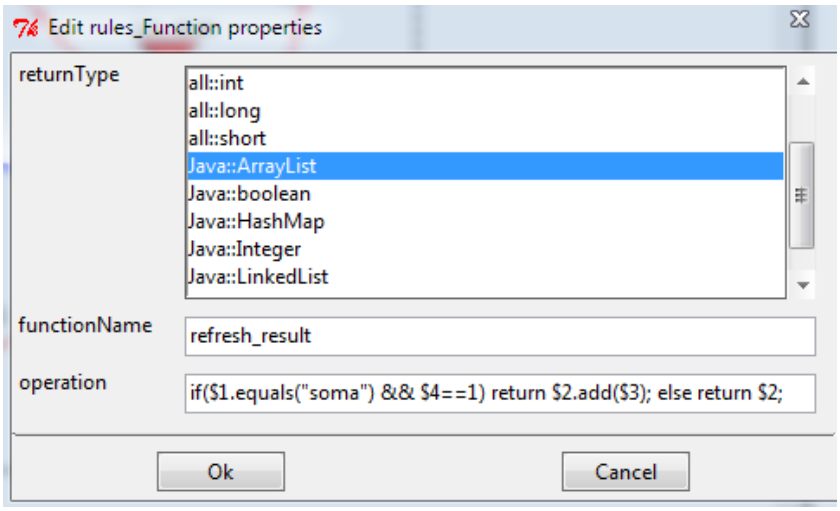

Figura 22 Função refresh_result