Engenhraria Gramatical
Exercício para Avaliação n.º 1

Bruno Azevedo Miguel Costa

28 de Novembro de 2011

Resumo

Este documento apresenta a resolução do Exercício Prático n.º 1 do módulo de Engenharia de Linguagens. O exercício está relacionado com Gramática Independente de Contexto e Gramática de Atributos para resolver um problema de cálculo de elementos mistos (palavras e números).

Conteúdo

1 Ambiente de Trabalho

Tal como seria de esperar, um exercício deste tipo é resolvido inicialmente em papel, para ser mais fácil estruturar o problema e fazer uma boa abordagem à resolução que se irá fazer. Depois de analisado e tomado notas no papel, passamos o exercício para a ferramenta VisualLisa. Através desta ferramenta é possível ter uma visão de como as símbolos de toda a linguagem estão relacionados e como "falam" entre si.

2 Descrição informal 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]
```

3 Resolução no papel

3.1 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]+
```

3.2 Gramática de Atributos

Depois de definida e analisada a GIC, definimos a Gramática de Atributos 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).

3.2.1 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
```

3.2.2 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;
        }
```

4 Resolução no VisualLisa

Este problema foi também resolvido visualmente com a ajuda da ferramenta VisualLisa. Esta seção mostra como ficou resolvido visualmente o exercício.

4.1 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.

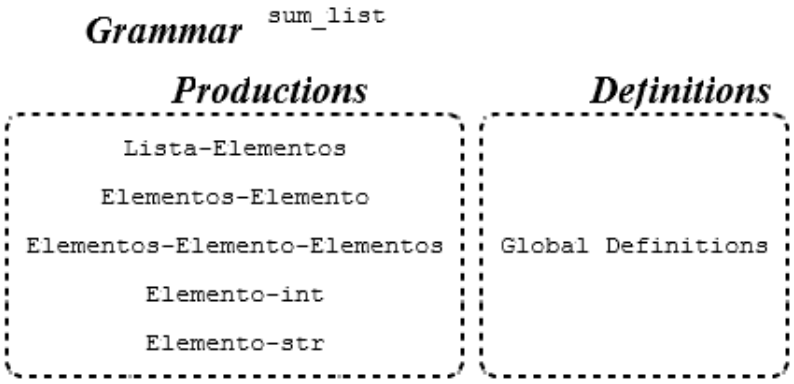

Figura 1: Produções

4.1.1 Lista -> Elementos

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

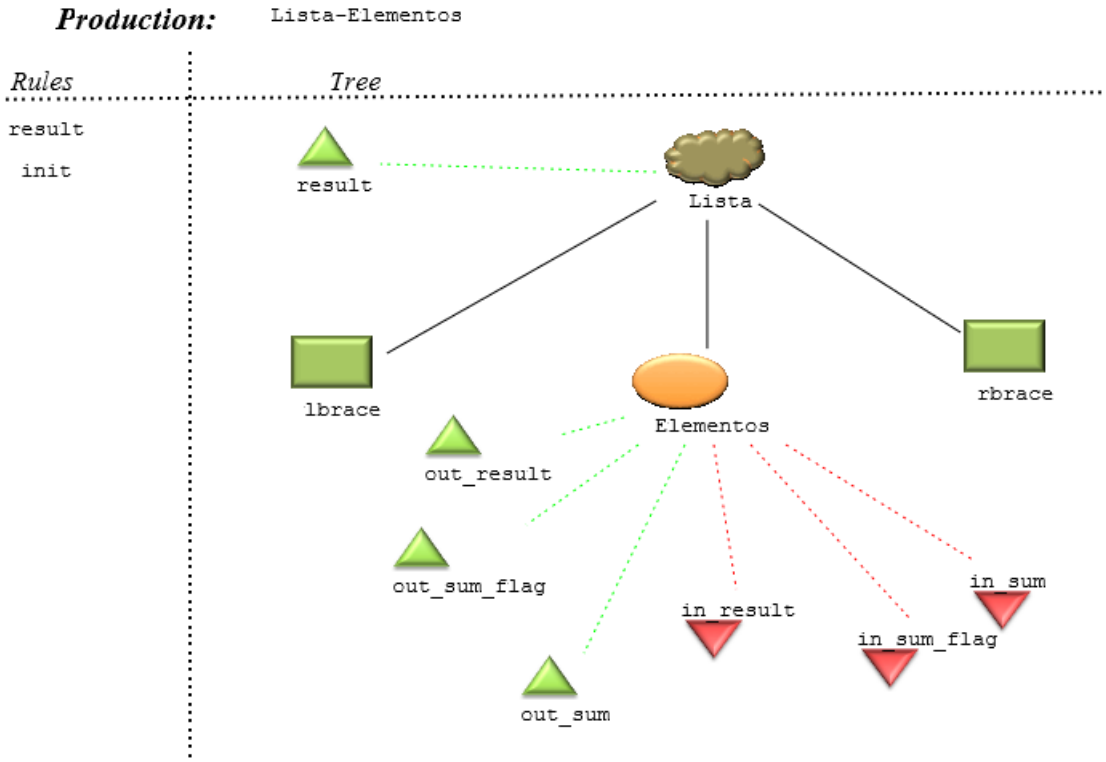

Figura 2: Produção P0

4.1.2 Elementos -> Elemento

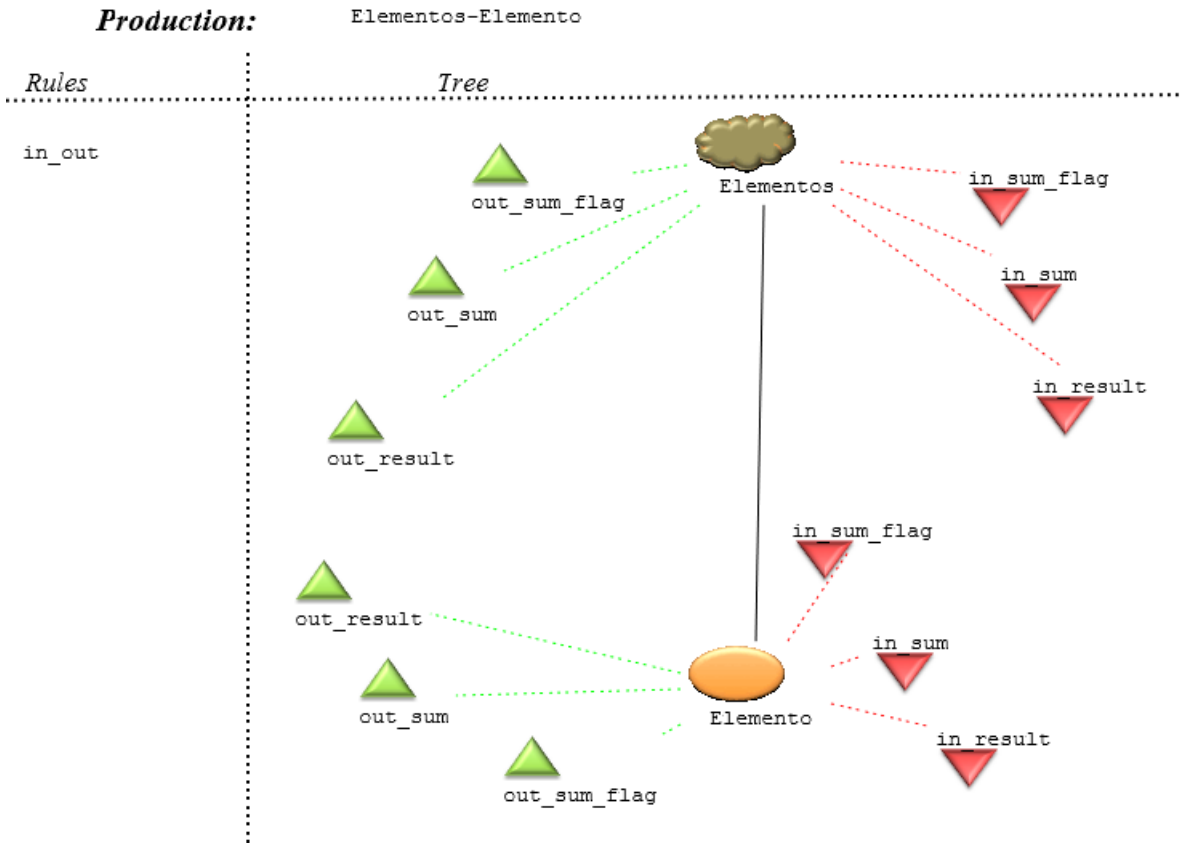

Figura 3: Produção P1

4.1.3 Elementos -> Elemento ',' Elementos

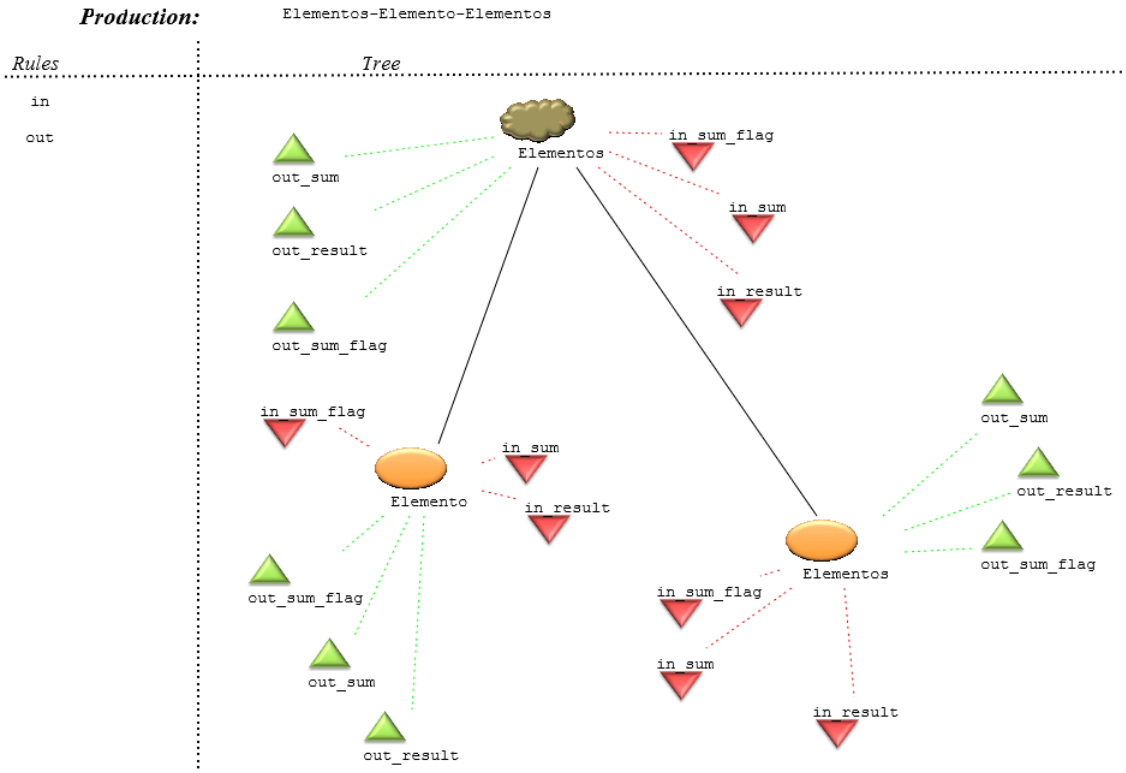

Figura 4: Produção P2

4.1.4 Elemento -> int

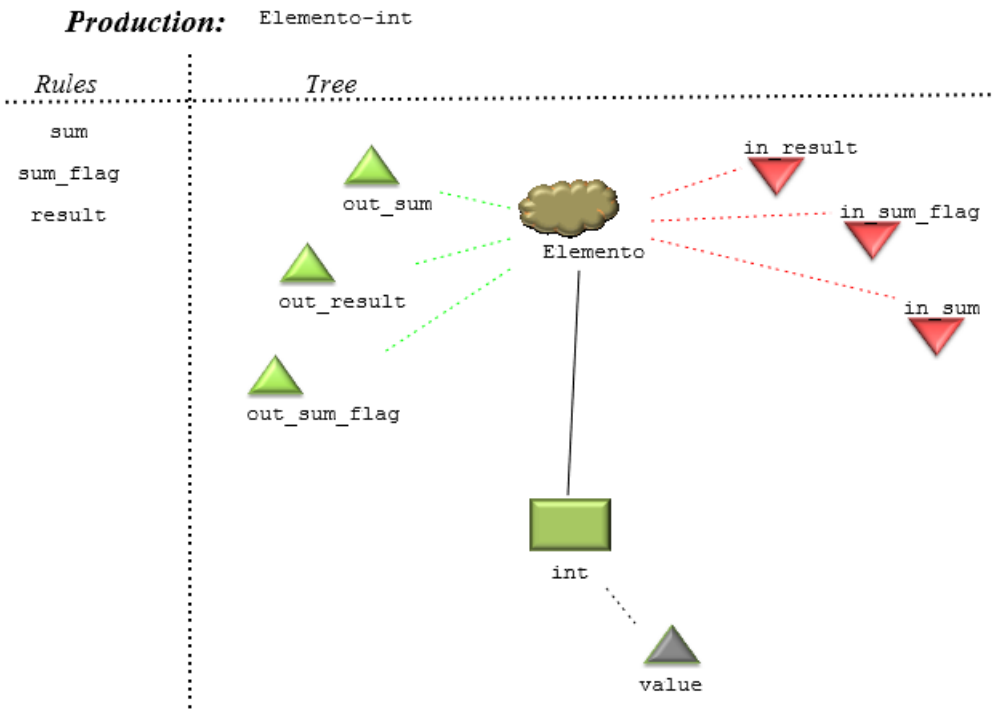

Figura 5: Produção P3

4.1.5 Elemento -> str

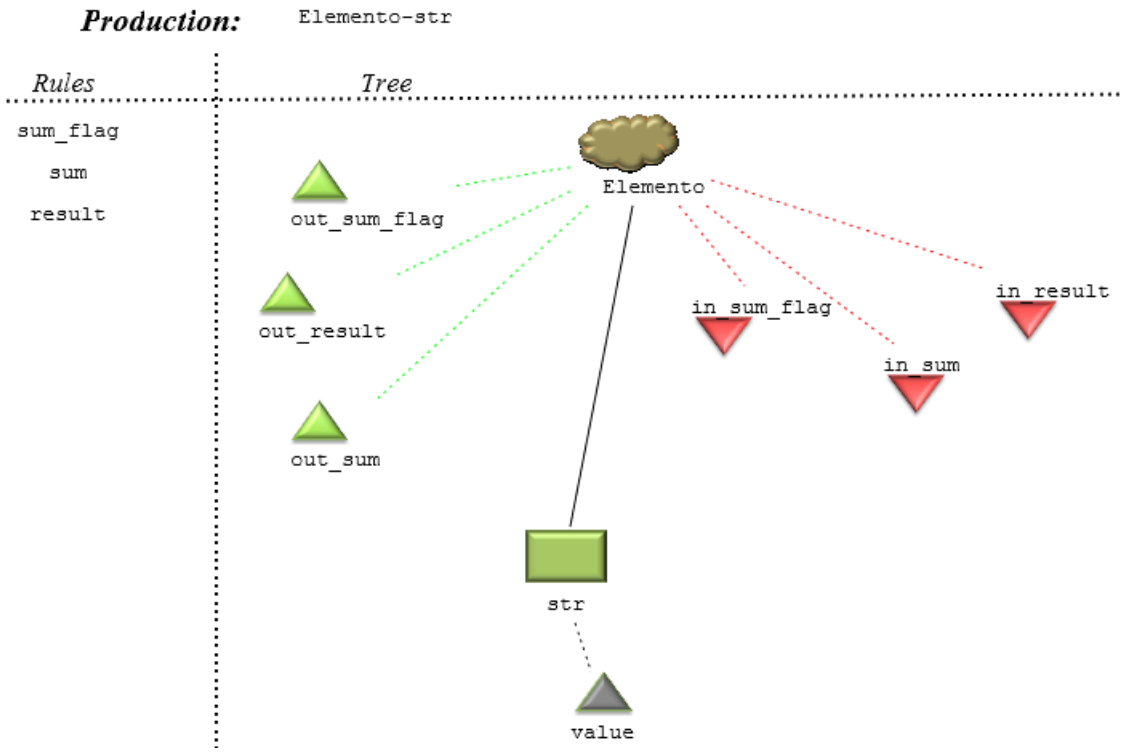

Figura 6: Produção P4

4.2 Regras

4.2.1 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`

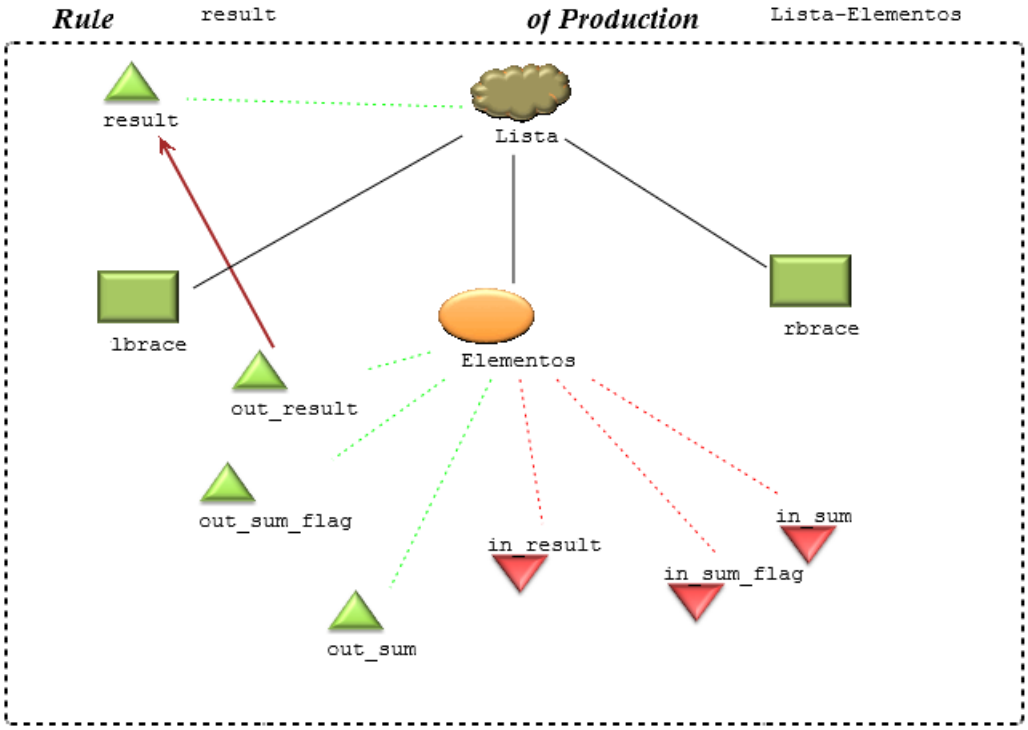

Figura 7: Regra para Lista.result

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
```

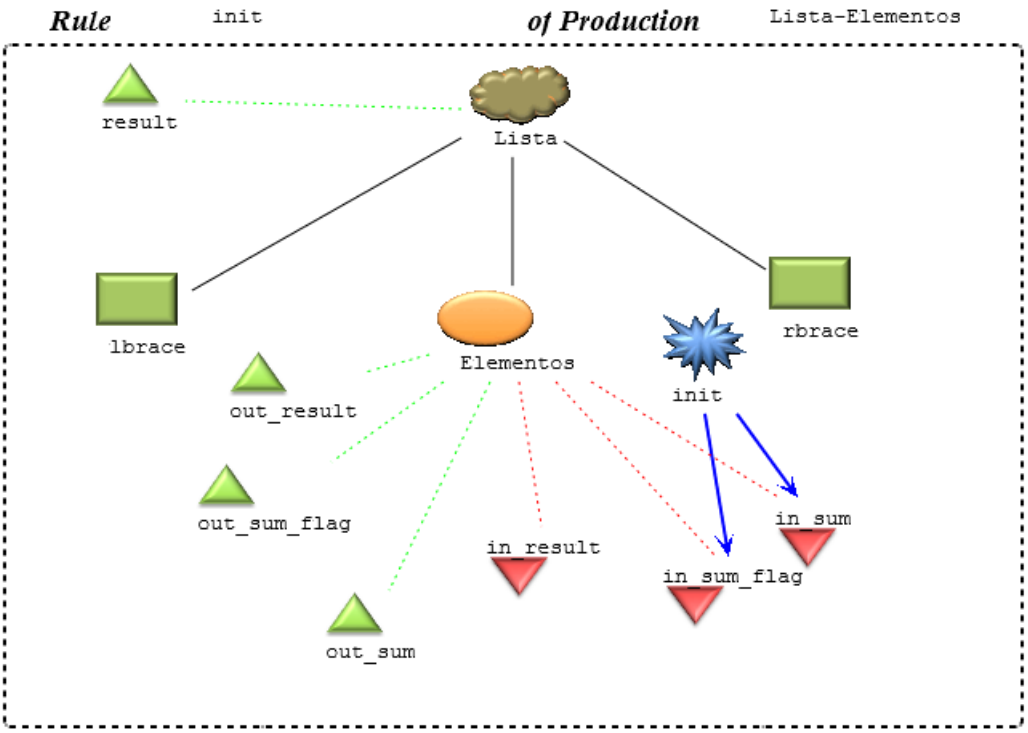

Figura 8: Regra para inicializar variáveis

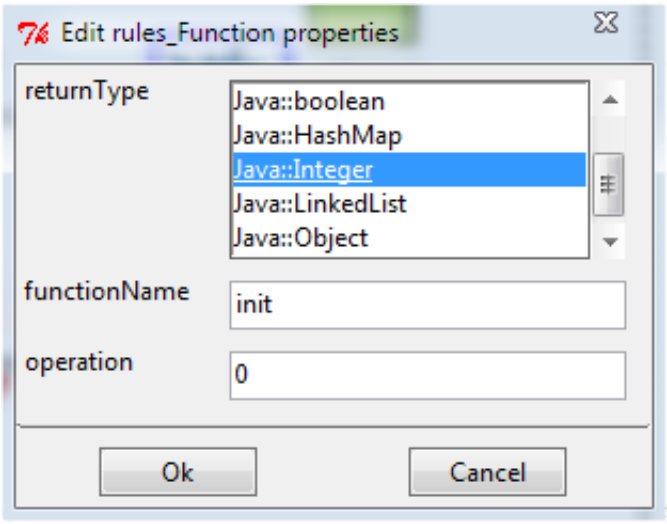

Figura 9: Função init

4.2.2 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
```

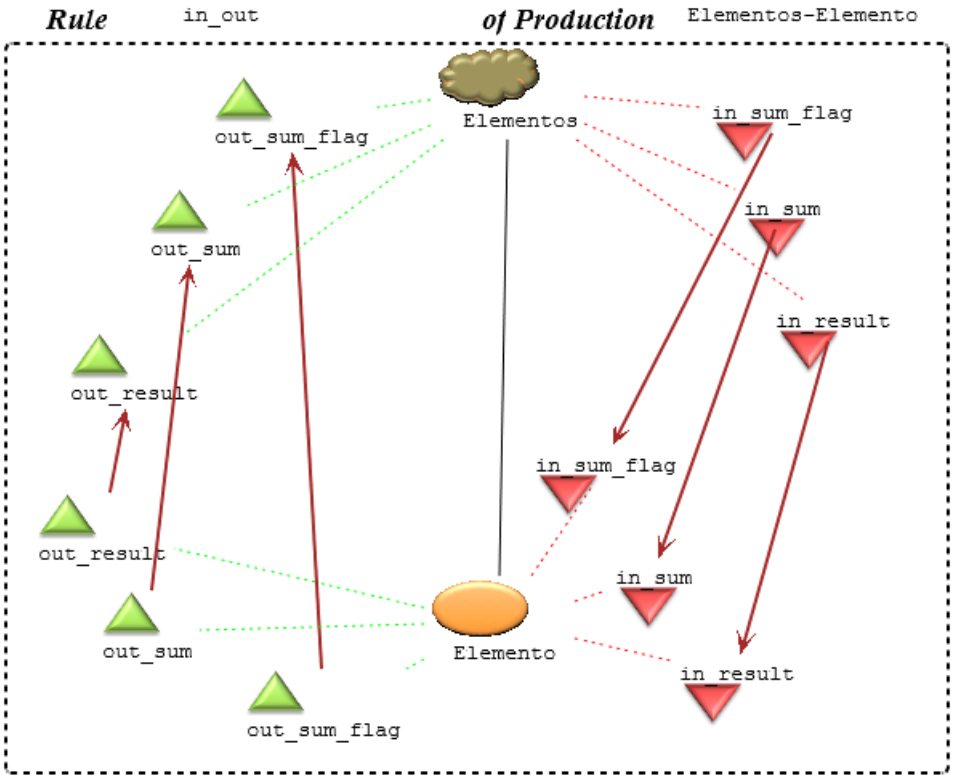

Figura 10: Regras in out

4.2.3 Elementos -> Elemento ',' Elementos

in
Regras:

```
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
```

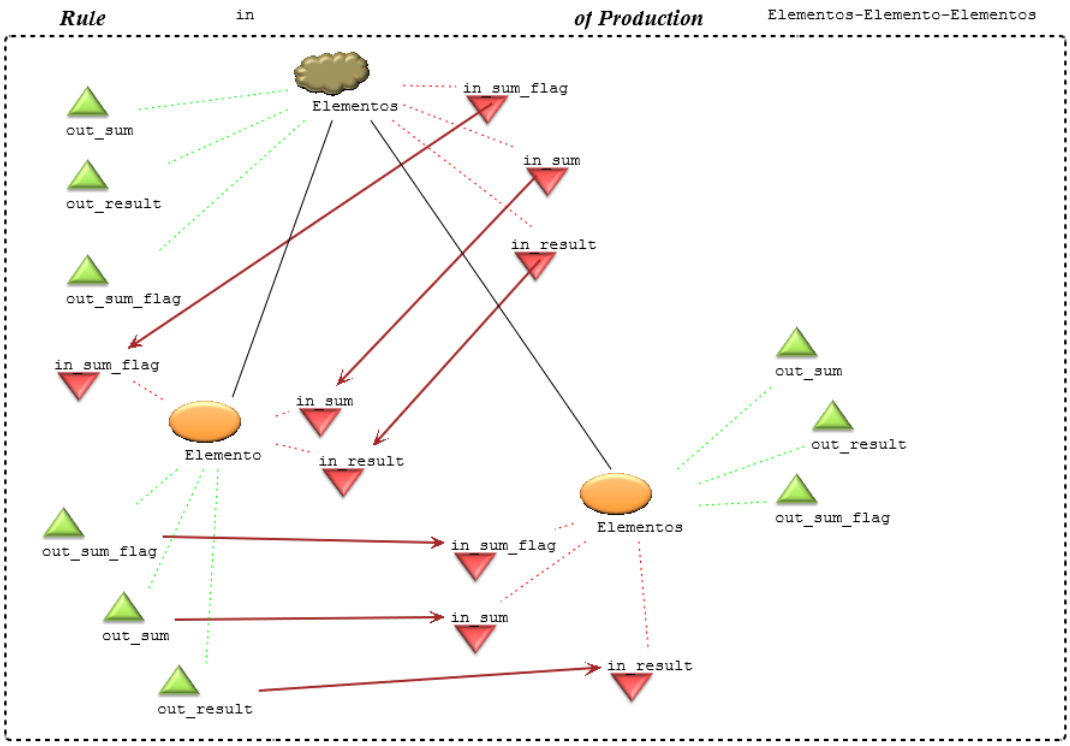

Figura 11: Regras in

out

```
Elementos0.out_sum = Elementos1.out_sum
Elementos0.out_sum_flag = Elementos1.out_sum_flag
Elementos0.out_result = Elementos1.out_result
```

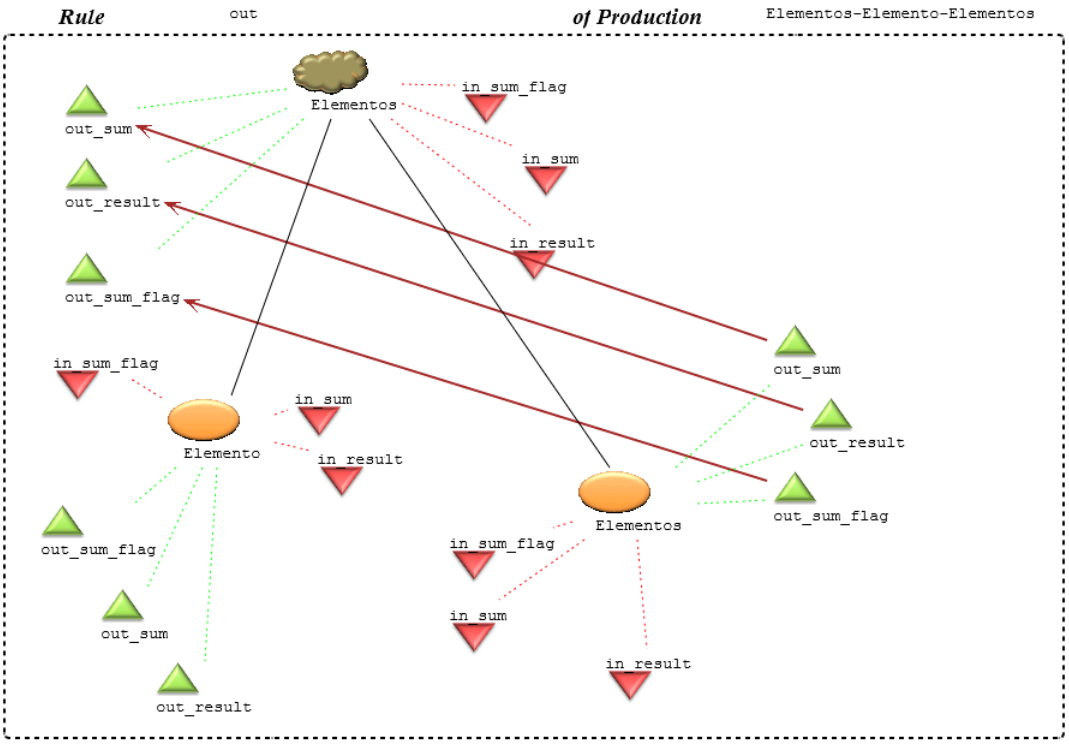

Figura 12: Regras out

4.2.4 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;
}
```

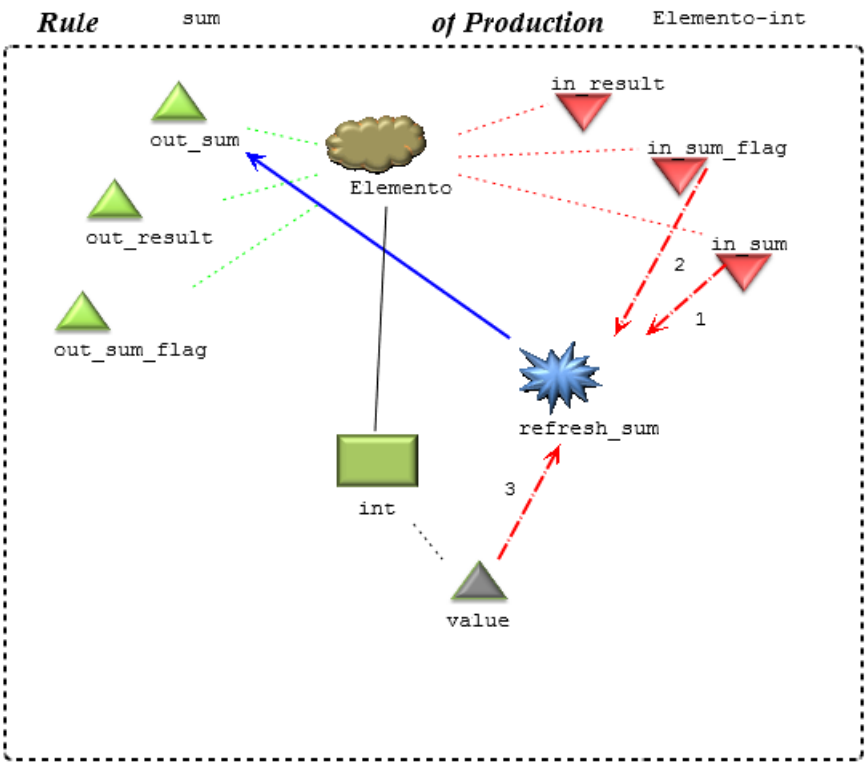

Figura 13: Regras sum

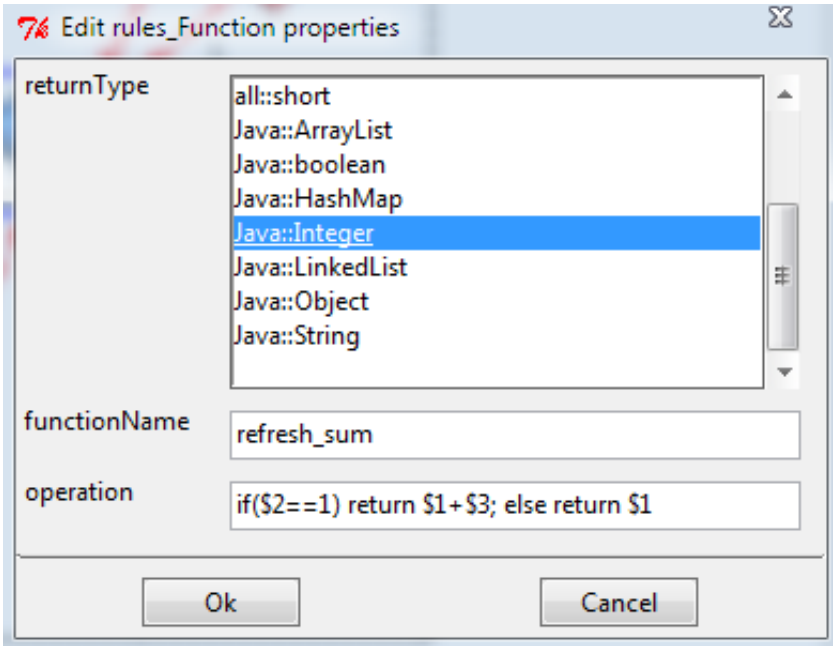

Figura 14: Função refresh sum

sum flag

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

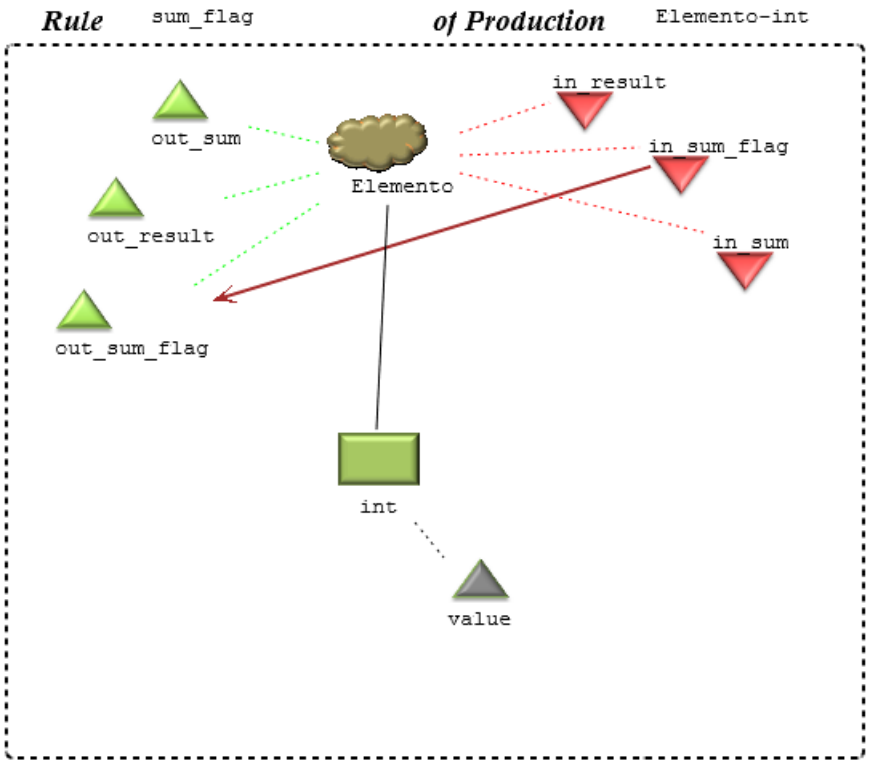

Figura 15: Regra sum flag

result

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

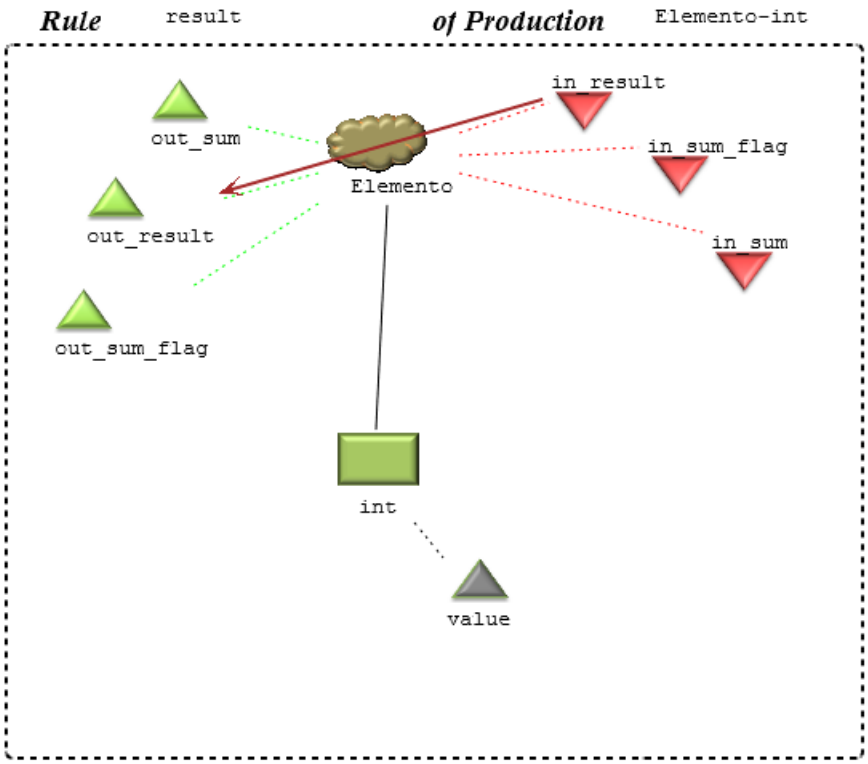

Figura 16: Regra result

4.2.5 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;
}
```

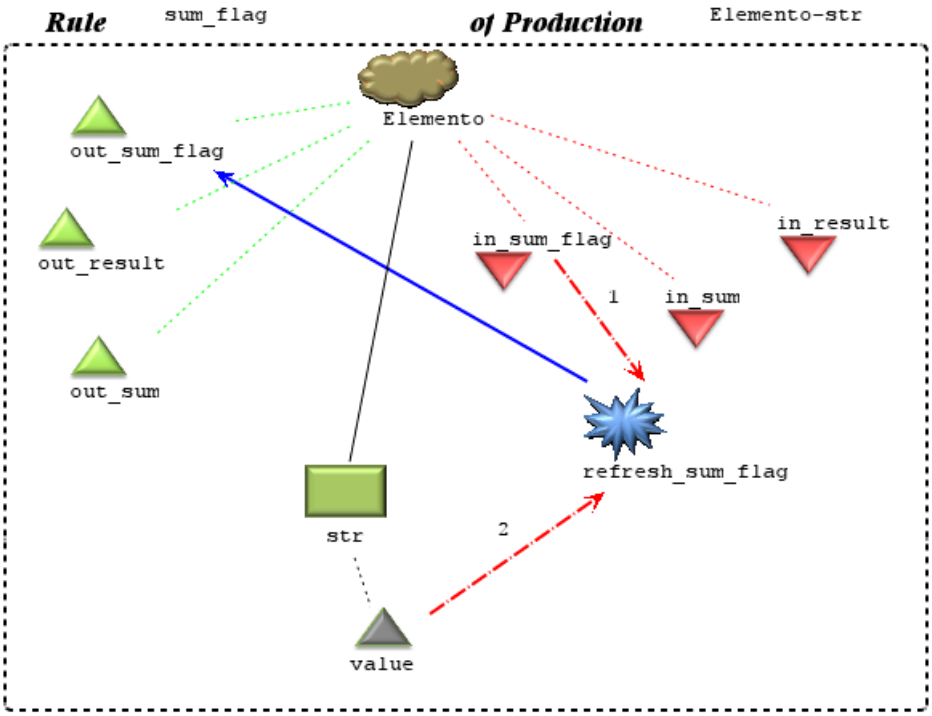

Figura 17: Regra sum flag

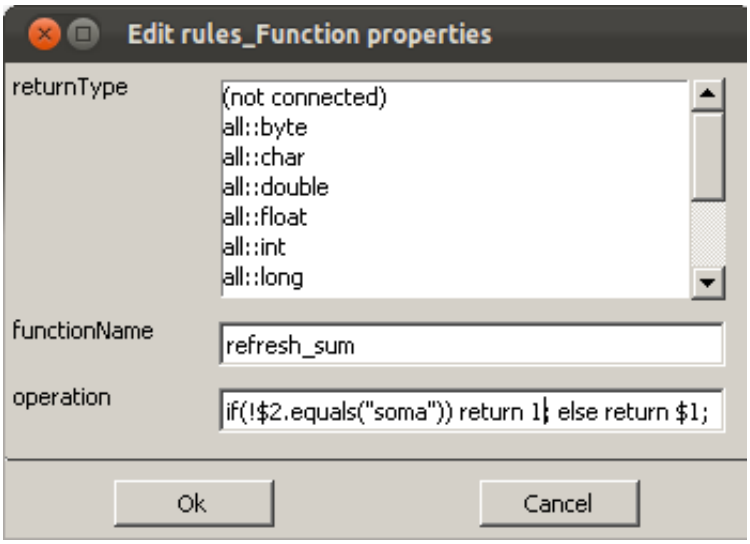

Figura 18: Função refresh sum flag

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;
}
```

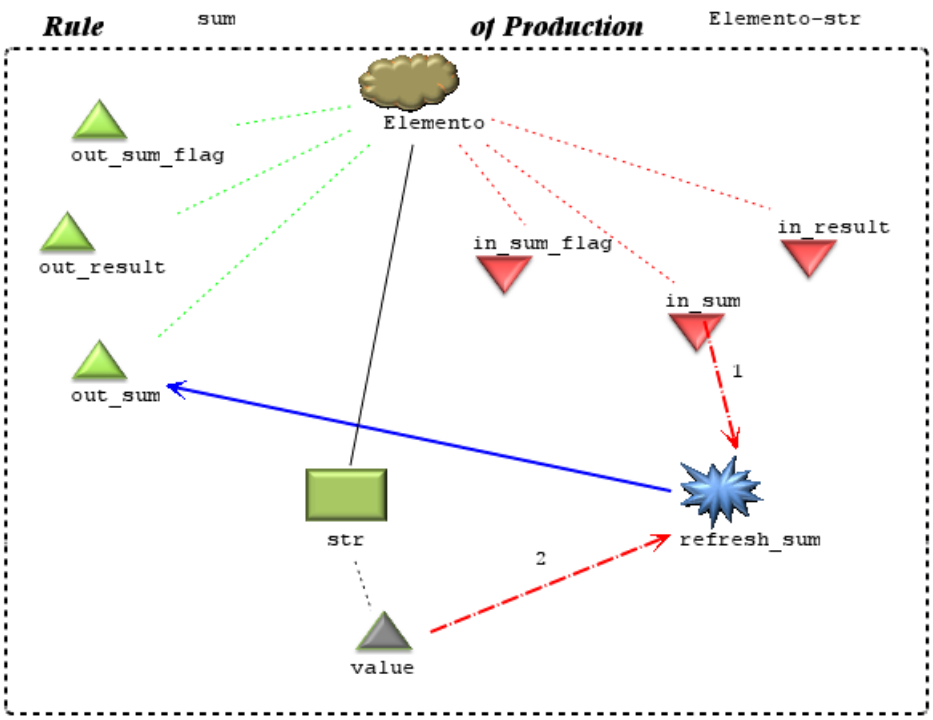

Figura 19: Regra sum

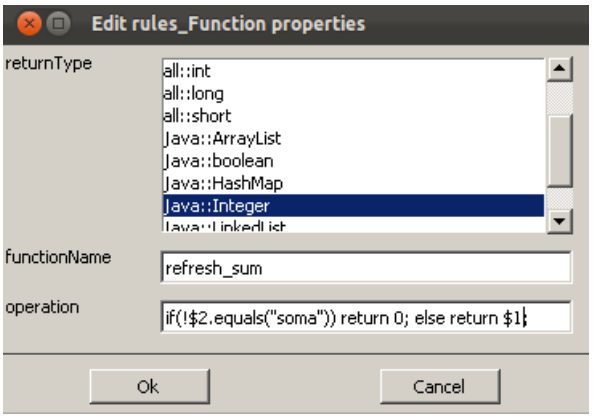

Figura 20: Função refresh sum

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;
}
```

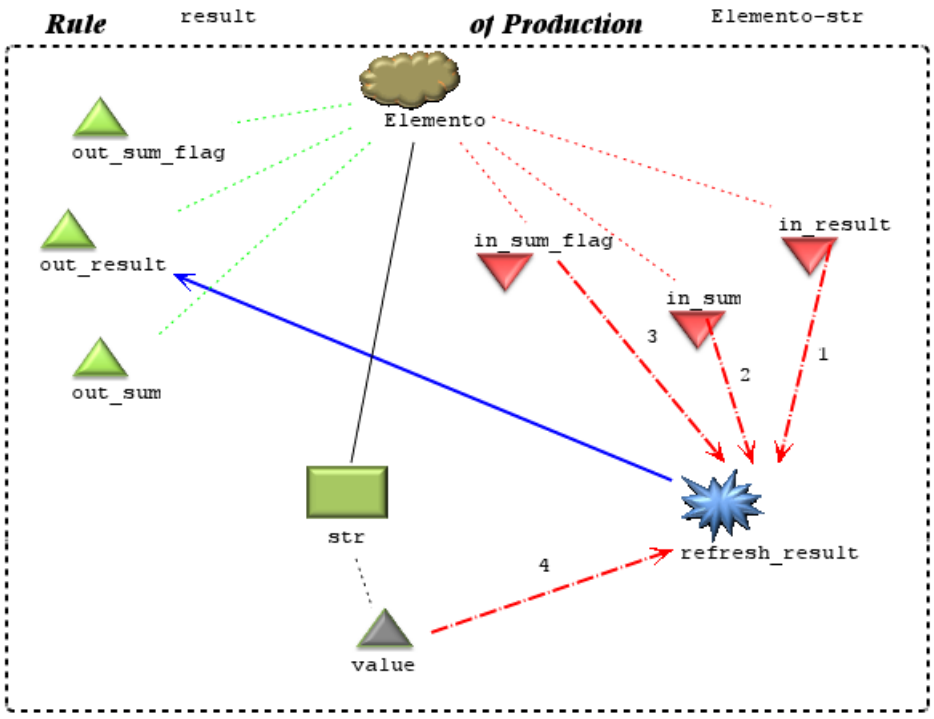

Figura 21: Regra result

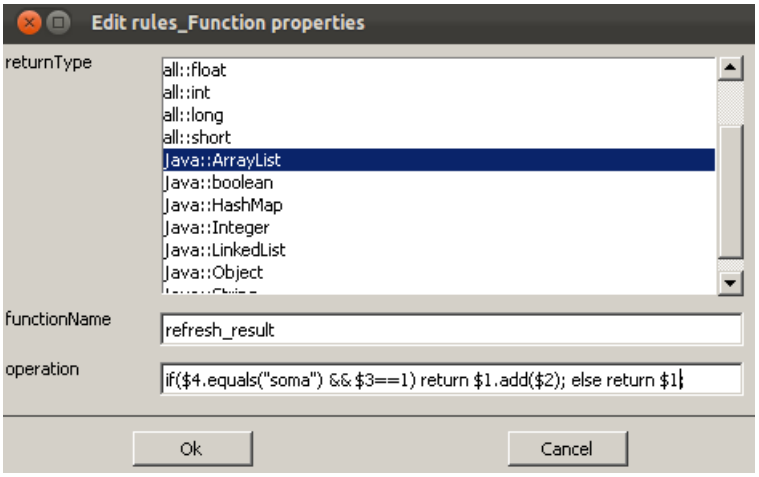

Figura 22: Função refresh result

5 Conclusões

A resolução deste exercício permitiu perceber melhor a forma como as linguagens de estrutura para a resolução de determinados problemas. Depois de definida a GIC e criando a GA, conseguimos realizar os cálculos que eram pretendidos para a soma. Serviu de consolidação da matéria dada até agora no módulo de Engenharia de Linguagens, tendo em conta que conseguimos resolver o exercício com sucesso.