Exercício para Avaliação n.º 1 e n.º2

Bruno Azevedo*and Miguel Costa†

Módulo Engenharia Gramatical,
UCE30 Engenharia de Linguagens,
Mestrado em Engenharia Informátical,
Universidade do Minho

8 de Dezembro de 2011

Resumo

Este documento apresenta as resoluções dos Exercícios Práticos n.º 1 e n.º2 do módulo de Engenharia de Linguagens. Os exercícios estão relacionados com Gramática Independente de Contexto e Gramática de Atributos para resolver um problema de cálculo de elementos mistos (palavras e números).

*Email: azevedo.252@gmail.com
†Email: miguelpintodacosta@gmail.com

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 Ex1 - 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 de 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 Ex1 - 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}
Símbolos 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)`. Uma nota importante, é que o valor de `sum` só é adicionado se este for maior que 0.

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 Ex1 - Resolução no VisualLisa

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

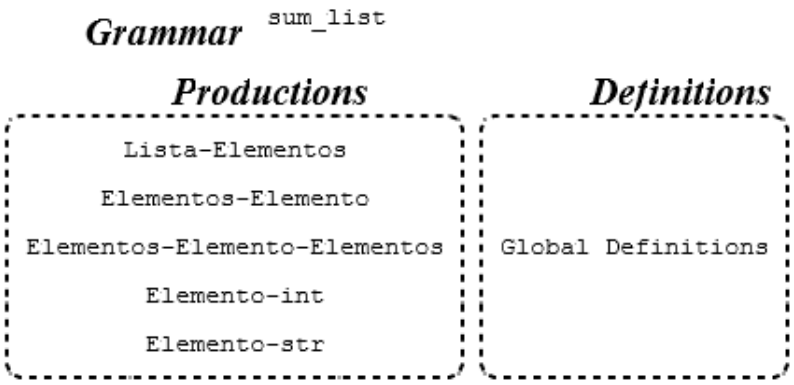

Figura 1: Produções

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.

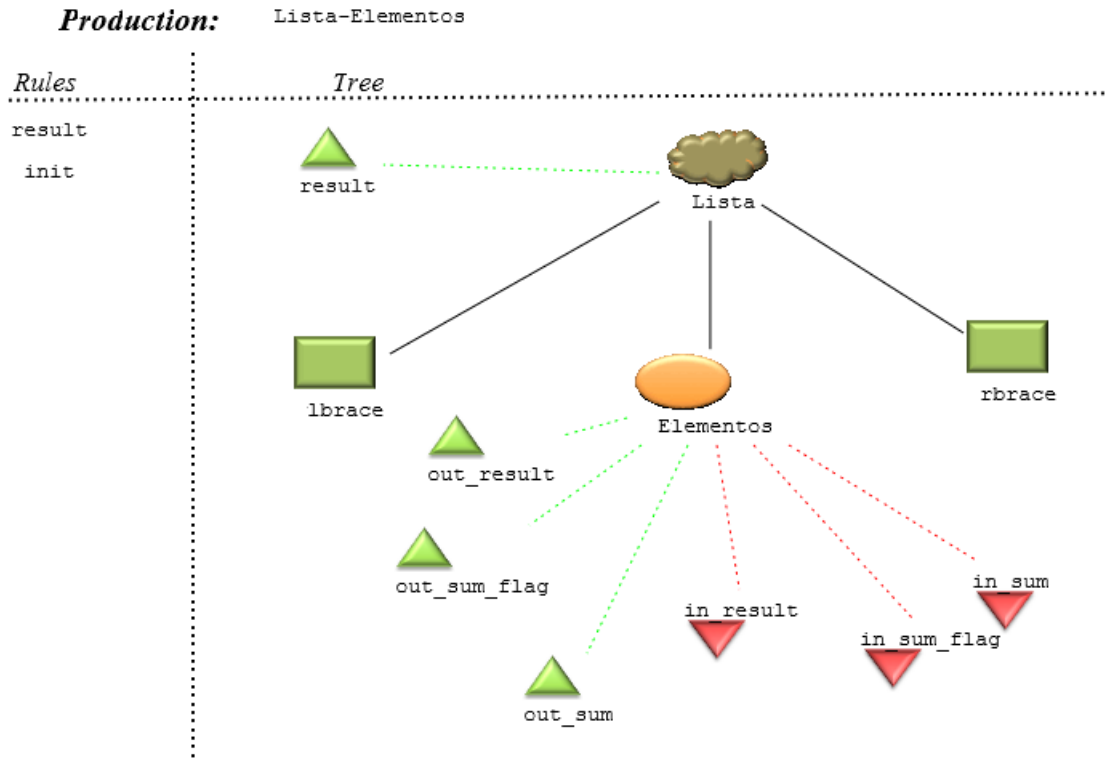

Figura 2: Produção P0

4.1.2 Elementos -> Elemento

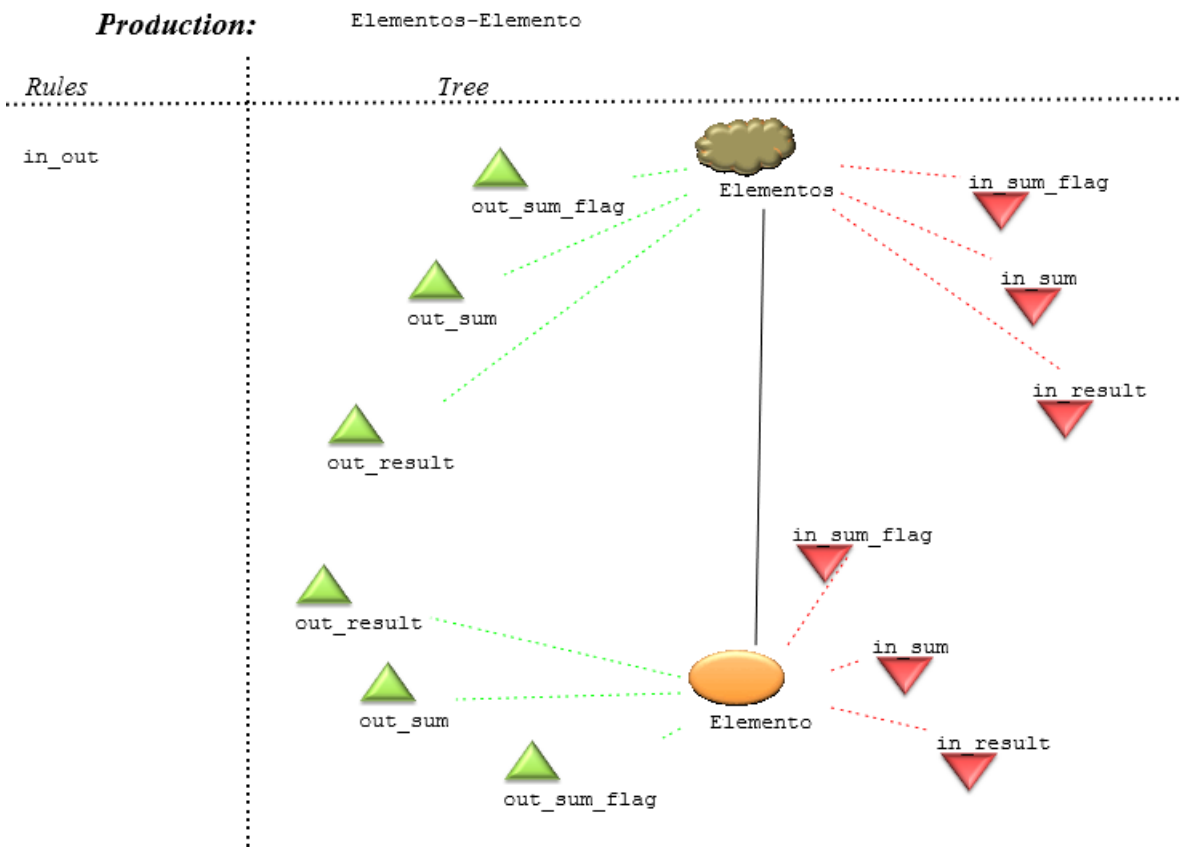

Figura 3: Produção P1

4.1.3 Elementos -> Elemento ',' Elementos

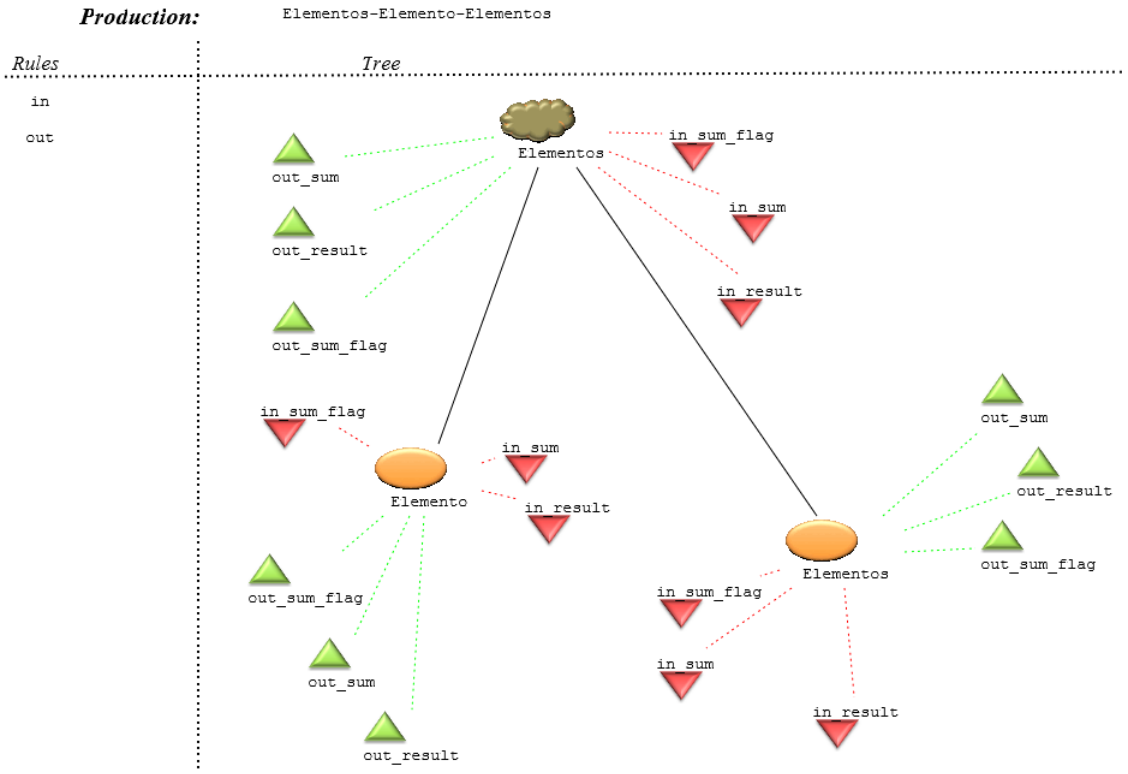

Figura 4: Produção P2

4.1.4 Elemento -> int

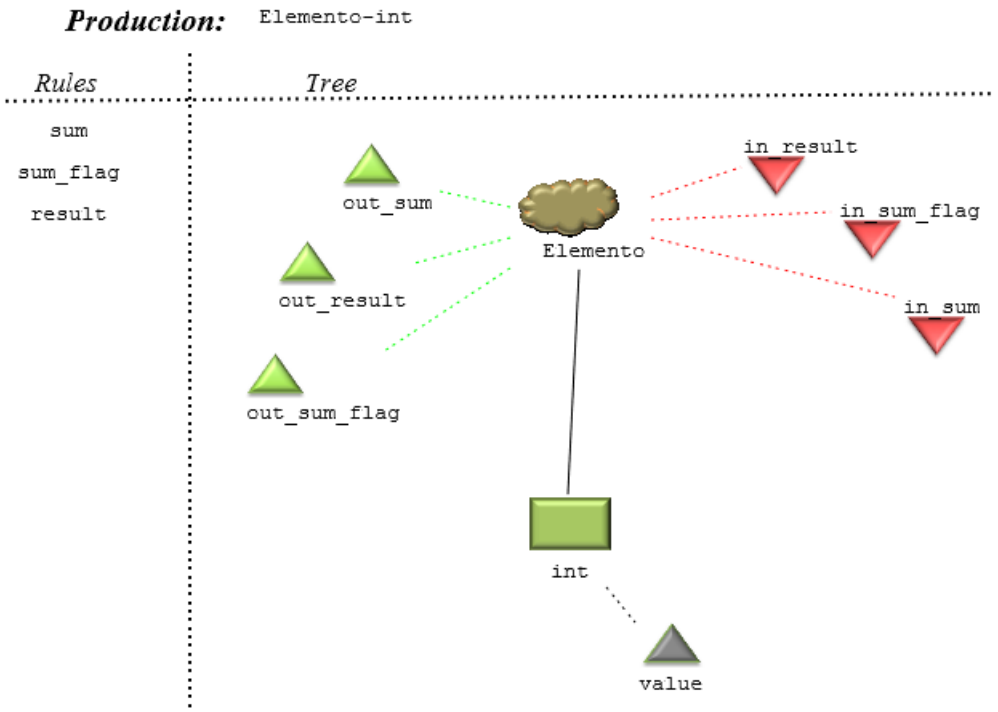

Figura 5: Produção P3

4.1.5 Elemento -> str

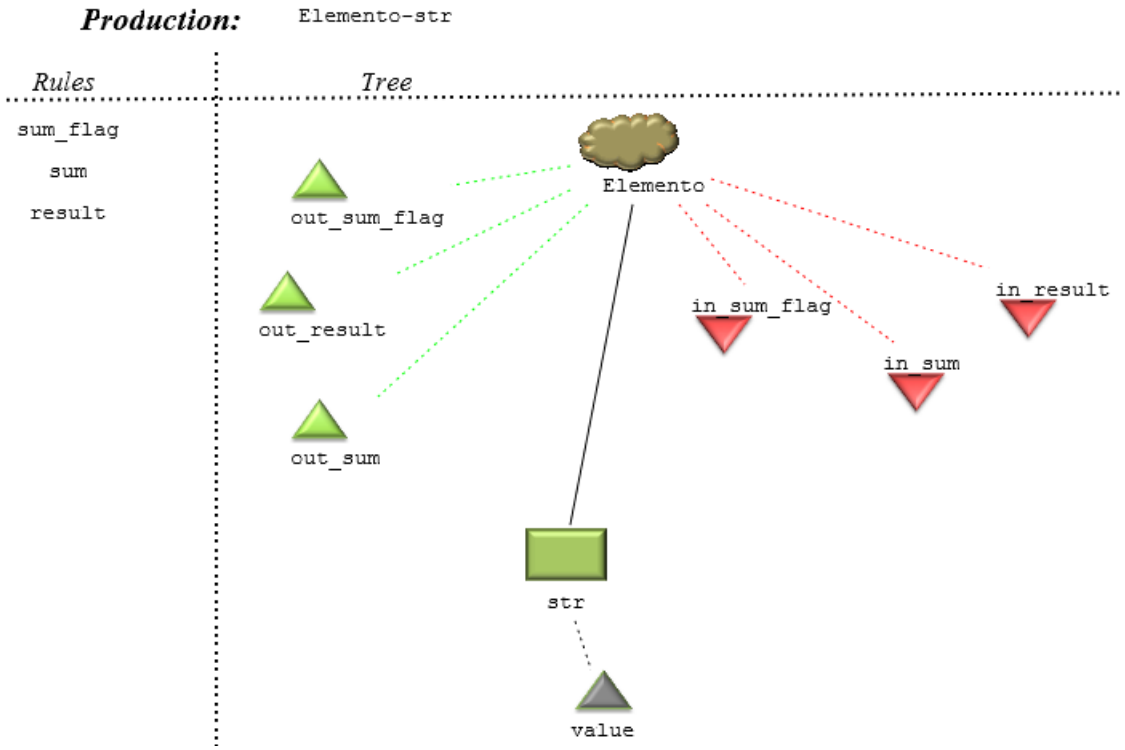

Figura 6: Produção P4

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`

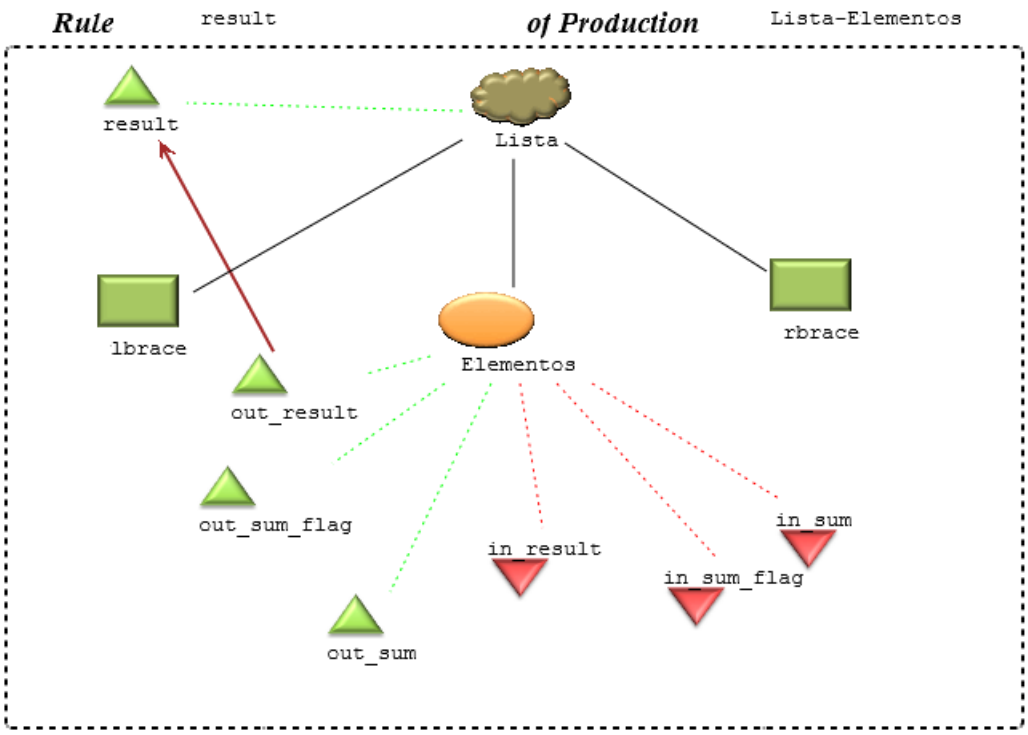

Figura 7: Regra para Lista.result

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
```

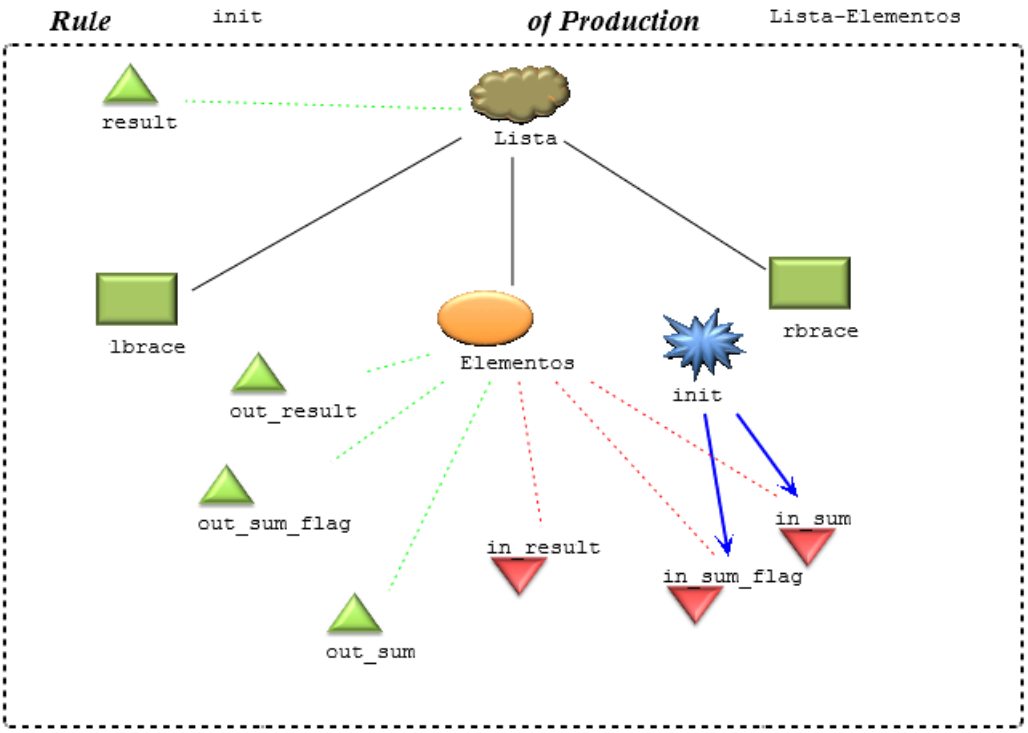

Figura 8: Regra para inicializar variáveis

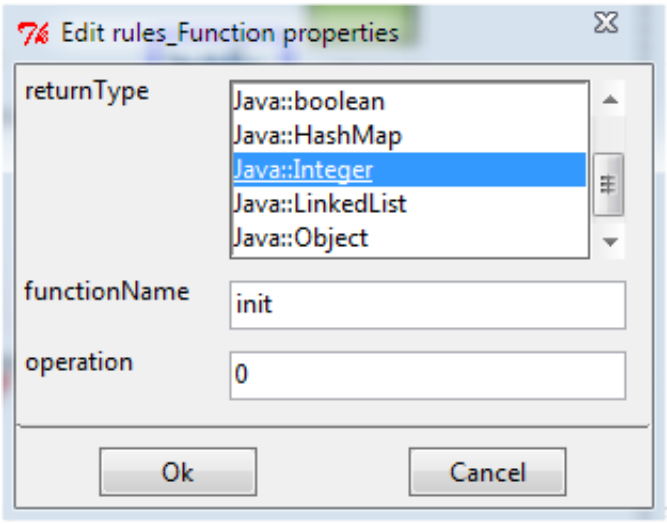

Figura 9: Função init

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
```

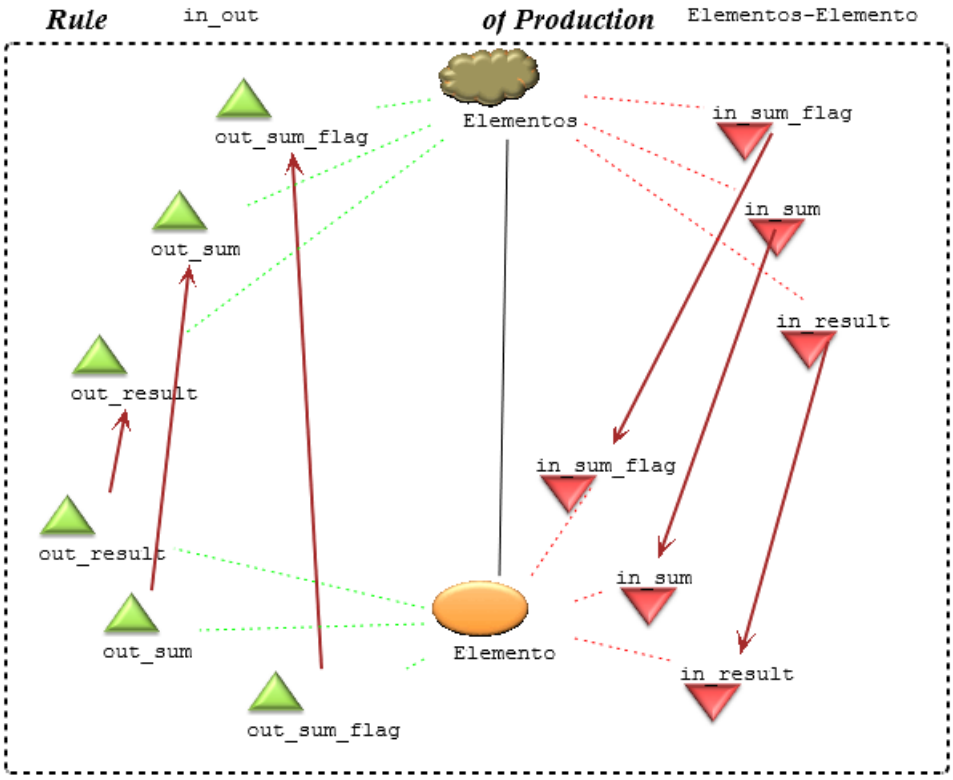

Figura 10: Regras in out

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
```

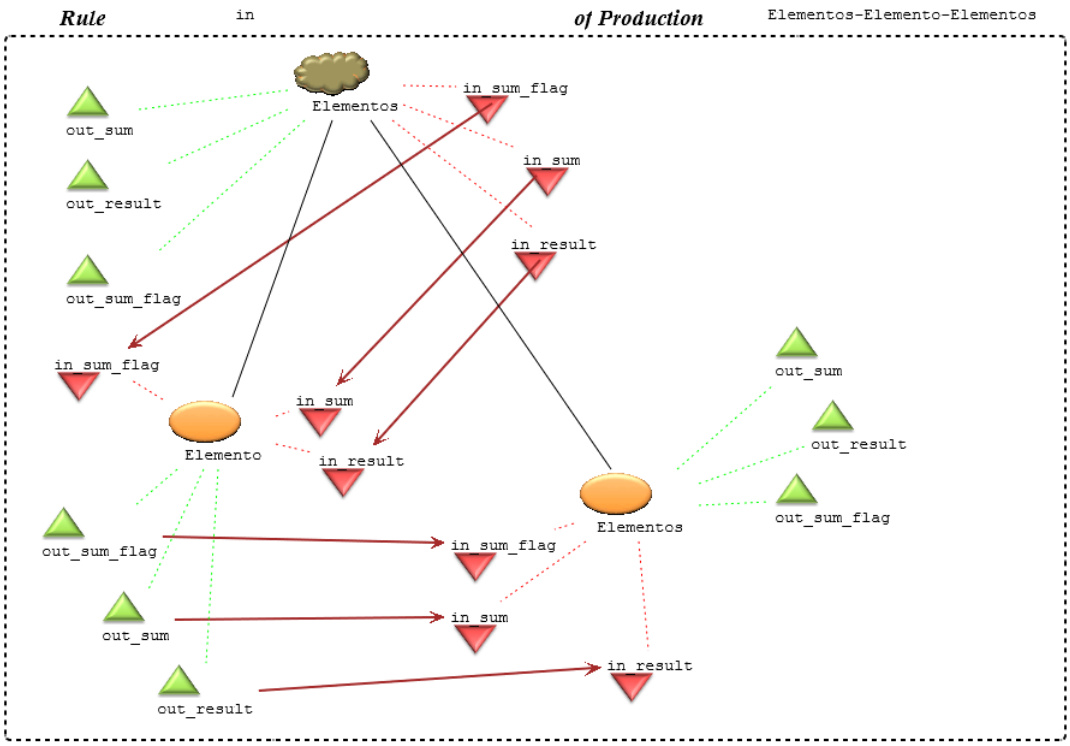

Figura 11: Regras in

out

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

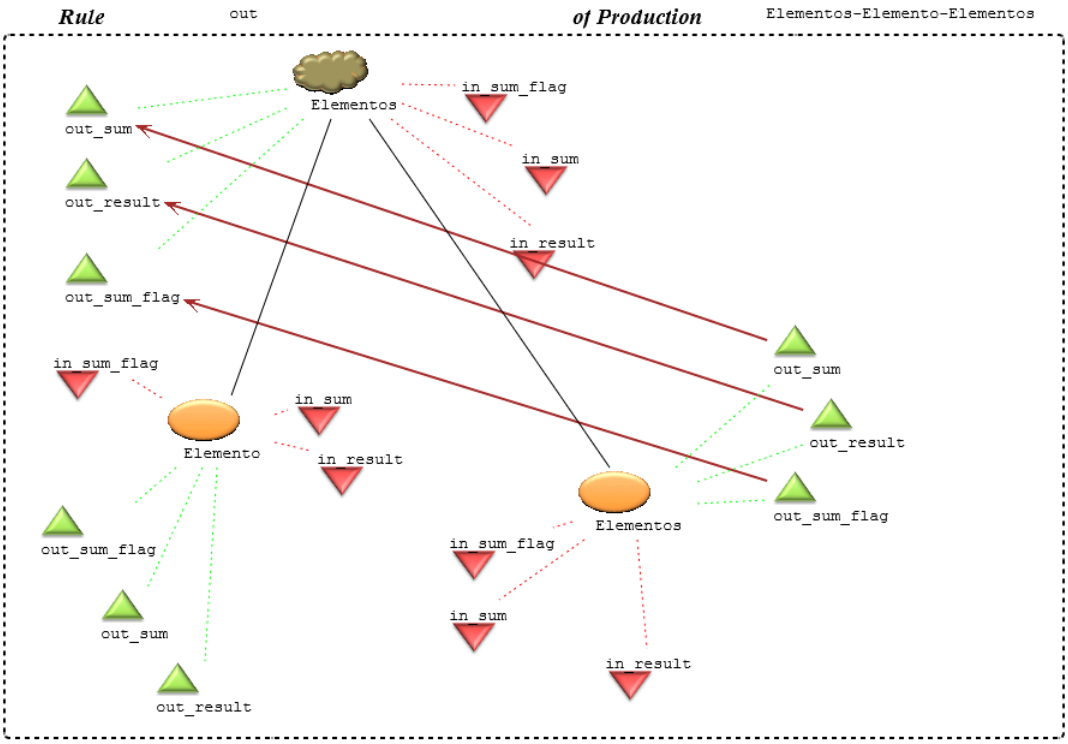

Figura 12: Regras out

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;
}
```

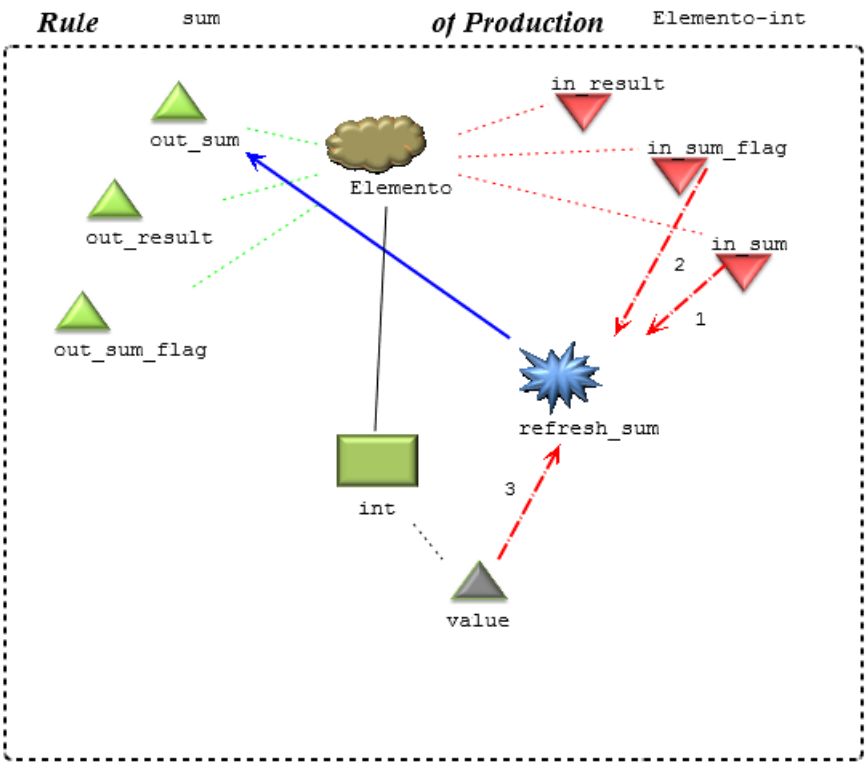

Figura 13: Regras sum

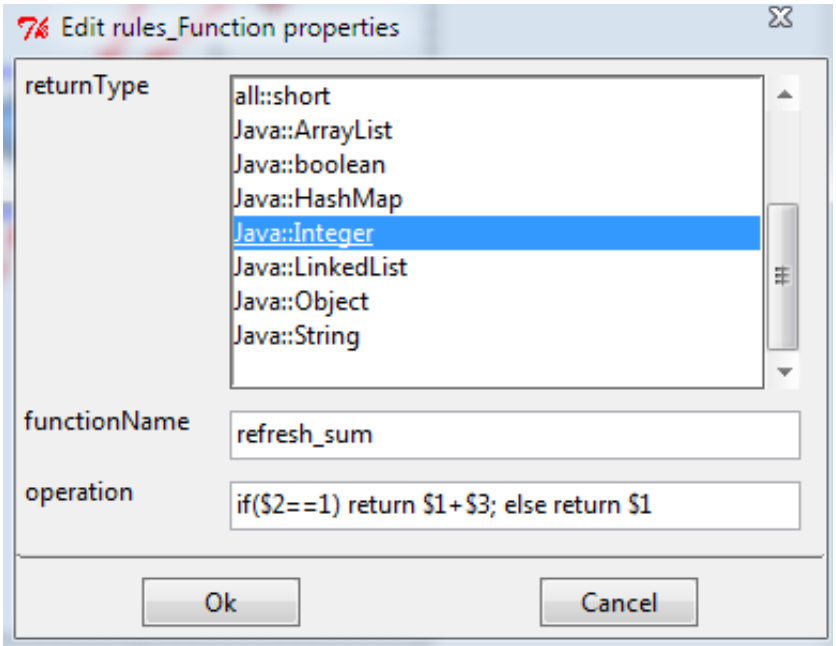

Figura 14: Função refresh sum

sum flag

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

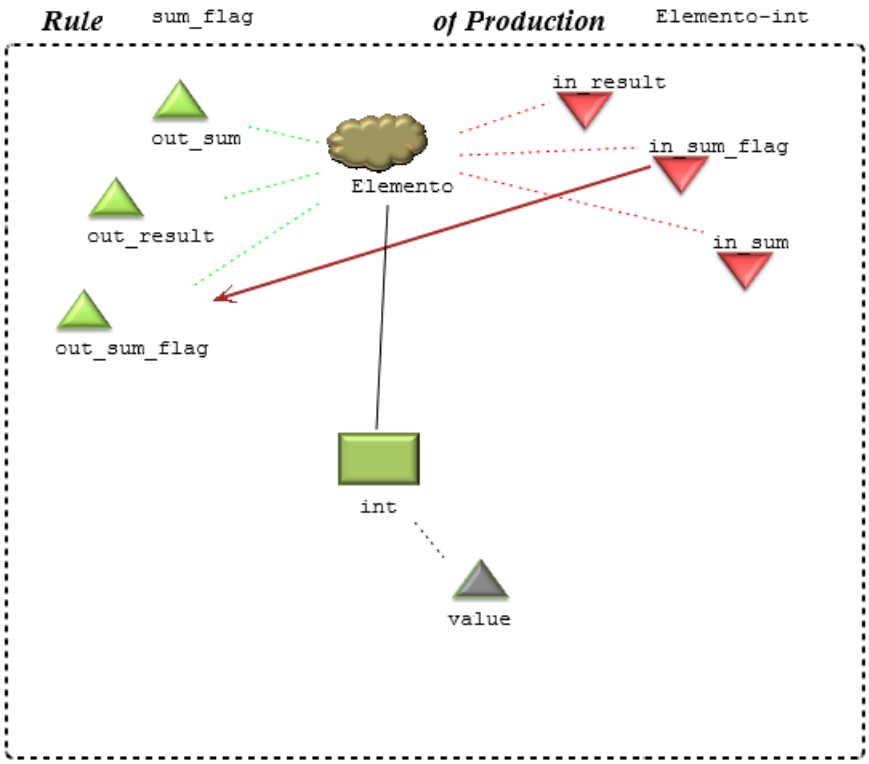

Figura 15: Regra sum flag

result

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

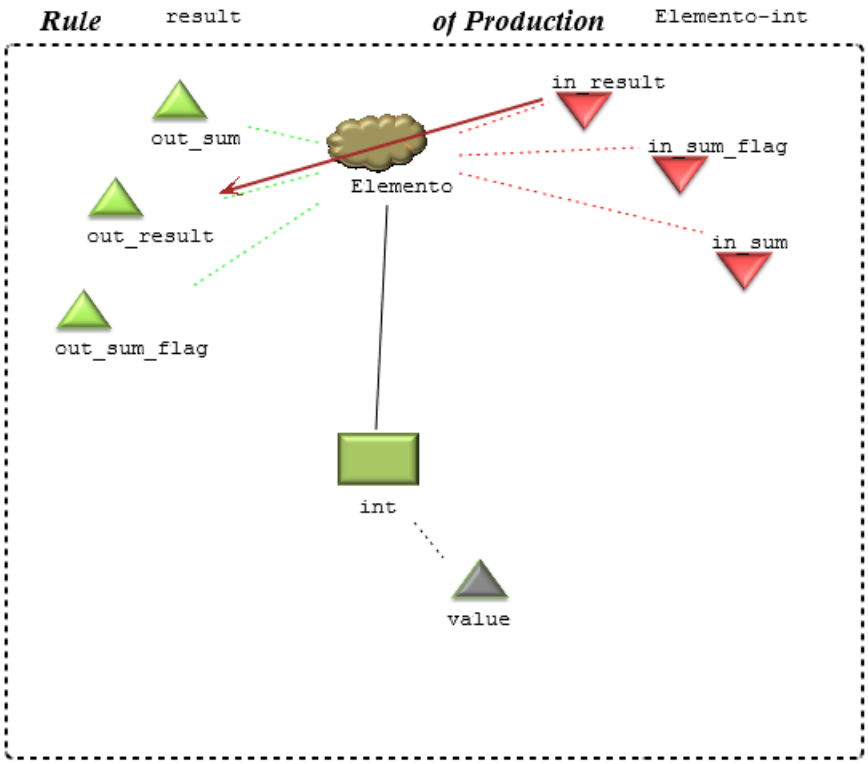

Figura 16: Regra result

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;
}
```

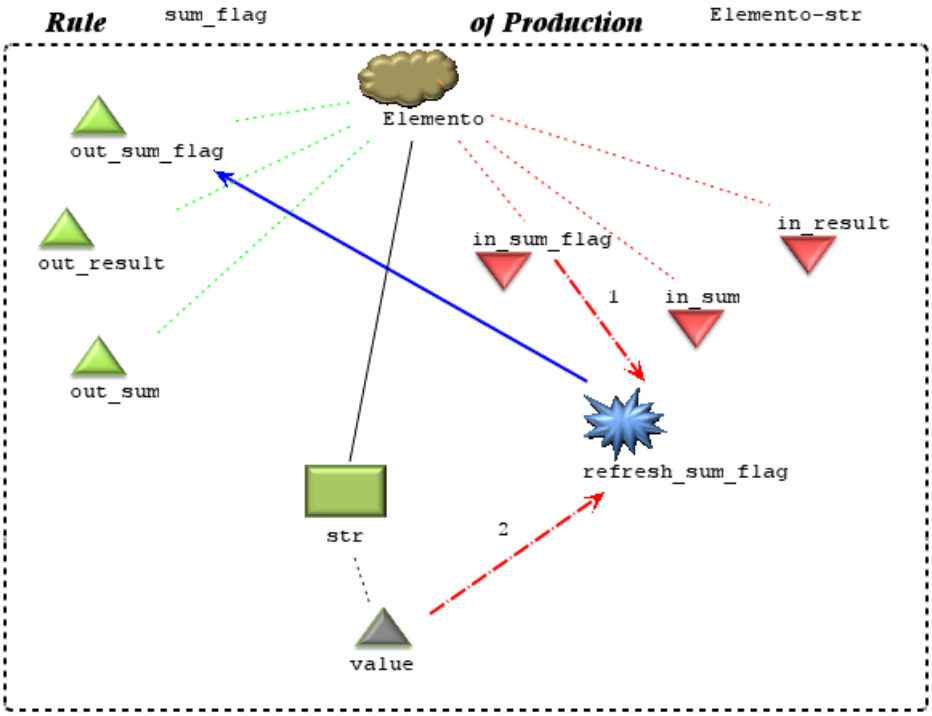

Figura 17: Regra sum flag

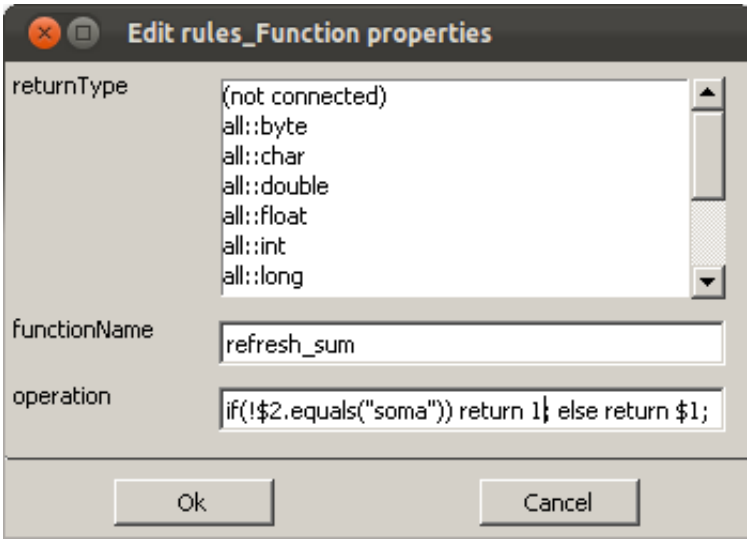

Figura 18: Função refresh sum flag

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;
}
```

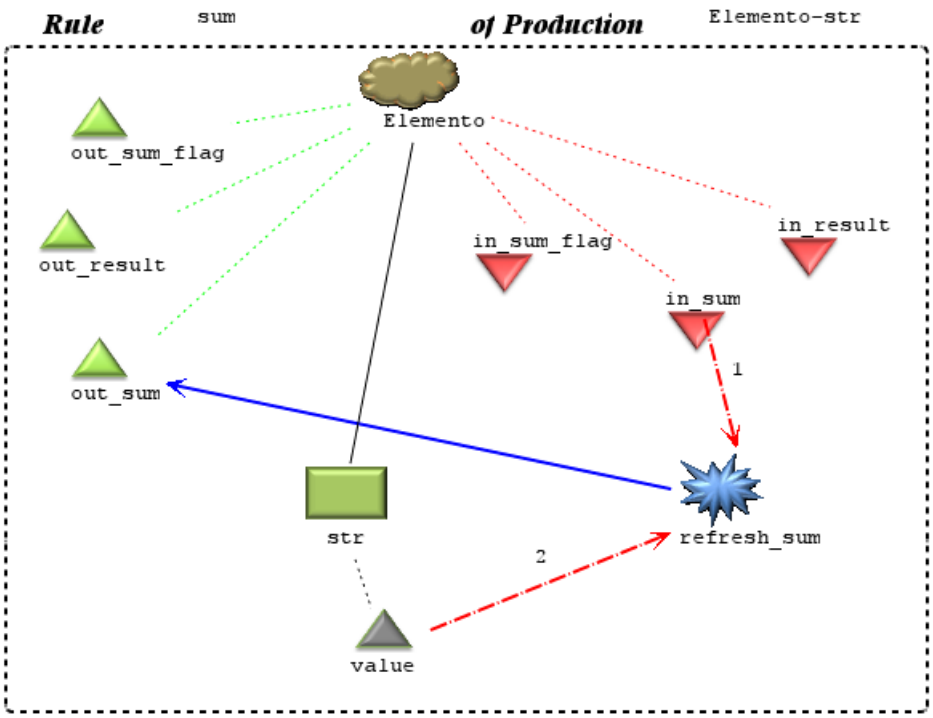

Figura 19: Regra sum

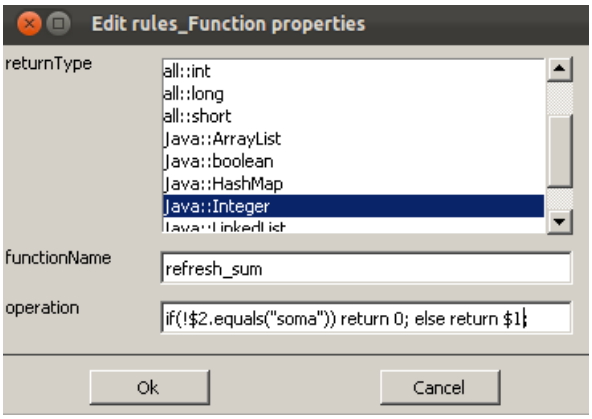

Figura 20: Função refresh sum

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;
}
```

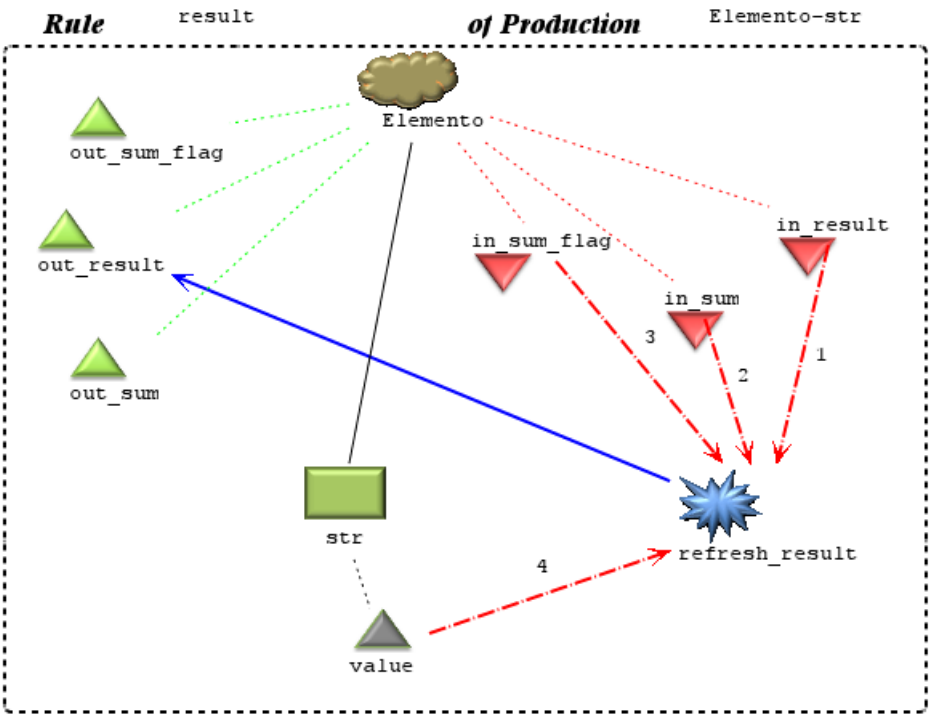

Figura 21: Regra result

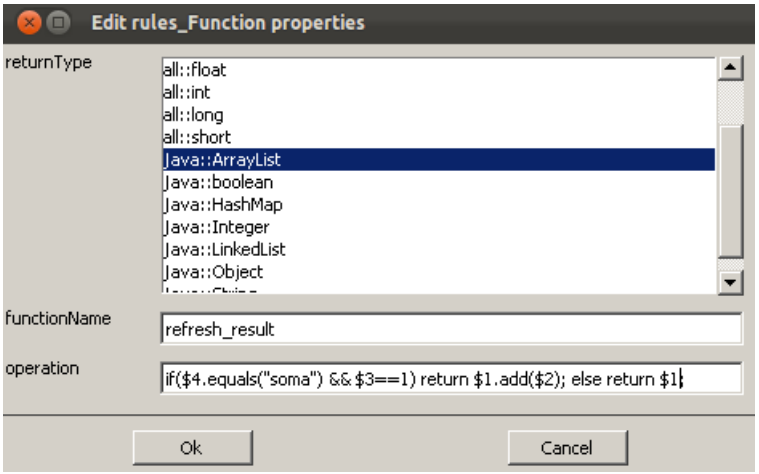

Figura 22: Função refresh result

5 Ex2 - Descrição do Problema

Era pretendido que se usasse o processador da Lista de Elementos Mistos (letras e inteiros), e criar uma Gramática de Atributos (GA) de modo a calcular o somatório de cada sequência de inteiros que surjam a seguir à sequência de 3 ou mais letras.

Exemplo:

```
A frase "[a b 1 1 1 1 c d e f 2 2 g h 3 3 3 i j k l m 4 4 4 4]"
Dá como resultado: [4,16]
```

6 Ex2 - Resolução no papel

6.1 Gramática Independente do Contexto

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

```
GIC = (T, N, S, P)
Símbolos terminais (T):         {char, int, '[' , ']', ','}
Símbolos não terminais (N):     {Lista, Elementos, Elemento}
Símbolos Inicial (S):           Lista
Produções (P):
                                P0: Lista       -> '[' Elementos ']'
                                P1: Elementos   -> Elemento
                                P2:             | Elementos ',' Elemento
                                P3: Elemento    -> int
                                P4:             | char

                                char = [a-zA-Z]
                                int = [0-9]+
```

6.2 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`
- `back`
- `seq_char`
- `result`

A variável `back` é inicializada a -1, é ela que nos diz se o elemento anterior foi uma letra (`back=1`) ou um número(`back=0`). `Seq_char`, responsável por armazenar o número de caracteres que aparecem consecutivamente, se estiver maior ou igual a 3, quando foram encontrados inteiros serão somados à variável sum. Se for encontrado novamente um caracter, se o valor de sum for maior que 0 é adicionado a result e faz "reset" às outras variáveis.

`Result` é um array que vai conter os resultados das várias "somas" que vão surgindo, ele é alterado quando se encontra depois de uma sequencia de 3 ou mais letras vários dígitos, em que é adicionado a soma desses dígitos caso esta seja maior que 0, vai ficar então: 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_back`
  - `out_seq_char`
  - `out_result`
- Atributos herdados
  - `in_sum`
  - `in_back`
  - `in_seq_char`
  - `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).

6.2.1 Atributos (A)

```
Lista           result : ArrayList<Integer>

Elementos       in_result : ArrayList<Integer>
                out_result : ArrayList<Integer>
                in_sum : int
                out_sum : int
                in_back : int
                out_back :int
                in_seq_char : int
                out_seq_char : int


Elemento        in_result : ArrayList<Integer>
                out_result : ArrayList<Integer>
                in_sum : int
                out_sum : int
                in_back : int
                out_back :int
                in_seq_char : int
                out_seq_char : int
```

6.2.2 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_back = -1
        Elementos.in_seq_char = 0;

P1: Elementos -> Elemento
        Elemento.in_result = Elementos.in_result
        Elemento.in_sum = Elementos.in_sum
        Elemento.in_back = Elementos.in_back
        Elemento.in_seq_char = Elementos.in_seq_char
        Elementos.out_result = Elemento.out_result
```

```
        Elementos.out_sum = Elemento.out_sum
        Elementos.out_back = Elemento.out_back
        Elementos.out_seq_char = Elemento.out_seq_char

P2: Elementos0 -> Elementos1 ',' Elemento
        Elementos0.out_sum = Elemento.out_sum
        Elementos0.out_back = Elemento.out_back
        Elementos0.out_result = Elemento.out_result
        Elementos0.out_seq_char = Elemento.out_seq_char
        Elemento.in_sum = Elementos1.out_sum
        Elemento.in_back = Elementos1.out_back
        Elemento.in_result = Elementos1.out_result
        Elemento.in_seq_char = Elementos1.out_seq_char
        Elementos1.in_sum = Elementos0.in_sum
        Elementos1.in_back = Elementos0.in_back
        Elementos1.in_result = Elementos0.in_result
        Elementos1.in_seq_char = Elementos0.in_seq_char

P3: Elemento -> int
        Elemento.out_result = Elemento.in_result
        Elemento.out_sum = function refresh_sum
        Elemento.out_back = 0
        Elemento.out_seq_char = Elemento.in_seq_char

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

P4: Elemento -> str
        Elemento.out_result = function refresh_result
        Elemento.out_sum = 0
        Elemento.out_back = 1
        Elemento.out_seq_char = function refresh_seq_char

        $1 = Elemento.in_result,
        $2 = Elemento.in_sum,
        $3 = Elemento.in_back,
        ArrayList<Integer> refresh_result($1, $2, $3){
                if($3==0 && $2 > 0) return $1.add($2); else return $2;
        }

        $1 = Elemento.in_seq_char
        $2 = Elemento.in_back
        int refresh_seq_char($1,$2){
                if($2==1) return $1+1; else return 1;
        }
```

7 Ex2 - Resolução no VisualLisa

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

7.1 Produções

As `Produções (P)`:

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

da gramática independente de contexto que já está definida, quando representada visualmente em VisualLisa fica:

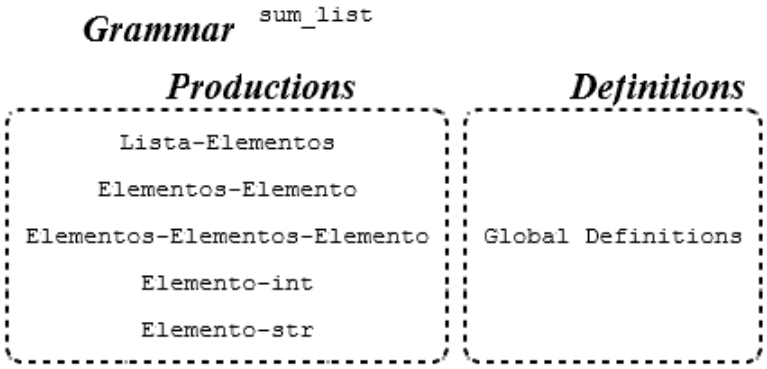

Figura 23: Produções

7.1.1 Lista -> Elementos

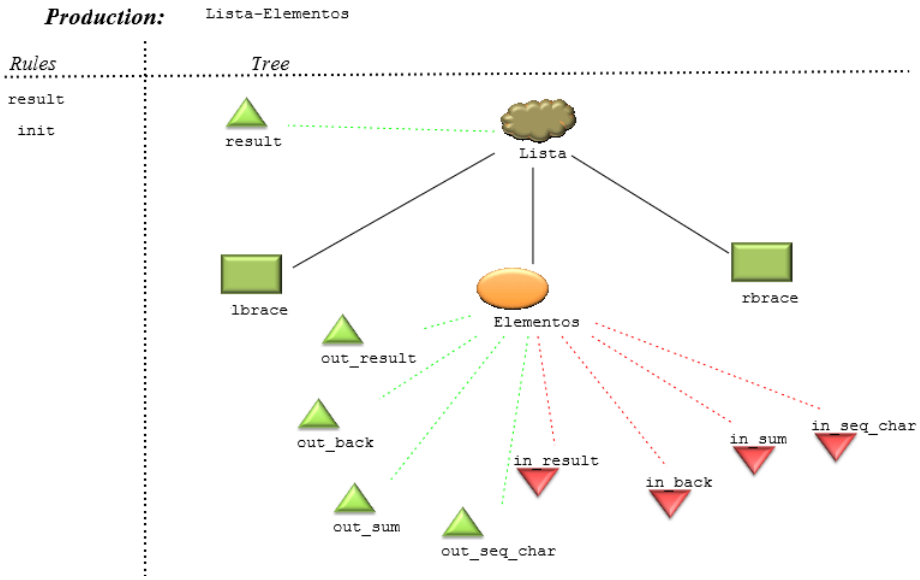

Figura 24: Produção P0

7.1.2 Elementos -> Elemento

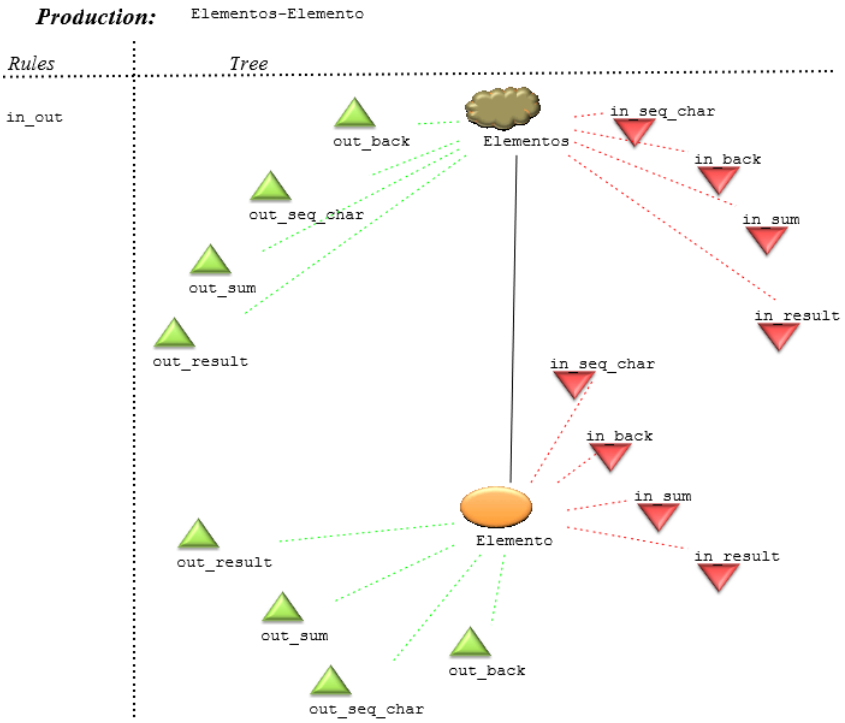

Figura 25: Produção P1

7.1.3 Elementos -> Elementos ',' Elemento

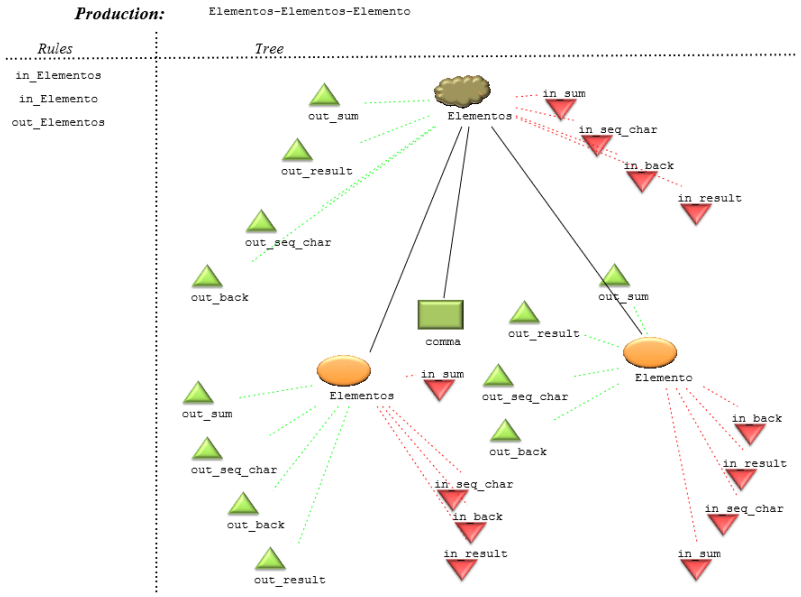

Figura 26: Produção P2

7.1.4 Elemento -> int

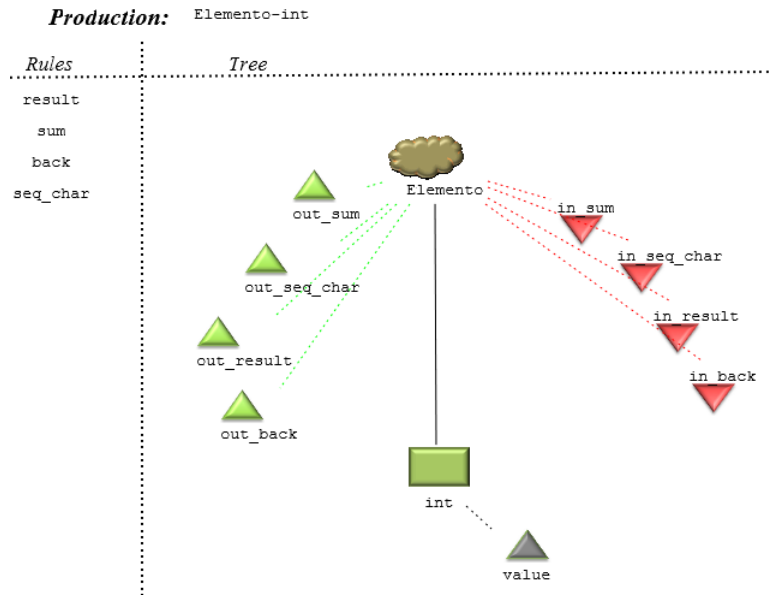

Figura 27: Produção P3

7.1.5 Elemento -> char

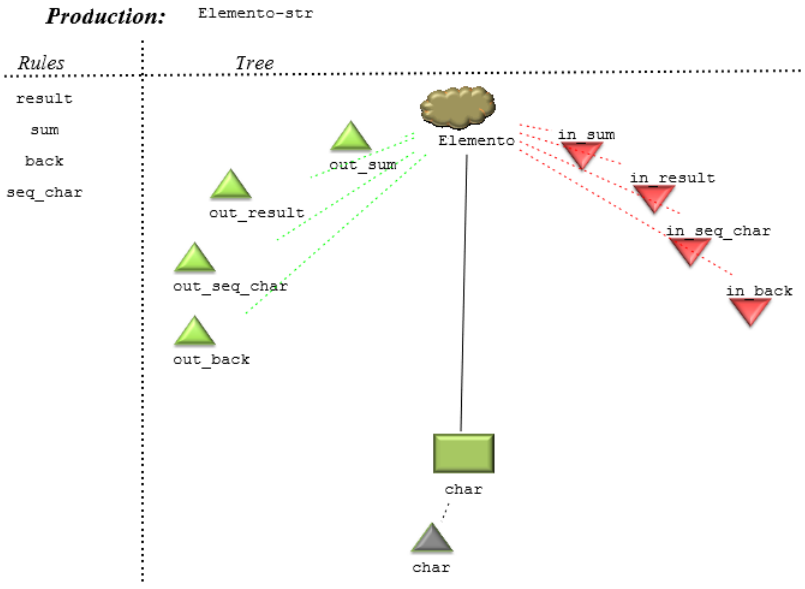

Figura 28: Produção P4

7.2 Regras

7.2.1 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.out_result`

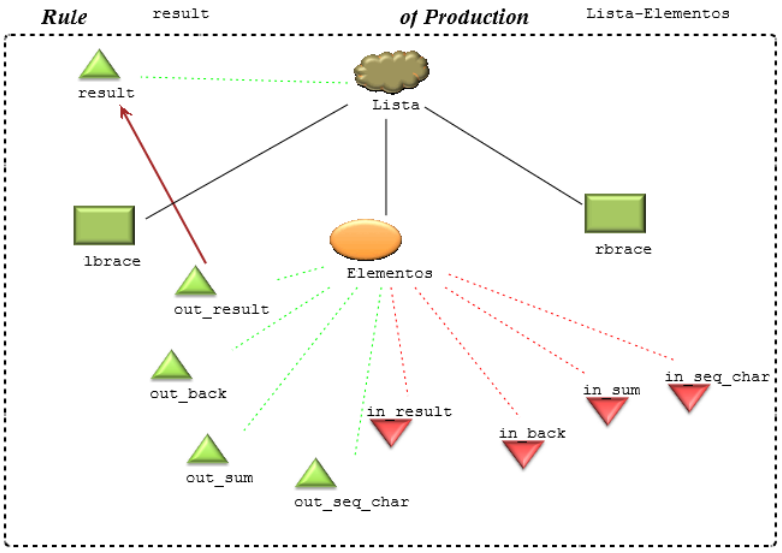

Figura 29: Regra para Lista.result

O que é feito nesta regra é inicializar as variáveis `in_sum`, `in_back`, `in_result` e `in_seq_char`.

```
Elementos.in_result = new ArrayList<Integer>();
Elementos.in_sum = 0
Elementos.in_back = -1
Elementos.in_seq_char = 0;
```

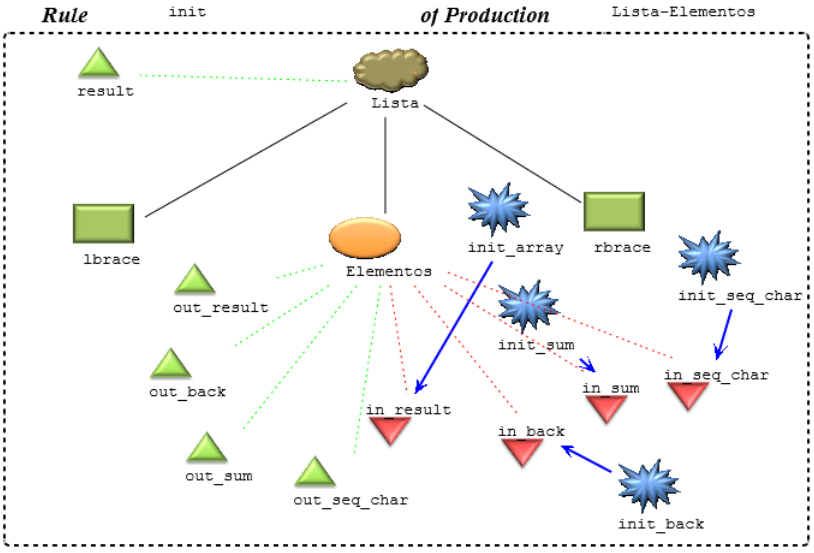

Figura 30: Regra para inicializar variáveis

7.2.2 Elementos -> Elemento

Aqui estão as regras:

```
Elemento.in_result = Elementos.in_result
Elemento.in_sum = Elementos.in_sum
```

```
Elemento.in_back = Elementos.in_back
Elemento.in_seq_char = Elementos.in_seq_char
Elementos.out_result = Elemento.out_result
Elementos.out_sum = Elemento.out_sum
Elementos.out_back = Elemento.out_back
Elementos.out_seq_char = Elemento.out_seq_char
```

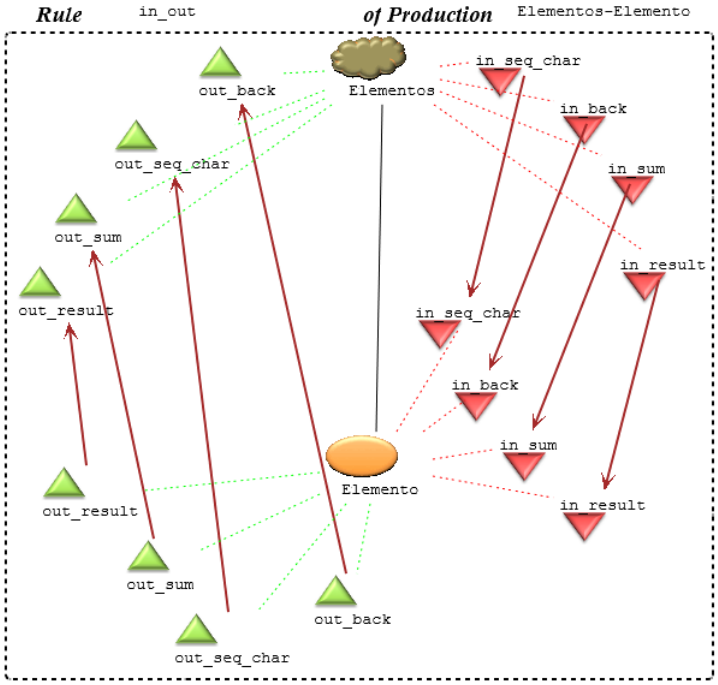

Figura 31: Regras in out

7.2.3 Elementos -> Elemento ',' Elementos

Regras:

```
Elementos0.out_sum = Elemento.out_sum
Elementos0.out_back = Elemento.out_back
Elementos0.out_result = Elemento.out_result
Elementos0.out_seq_char = Elemento.out_seq_char
Elemento.in_sum = Elementos1.out_sum
Elemento.in_back = Elementos1.out_back
Elemento.in_result = Elementos1.out_result
Elemento.in_seq_char = Elementos1.out_seq_char
Elementos1.in_sum = Elementos0.in_sum
Elementos1.in_back = Elementos0.in_back
Elementos1.in_result = Elementos0.in_result
Elementos1.in_seq_char = Elementos0.in_seq_char
```

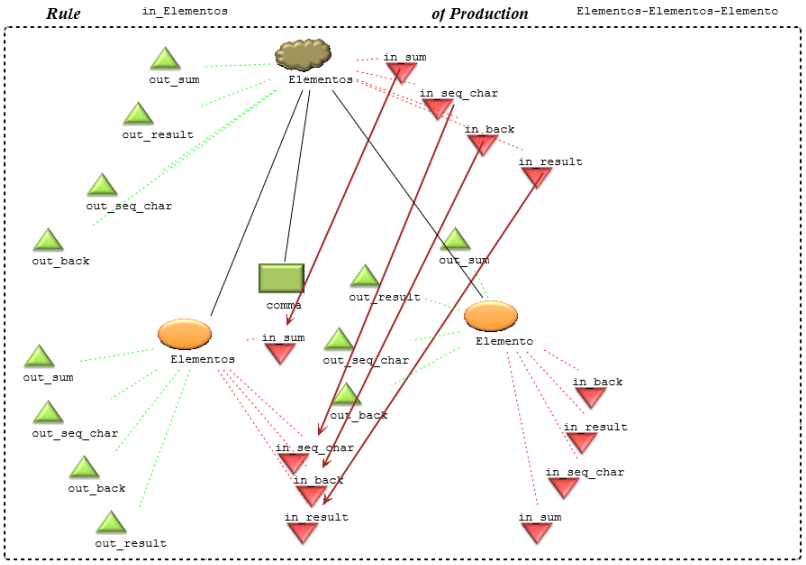

Figura 32: Regras in Elementos

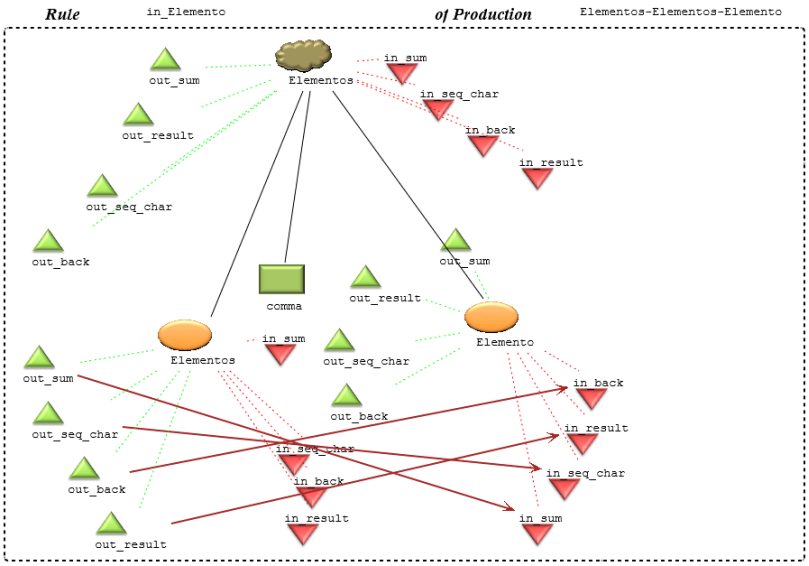

Figura 33: Regras in Elemento

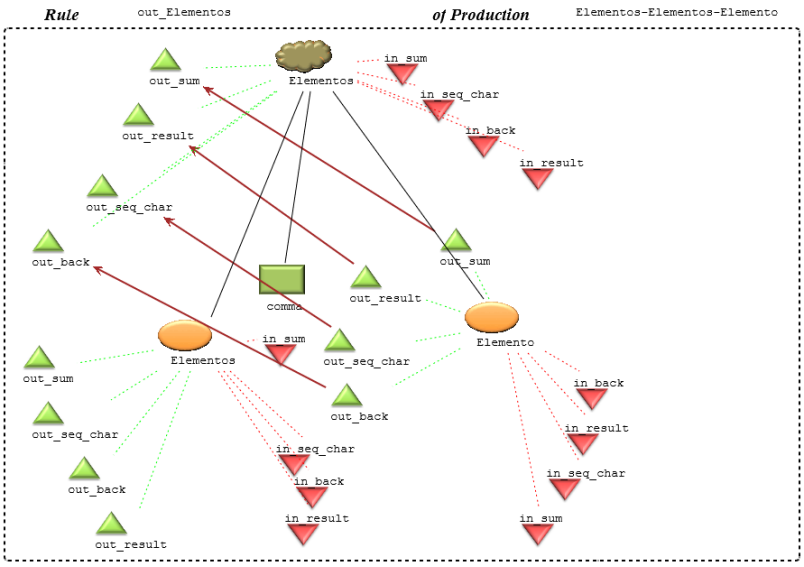

Figura 34: Regras out Elementos

7.2.4 Elemento -> int

```
Elemento.out_result = Elemento.in_result
Elemento.out_sum = function refresh_sum
Elemento.out_back = 0
Elemento.out_seq_char = Elemento.in_seq_char

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

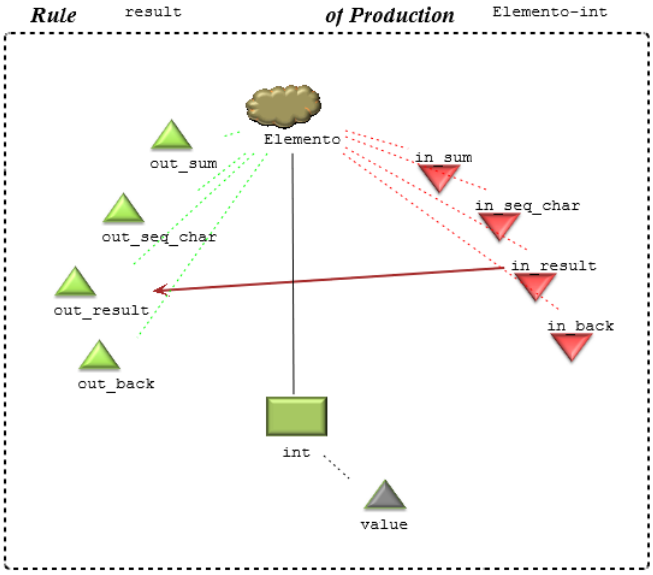

Figura 35: Regras result

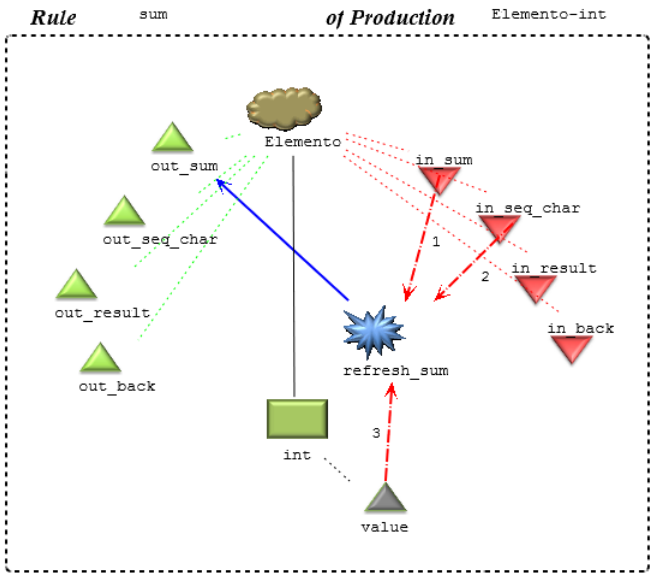

Figura 36: Regras sum

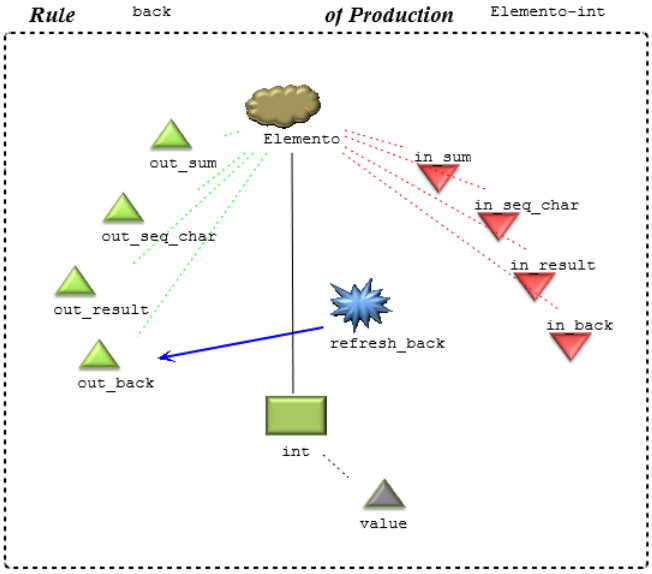

Figura 37: Regras back

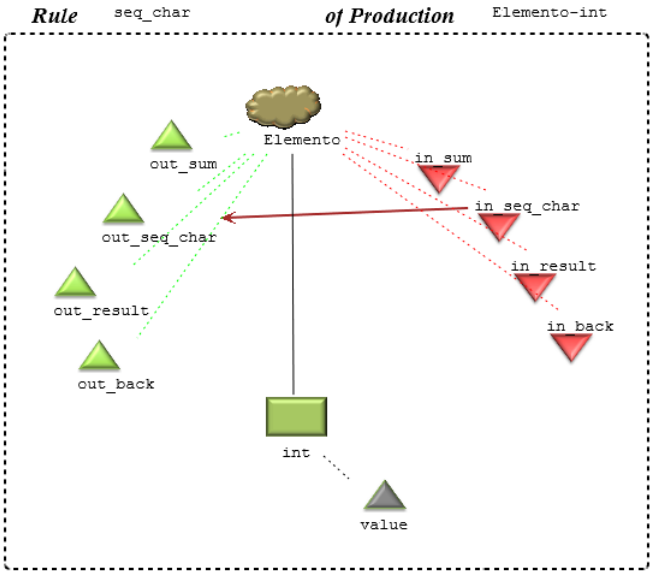

Figura 38: Regras seq char

7.2.5 Elemento -> str

```
Elemento.out_result = function refresh_result
Elemento.out_sum = 0
Elemento.out_back = 1
Elemento.out_seq_char = function refresh_seq_char

$1 = Elemento.in_result,
$2 = Elemento.in_sum,
$3 = Elemento.in_back,
ArrayList<Integer> refresh_result($1, $2, $3){
        if($3==0 && $2 > 0) return $1.add($2); else return $2;
}

$1 = Elemento.in_seq_char
$2 = Elemento.in_back
int refresh_seq_char($1,$2){
        if($2==1) return $1+1; else return 1;
}
```

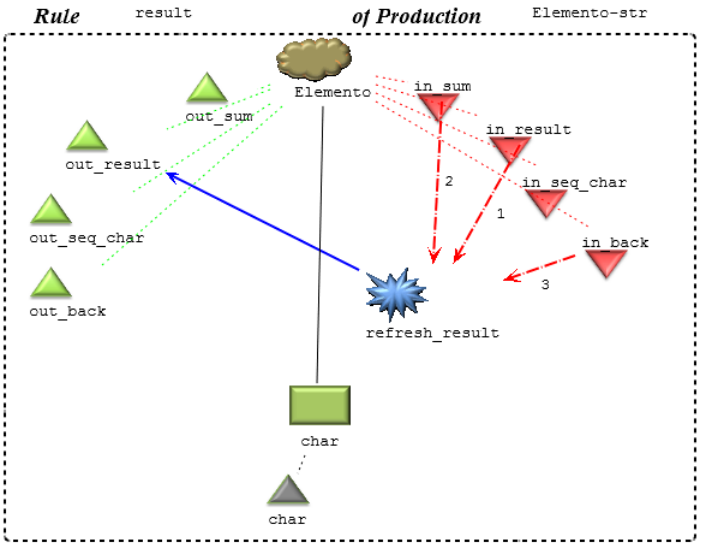

Figura 39: Regra result

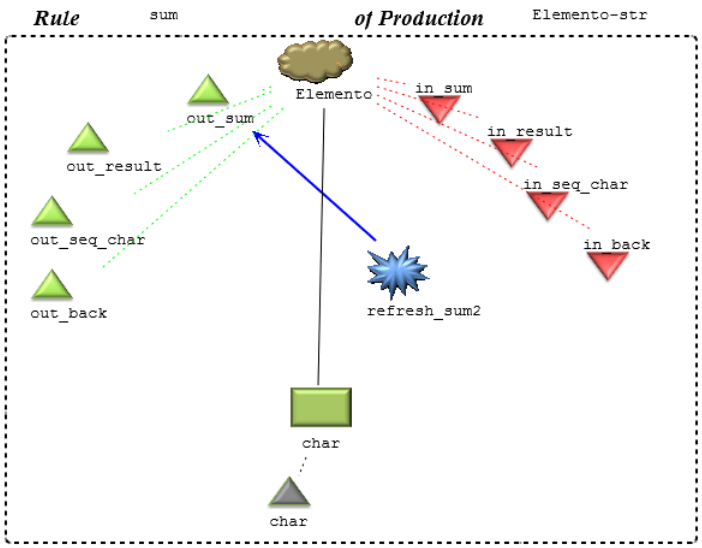

Figura 40: Regra sum

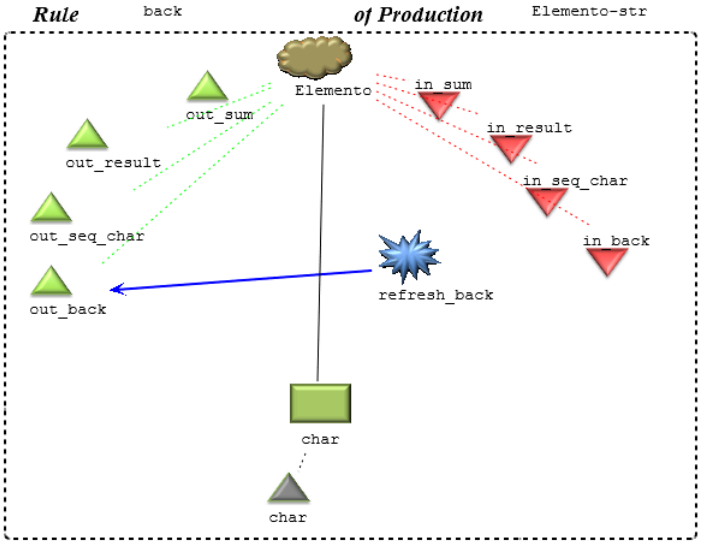

Figura 41: Regra back

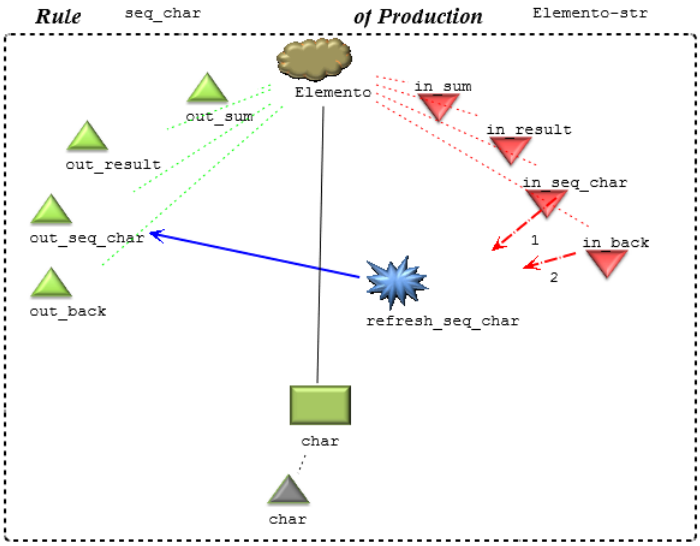

Figura 42: Regra seq char

8 Ex2 - Resultado Gerado pelo VisualLisa

Nesta secção está o que foi gerado a partir do VisualLisa.

8.1 BNF Grammer

```
Lista ->  lbrace Elementos rbrace
Elementos ->  Elemento
Elementos ->  Elementos comma Elemento
Elemento ->  int
Elemento ->  char
```

8.2 Linguagem LISA

```
language sum_list {

   lexicon{
      Lbrace    [
      Rbrace    ]
      Comma     ,
      Int     [0-9]
      Char     [a-zA-Z]
      ignore     [\0x09\0x0A\0x0D\ ]+
   }

   attributes
      java.util.ArrayList  LISTA.result;
      int  ELEMENTOS.out_back;
      int  ELEMENTOS.out_seq_char;
      int  ELEMENTOS.out_sum;
      java.util.ArrayList  ELEMENTOS.out_result;
      int  ELEMENTOS.in_seq_char;
      java.util.ArrayList  ELEMENTOS.in_result;
      int  ELEMENTOS.in_sum;
      int  ELEMENTOS.in_back;
      int  ELEMENTO.out_back;
      int  ELEMENTO.out_seq_char;
      java.util.ArrayList  ELEMENTO.out_result;
      int  ELEMENTO.out_sum;
      int  ELEMENTO.in_seq_char;
      int  ELEMENTO.in_back;
      java.util.ArrayList  ELEMENTO.in_result;
      int  ELEMENTO.in_sum;



   rule Lista_Elementos {
      LISTA ::=  #Lbrace ELEMENTOS #Rbrace compute {
         LISTA.result = ELEMENTOS.out_result;
         ELEMENTOS.in_sum = 0;
         ELEMENTOS.in_result = new ArrayList<Integer>();
         ELEMENTOS.in_back = -1;
```

```
      ELEMENTOS.in_seq_char = 0;
   };
}

rule Elementos_Elemento {
   ELEMENTOS ::=  ELEMENTO compute {
      ELEMENTO.in_seq_char = ELEMENTOS.in_seq_char;
      ELEMENTO.in_sum = ELEMENTOS.in_sum;
      ELEMENTO.in_back = ELEMENTOS.in_back;
      ELEMENTO.in_result = ELEMENTOS.in_result;
      ELEMENTOS.out_result = ELEMENTO.out_result;
      ELEMENTOS.out_sum = ELEMENTO.out_sum;
      ELEMENTOS.out_seq_char = ELEMENTO.out_seq_char;
      ELEMENTOS.out_back = ELEMENTO.out_back;
   };
}

rule Elementos_Elementos_Elemento {
   ELEMENTOS ::=  ELEMENTOS #Comma ELEMENTO compute {
      ELEMENTOS[1].in_sum = ELEMENTOS.in_sum;
      ELEMENTOS[1].in_seq_char = ELEMENTOS.in_seq_char;
      ELEMENTOS[1].in_back = ELEMENTOS.in_back;
      ELEMENTOS[1].in_result = ELEMENTOS.in_result;
      ELEMENTO.in_back = ELEMENTOS[1].out_back;
      ELEMENTO.in_result = ELEMENTOS[1].out_result;
      ELEMENTO.in_seq_char = ELEMENTOS[1].out_seq_char;
      ELEMENTO.in_sum = ELEMENTOS[1].out_sum;
      ELEMENTOS.out_seq_char = ELEMENTO.out_seq_char;
      ELEMENTOS.out_sum = ELEMENTO.out_sum;
      ELEMENTOS.out_back = ELEMENTO.out_back;
      ELEMENTOS.out_result = ELEMENTO.out_result;
   };
}

rule Elemento_int {
   ELEMENTO ::=  #Int compute {
      ELEMENTO.out_result = ELEMENTO.in_result;
      ELEMENTO.out_sum = if(ELEMENTO.in_seq_char>=3)
                           return ELEMENTO.in_sum+#Int.value;
                           else return ELEMENTO.in_sum;
      ELEMENTO.out_back = 0;
      ELEMENTO.out_seq_char = ELEMENTO.in_seq_char;
   };
}

rule Elemento_str {
   ELEMENTO ::=  #Char compute {
      ELEMENTO.out_result = if(ELEMENTO.in_back==0 && ELEMENTO.in_sum > 0)
                           return ELEMENTO.in_result.add(ELEMENTO.in_sum);
                           else return ELEMENTO.in_sum;
      ELEMENTO.out_sum = 0;
      ELEMENTO.out_back = 1;
      ELEMENTO.out_seq_char = if(ELEMENTO.in_back==1)
                                 return ELEMENTO.in_seq_char+1;
                                 else return 1;
   };
```

```
   }


   method user_Definitions {

   }
}
```

8.3 XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
   <attributeGrammar name="sum_list">
      <symbols>
         <terminals>
            <terminal id="lbrace">[</terminal>
            <terminal id="rbrace">]</terminal>
            <terminal id="comma">,</terminal>
            <terminal id="int">[0-9]</terminal>
            <terminal id="char">[a-zA-Z]</terminal>
            <terminal id="ignore">[\0x09\0x0A\0x0D\ ]+</terminal>
         </terminals>
         <nonterminals>
            <nonterminal id="Lista" />
            <nonterminal id="Elementos" />
            <nonterminal id="Elemento" />
         </nonterminals>
         <start nt="Lista" />
      </symbols>
      <attributesDecl>
         <declaration>
            <attribute id="Lista.result" type="java.util.ArrayList" class="SyntAttribute" />
         </declaration>
         <declaration>
            <attribute id="Elementos.out_back" type="int" class="SyntAttribute" />
            <attribute id="Elementos.out_seq_char" type="int" class="SyntAttribute" />
            <attribute id="Elementos.out_sum" type="int" class="SyntAttribute" />
            <attribute id="Elementos.out_result" type="java.util.ArrayList" class="SyntAttribute" />
            <attribute id="Elementos.in_seq_char" type="int" class="InhAttribute" />
            <attribute id="Elementos.in_result" type="java.util.ArrayList" class="InhAttribute" />
            <attribute id="Elementos.in_sum" type="int" class="InhAttribute" />
            <attribute id="Elementos.in_back" type="int" class="InhAttribute" />
         </declaration>
         <declaration>
            <attribute id="Elemento.out_back" type="int" class="SyntAttribute" />
            <attribute id="Elemento.out_seq_char" type="int" class="SyntAttribute" />
            <attribute id="Elemento.out_result" type="java.util.ArrayList" class="SyntAttribute" />
            <attribute id="Elemento.out_sum" type="int" class="SyntAttribute" />
            <attribute id="Elemento.in_seq_char" type="int" class="InhAttribute" />
            <attribute id="Elemento.in_back" type="int" class="InhAttribute" />
            <attribute id="Elemento.in_result" type="java.util.ArrayList" class="InhAttribute" />
            <attribute id="Elemento.in_sum" type="int" class="InhAttribute" />
         </declaration>
```

```
   <declaration>
      <attribute id="int.value" type="int" class="IntrinsicValueAttribute" />
   </declaration>
   <declaration>
      <attribute id="char.char" type="char" class="IntrinsicValueAttribute" />
   </declaration>
</attributesDecl>
<semanticProds>
   <semanticProd name="Lista-Elementos">
      <lhs nt="Lista" />
      <rhs>
         <element symbol="lbrace" />
         <element symbol="Elementos" />
         <element symbol="rbrace" />
      </rhs>
      <computations>
         <computation name="result">
            <assignedAttribute att="Lista.result" position="0" />
            <operation returnType="java.util.ArrayList">
               <argument att="Elementos.out_result" position="2" />
               <modus> $1 </modus>
            </operation>
         </computation>
         <computation name="init">
            <assignedAttribute att="Elementos.in_sum" position="2" />
            <operation returnType="int">

               <modus> 0 </modus>
            </operation>
         </computation>
         <computation name="init">
            <assignedAttribute att="Elementos.in_result" position="2" />
            <operation returnType="java.util.ArrayList">

               <modus> new ArrayList<Integer>() </modus>
            </operation>
         </computation>
         <computation name="init">
            <assignedAttribute att="Elementos.in_back" position="2" />
            <operation returnType="int">

               <modus> -1 </modus>
            </operation>
         </computation>
         <computation name="init">
            <assignedAttribute att="Elementos.in_seq_char" position="2" />
            <operation returnType="int">

               <modus> 0 </modus>
            </operation>
         </computation>
      </computations>
   </semanticProd>
   <semanticProd name="Elementos-Elemento">
      <lhs nt="Elementos" />
      <rhs>
```

```
   <element symbol="Elemento" />
</rhs>
<computations>
   <computation name="in_out">
      <assignedAttribute att="Elemento.in_seq_char" position="1" />
      <operation returnType="int">
         <argument att="Elementos.in_seq_char" position="0" />
         <modus> $1 </modus>
      </operation>
   </computation>
   <computation name="in_out">
      <assignedAttribute att="Elemento.in_sum" position="1" />
      <operation returnType="int">
         <argument att="Elementos.in_sum" position="0" />
         <modus> $1 </modus>
      </operation>
   </computation>
   <computation name="in_out">
      <assignedAttribute att="Elemento.in_back" position="1" />
      <operation returnType="int">
         <argument att="Elementos.in_back" position="0" />
         <modus> $1 </modus>
      </operation>
   </computation>
   <computation name="in_out">
      <assignedAttribute att="Elemento.in_result" position="1" />
      <operation returnType="java.util.ArrayList">
         <argument att="Elementos.in_result" position="0" />
         <modus> $1 </modus>
      </operation>
   </computation>
   <computation name="in_out">
      <assignedAttribute att="Elementos.out_result" position="0" />
      <operation returnType="java.util.ArrayList">
         <argument att="Elemento.out_result" position="1" />
         <modus> $1 </modus>
      </operation>
   </computation>
   <computation name="in_out">
      <assignedAttribute att="Elementos.out_sum" position="0" />
      <operation returnType="int">
         <argument att="Elemento.out_sum" position="1" />
         <modus> $1 </modus>
      </operation>
   </computation>
   <computation name="in_out">
      <assignedAttribute att="Elementos.out_seq_char" position="0" />
      <operation returnType="int">
         <argument att="Elemento.out_seq_char" position="1" />
         <modus> $1 </modus>
      </operation>
   </computation>
   <computation name="in_out">
      <assignedAttribute att="Elementos.out_back" position="0" />
      <operation returnType="int">
         <argument att="Elemento.out_back" position="1" />
```

```
            <modus> $1 </modus>
         </operation>
      </computation>
   </computations>
</semanticProd>
<semanticProd name="Elementos-Elementos-Elemento">
   <lhs nt="Elementos" />
   <rhs>
      <element symbol="Elementos" />
      <element symbol="comma" />
      <element symbol="Elemento" />
   </rhs>
   <computations>
      <computation name="in_Elementos">
         <assignedAttribute att="Elementos.in_sum" position="1" />
         <operation returnType="int">
            <argument att="Elementos.in_sum" position="0" />
            <modus> $1 </modus>
         </operation>
      </computation>
      <computation name="in_Elementos">
         <assignedAttribute att="Elementos.in_seq_char" position="1" />
         <operation returnType="int">
            <argument att="Elementos.in_seq_char" position="0" />
            <modus> $1 </modus>
         </operation>
      </computation>
      <computation name="in_Elementos">
         <assignedAttribute att="Elementos.in_back" position="1" />
         <operation returnType="int">
            <argument att="Elementos.in_back" position="0" />
            <modus> $1 </modus>
         </operation>
      </computation>
      <computation name="in_Elementos">
         <assignedAttribute att="Elementos.in_result" position="1" />
         <operation returnType="java.util.ArrayList">
            <argument att="Elementos.in_result" position="0" />
            <modus> $1 </modus>
         </operation>
      </computation>
      <computation name="in_Elemento">
         <assignedAttribute att="Elemento.in_back" position="3" />
         <operation returnType="int">
            <argument att="Elementos.out_back" position="1" />
            <modus> $1 </modus>
         </operation>
      </computation>
      <computation name="in_Elemento">
         <assignedAttribute att="Elemento.in_result" position="3" />
         <operation returnType="java.util.ArrayList">
            <argument att="Elementos.out_result" position="1" />
            <modus> $1 </modus>
         </operation>
      </computation>
      <computation name="in_Elemento">
```

```
            <assignedAttribute att="Elemento.in_seq_char" position="3" />
            <operation returnType="int">
               <argument att="Elementos.out_seq_char" position="1" />
               <modus> $1 </modus>
            </operation>
         </computation>
         <computation name="in_Elemento">
            <assignedAttribute att="Elemento.in_sum" position="3" />
            <operation returnType="int">
               <argument att="Elementos.out_sum" position="1" />
               <modus> $1 </modus>
            </operation>
         </computation>
         <computation name="out_Elementos">
            <assignedAttribute att="Elementos.out_seq_char" position="0" />
            <operation returnType="int">
               <argument att="Elemento.out_seq_char" position="3" />
               <modus> $1 </modus>
            </operation>
         </computation>
         <computation name="out_Elementos">
            <assignedAttribute att="Elementos.out_sum" position="0" />
            <operation returnType="int">
               <argument att="Elemento.out_sum" position="3" />
               <modus> $1 </modus>
            </operation>
         </computation>
         <computation name="out_Elementos">
            <assignedAttribute att="Elementos.out_back" position="0" />
            <operation returnType="int">
               <argument att="Elemento.out_back" position="3" />
               <modus> $1 </modus>
            </operation>
         </computation>
         <computation name="out_Elementos">
            <assignedAttribute att="Elementos.out_result" position="0" />
            <operation returnType="java.util.ArrayList">
               <argument att="Elemento.out_result" position="3" />
               <modus> $1 </modus>
            </operation>
         </computation>
      </computations>
   </semanticProd>
   <semanticProd name="Elemento-int">
      <lhs nt="Elemento" />
      <rhs>
         <element symbol="int" />
      </rhs>
      <computations>
         <computation name="result">
            <assignedAttribute att="Elemento.out_result" position="0" />
            <operation returnType="java.util.ArrayList">
               <argument att="Elemento.in_result" position="0" />
               <modus> $1 </modus>
            </operation>
         </computation>
```

```
      <computation name="sum">
         <assignedAttribute att="Elemento.out_sum" position="0" />
         <operation returnType="int">
            <argument att="Elemento.in_sum" position="0" />
            <argument att="Elemento.in_seq_char" position="0" />
            <argument att="int.value" position="1" />
            <modus> if($2>=3) return $1+$3; else return $1; </modus>
         </operation>
      </computation>
      <computation name="back">
         <assignedAttribute att="Elemento.out_back" position="0" />
         <operation returnType="int">

            <modus> 0 </modus>
         </operation>
      </computation>
      <computation name="seq_char">
         <assignedAttribute att="Elemento.out_seq_char" position="0" />
         <operation returnType="int">
            <argument att="Elemento.in_seq_char" position="0" />
            <modus> $1 </modus>
         </operation>
      </computation>
   </computations>
</semanticProd>
<semanticProd name="Elemento-str">
   <lhs nt="Elemento" />
   <rhs>
      <element symbol="char" />
   </rhs>
   <computations>
      <computation name="result">
         <assignedAttribute att="Elemento.out_result" position="0" />
         <operation returnType="java.util.ArrayList">
            <argument att="Elemento.in_result" position="0" />
            <argument att="Elemento.in_sum" position="0" />
            <argument att="Elemento.in_back" position="0" />
            <modus> if($3==0 && $2 > 0) return $1.add($2); else return $2; </modus>
         </operation>
      </computation>
      <computation name="sum">
         <assignedAttribute att="Elemento.out_sum" position="0" />
         <operation returnType="int">

            <modus> 0 </modus>
         </operation>
      </computation>
      <computation name="back">
         <assignedAttribute att="Elemento.out_back" position="0" />
         <operation returnType="int">

            <modus> 1 </modus>
         </operation>
      </computation>
      <computation name="seq_char">
         <assignedAttribute att="Elemento.out_seq_char" position="0" />
```

```
            <operation returnType="int">
               <argument att="Elemento.in_seq_char" position="0" />
               <argument att="Elemento.in_back" position="0" />
               <modus> if($2==1) return $1+1; else return 1; </modus>
            </operation>
         </computation>
      </computations>
   </semanticProd>
</semanticProds>
<importations>

</importations>
<functions>

</functions>
</attributeGrammar>
```

9 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.

Apesar de serem dois exercícios para calcular um resultado de forma diferente, deu para perceber que o reaciocínio para resolver é idêntifo com ambos os casos.

Serviu de consolidação da matéria dada até agora no módulo de Engenharia de Linguagens, tendo em conta que conseguimos resolver os exercícios com sucesso.