Simulator Language -> Gramática BNF

```
<program> ::= <program-section> <constant-section>
      <parameter-section> <variable-section>  <signature-section>
      <process-section> <topology-section>
      <semantic-section> <invariant-section>
      <closure-section> <leadsto-section>
<program-section> ::= program <program-name>
<program-name> :: <identifier-name>
<constant-section> ::= <empty>
      | const <constant-list>
<constant-list> ::= <constant>;
      | <constant-list> <constant>;
<constant> ::= <const-name> = <integer>
      | <const-name> = <boolean-value>
<constant-name> ::= <identifier-name>
<parameter-section> ::= <empty>
      | parameter <parameter-list>

<parameter-list> ::= <parameter>;
      | <parameter>; <parameter-list>
<parameter> ::= <parameter-name> : <interval>
<interval> ::= <lower>..<upper>
<lower> ::= <integer>
<upper> ::= <integer>
<variable-section> ::= var <variable-list>
<variable-list> ::= <variable-declaration> ;
      | <variable-list> <variable-declaration>;
<variable-declaration> ::= <variable-name><facet> : <type>
             | <variable-name><facet> : <type> <variable-init>
<variable_init> ::= [ <init_expression_list> ]
<init_expression_list> ::= <init_expression_list> , <init_expression>
             | <init_expression>
```

```
<init_expression> ::= <expression>

             | [ <init_expression_list> ]

<type> ::= <basic>

      | <constructor> of <basic>

<basic> ::= integer

     | boolean

       | real

       | string

       | record

<constructor> ::= set

<facet> ::= <empty>

      | .<parameter-name>

      | .<interval>

      | <facet>.<parameter-name>

      | <facet>.<interval>

<signature-section> ::= signature <signature-list>

<signature-list> ::= <signature-declaration> ;

      | <signature-list> <signature-declaration> ;

<signature-declaration> ::= <signature-name> : ( <typelist> )

<typelist> ::= <type>

      | <type> <typelist> ;

<process-section> ::= process <identifire-name> : <interval> actions <action-
list>

<action-list> ::= <action>

      | <action-list> [] <action>

<action> ::= <time-interval> <full-guards> -> <statements>

<time-interval> ::= <empty>

| [ <lower> , <upper> ]

| ( <lower> , <upper> )

| ( <lower> , <upper> ]

| [ <lower> , <upper> )

<full-guards> ::= <guards>

| <receive-guards>
| <guards> and <receive-guards>

<guards> ::= <guard>

      | <guards> and <guards>

      | <guards> or <guards>

      | <guards> implies <guards>

      | <guards> follows <guards>

      | <guards> equivalent <guards>

      | <guards> differs <guards>

<guard> ::= <expression>

      | <expression> <rel-op>

      <expression>

<rel-op> ::=  ==

      | <

      | !=

      | >=

      | >

      | <=

<receive-guards> ::= <receive-guard>

      | <receive-guard> and <receive-guards>

<receive-guard> ::= ( ? : <expression> : <target-list> )

<target-list> ::= <tag> , <target-list-tail>

<target-list-tail> ::=

      | <variable> , <target-list-tail>

<tag> ::= <expression>

<expression> ::= <unary-op> <term>

      | <unary-sign> <term> <binary-op-low> <term>


<term> ::= <factor>

      | <factor> <binary-op-high> <factor>

<factor> ::= <constant-name>

      | <parameter-name>

      | <variable>
      | ( <guards> )
      | not <factor>
      | constant
       | | <expression> |
       | <built-in-function> ( <expression-list> )
<unary-op> ::= <empty> | + | -
<binary-op-low> ::= +
      | -
       | union
       | intersection
       | in
<binary-op-high> ::= *
      | /
       | mod
<built-in-function> ::= max
      | min
       | rnd
       | ifsig
       | pack
<expression-list> ::= <expression>
      | <expression-list> , <expression>
<variable> ::= <variable-name>
      | <variable-name><facet-expression>
<facet-expression> ::= <empty>
      | .<factor>
      | <facet-expression>.<factor>
<variable-list> ::= <variable>
      | <variable-list> , <variable>
<statements> ::= <statement-list>
<statement-list> ::= <statement>
      | <statement-list>; <statement>
<statement> ::= <simple-statement>
```

```
      | <parallel-statement>

<simple-statement> ::= <assignment-statement>

      | <if-statement>

       | <print-statement>

       | <send-statement>

       | <unpack-statement>

      | <do-statement>

<assignment-statement> ::= <lhs> := <rhs>

<lhs> ::= <variable>

      | <lhs>,<variable>

<rhs> ::= <guard>

      | <rhs>,<guard>

<print-statement> ::= print ( <expressionlist> )

<send-statement> ::= (> <expression> : <expressionlist> )

<unpack-statement> ::= unpack ( <expression> : <signature-name> > <variable-list>
)

<if-statement> ::= if <guards> -> <statement-list> fi

<do-statement> ::= do <guards> -> <statement-list> od

<parallel-statement> ::= ( parallel <parameter-name> :

          <assign-or-if-statement> )

<empty> ::=   nil

<boolean-value> ::= true | false

<identifier-name> ::= <letter>

      | <identifier-name><letter>

      | <identifier-name><digit>


<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<letter> ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|

      A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

<integer> ::= <digit> | <integer><digit>

<topology-section> ::= topology <topology-definition>

<topology-definition> ::= <integer>

                | <integer> <topology-description>
```

```
<topology-description> ::= complete | linear | ring | binarytree

<semantic-section> ::= semantics <semantics>

<semantics> ::= synchronous

      | max-parallel

      | min-parallel

      | interleaving

<invariant-section>  ::= invariant <expression-list>

      | <empty>

<closed-section>  ::= closed <expression-list>

      | <empty>

<leadsto-section>  ::= leadsto <leadsto-expression-list>

      | <empty>

<expression-list>  ::= <expression>

      | <expression-list>, <expression

<leadsto-expression-list> ::= <expression> |-> <expression>

      | <leadsto-expression-list> <expression> |-> <expression>

<fault-section> ::= faults <action-list>
```