Cosmos 1.0 Input File Grammar

Hilal Djafri

1 Syntax

Generic Syntax Let us start by the grammar of some common symbols before giving the grammar of each input file.

A natural number <Integer>, a real number <Real> or string type <Str> are defined like this:

⟨Integer⟩ ::= [0-9]+| [0-9]*
⟨Real⟩ ::= ([0-9]+| [0-9]*.[0-9]+)([eE][-+]?[0-9]+)?
⟨Str⟩ ::= [a-zA-Z][a-zA-Z_0-9]*

All the symbols finishing by "**Tag**" refer to a tag of an object. These symbols are string type :

- <IConstTag>: A tag of a natural number constant.
- <RConstTag>: A tag of a real constant.
- <PTag>: A tag of a Petri net place.
- <TTag>: A tag of a Petri net transition.
- <LTag>: A tag of an automaton location.
- <VTag>: A tag of an automaton variable.

It is useful to define some integer constants <IConstant> or real constants <RConstant> which can be used by other definitions:

⟨IConstant⟩ ::= "const" "int" ⟨IConstTag⟩ "=" ⟨Integer⟩ ";"
⟨RConstant⟩ ::= "const" "double" ⟨RConstTag⟩ "=" ⟨Real⟩ ";"

Some numerical attributes (marking values, transitions parameters, arcs multiplicity, variables rate, etc.) may be introduced as a function of numerical values (real and/or integer), constants and/or Petri net places. Let us give the grammar of such functions:

The first kind of these function is <RFormula> for real formula. It intervenes numerical values (integer or real) and constants (integer or/and real).

⟨RFormula⟩ ::= ⟨Real⟩ | ⟨RConstTag⟩ | ⟨IFormula⟩ | ⟨RFormula⟩ ⟨ArOP⟩ ⟨RFormula⟩ | "(" ⟨RFormula⟩ ")"

The second kind is <IFormula> for integer formula. It intervenes numerical values (integer or real) and constants (integer or/and real) but its value should be always a natural number.

⟨IFormula⟩ ::= ⟨Integer⟩ | ⟨IConstTag⟩ | ⟨IFormula⟩ ⟨ArOpRes⟩ ⟨IFormula⟩ | "(" ⟨IFormula⟩ ")" | "floor" "(" ⟨RFormula⟩ ")"

The third kind of functions is <MRFormula> for marking real formula. It intervenes numerical values (integer or real), constants (integer or/and real) and Petri places.

⟨MRFormula⟩ ::= ⟨PTag⟩ | ⟨RFormula⟩ | ⟨MRFormula⟩ ⟨ArOp⟩ ⟨MRFormula⟩ | "(" ⟨MRFormula⟩ ")"

The last type of functions is <MIFormula> for marking integer formula. It intervenes numerical values (integer or real), constants (integer or/and real) and Petri places but its value should be always a natural number.

⟨MIFormula⟩ ::= ⟨PTag⟩ | ⟨IFormula⟩ | ⟨MIFormula⟩ ⟨ArOpRes⟩ ⟨MIFormula⟩ | "(" ⟨MIFormula⟩ ")" | "floor" "(" ⟨RFormula⟩ ")"

These, functions are defined with this set of arithmetic operators <ArOp> or a its restricted set <ArOpRes> :

⟨ArOp⟩ ::= "+" | "-" | "*" | "/" | "^"
⟨ArOpRes⟩ ::= "+" | "-" | "*" | "^"

GSPN Syntax The definition of the Petri net consists of:

⟨GSPN⟩ ::= {⟨IConstant⟩ } {⟨RConstant⟩ } ⟨NT⟩ ⟨NP⟩ ⟨PList⟩ ⟨TList⟩ ⟨InitMarking⟩ ⟨TransitionsDef⟩ [⟨InArcs⟩] [⟨OutArcs⟩] [⟨InhibArcs⟩]

In the first part, some integer and/or real constants can be declared. The size of the Petri net (number of transitions and places) must be declared.

⟨IConstant⟩ ::= "const" "int" ⟨IConstTag⟩ "=" ⟨Integer⟩ ";"
⟨RConstant⟩ ::= "const" "double" ⟨RConstTag⟩ "=" ⟨Real⟩ ";"
⟨NT⟩ ::= "NbTransitions" "=" ⟨Integer⟩ ";" | "NbTransitions" "=" ⟨IFormula⟩ ";"
⟨NP⟩ ::= "NbPlaces" "=" ⟨Integer⟩ ";" | "NbPlaces" "=" ⟨IFormula⟩ ";"

Then, the set of transitions and places must defined:

⟨PList⟩ ::= "PlacesList" "=" "{" ⟨PTags⟩ "}" ";"
⟨PTags⟩ ::= ⟨PTag⟩ | ⟨PTags⟩ "," ⟨PTag⟩
⟨TList⟩ ::= "TransitionsList" "=" "{" ⟨TTags⟩ "}" ";"
⟨TTags⟩ ::= ⟨TTag⟩ | ⟨TTags⟩ "," ⟨TTag⟩

After that, the initial marking is given. By default all the places contain zero token.

⟨InitMarking⟩ ::= "Marking" "=" "{" ⟨Inits⟩ "}"
⟨Inits⟩ ::= ⟨Init⟩ | ⟨Init⟩ "," ⟨Inits⟩
⟨Init⟩ ::= "(" ⟨PTag⟩ "," IFormula ")"

The next step consists of a complete description of the transitions. Note that transitions which exponentially distributed will be defined differently from those with other distributions.

⟨TransitionsDef⟩ ::= "Transitions" "=" "{" ⟨Transitions⟩ "}" ";"
⟨Transitions⟩ ::= ⟨Transition ⟩ | ⟨Transitions⟩ "," ⟨Transition⟩
⟨Transition⟩ ::= ⟨Exp⟩ | ⟨NonExp⟩

A transition with an exponential distribution can be marking dependent parameter. A priority and a weight will be given. A service policy will be chosen.

⟨Exp⟩ ::= "(" ⟨TTag⟩ "," "EXPONENTIAL" "(" ⟨MRFormula⟩ ")" "," ⟨Priority⟩ "," ⟨Weight⟩ "," ⟨Service⟩ "," ⟨Memory⟩ ")" | "(" ⟨TTag⟩ "," "EXPONENTIAL" "(" ⟨MRFormula⟩ ")" "," ⟨Priority⟩ "," ⟨Weight⟩ "," ⟨Service⟩ ")"
⟨Service⟩ ::= "SINGLE" | "INFINITE" | "MULTIPLE" "(" ⟨integer⟩ ")"
⟨Memory⟩ ::= "ENABLEDMEMORY" | "AGEMEMORY"

A transition with non exponential distribution can't be marking dependent parameters. A priority and weight will be given. There is no service policy to chose (the only possible is single service). Then A memory policy can be chosen, by default the policy is enabled memory:

⟨NonExp⟩ ::= "(" ⟨TTag⟩ "," ⟨Dist⟩ "," ⟨Priority⟩ "," ⟨Weight⟩ "," ⟨Memory⟩ ")" | "(" ⟨TTag⟩ "," ⟨Dist⟩ "," ⟨Priority⟩ "," ⟨Weight⟩ ")"
⟨Dist⟩ ::= "IMMEDIATE" | "DETERMINISTIC" "(" ⟨Real⟩ ")" | "UNIFORM" "(" ⟨Real⟩ "," ⟨Real⟩ ")" "ERLANG" "(" ⟨Integer⟩ "," ⟨Real⟩ ")" | "GAMMA" "(" ⟨Real⟩ "," ⟨Real⟩ ")" | "TRIANGLE" "(" ⟨Real⟩ "," ⟨Real⟩ "," ⟨Real⟩ ")" | "GEOMETRIC" "(" ⟨Real⟩ "," ⟨Real⟩ ")" | "LOGNORMAL" "(" ⟨Real⟩ "," ⟨Real⟩ ")"

The final part consists of introducing the different matrices of the net. Note that the arcs multiplicity can be marking dependent.

⟨In⟩ ::= "InArcs" "=" "{" ⟨InArcs⟩ "}" ";"
⟨InArcs⟩ ::= ⟨InArc⟩ | ⟨InArcs⟩ "," ⟨InArcs⟩
⟨InArc⟩ ::= "(" ⟨PTag⟩ "," ⟨TTag⟩ ")" | "(" ⟨PTag⟩ "," ⟨TTag⟩ "," ⟨MIFormula⟩ ")"
⟨Out⟩ ::= "OutArcs" "=" "{" ⟨OutArcs⟩ "}" ";"
⟨OutArcs⟩ ::= ⟨OutArc⟩ | ⟨OutArcs⟩ "," ⟨OutArcs⟩
⟨OutArc⟩ ::= "(" ⟨TTag⟩ "," ⟨PTag⟩ ")" | "(" ⟨TTag⟩ "," ⟨PTag⟩ "," ⟨MIFormula⟩ ")"
⟨Inhib⟩ ::= "InhibArcs" "=" "{" ⟨InhibArcs⟩ "}" ";"
⟨InhibArcs⟩ ::= ⟨InhibArc⟩ | ⟨InhibArcs⟩ "," ⟨InhibArcs⟩
⟨InhibArc⟩ ::= "(" ⟨PTag⟩ "," ⟨TTag⟩ ")" | "(" ⟨PTag⟩ "," ⟨TTag⟩ "," ⟨MIFormula⟩ ")"

HASL Syntax The definition of the HASL formula consists of:

⟨HASL⟩ ::= {⟨IConstant⟩ } {⟨RConstant⟩ } ⟨NL⟩ ⟨NV⟩ ⟨LList⟩ ⟨VList⟩ ⟨Expression⟩ ⟨InitLoc⟩ ⟨FinalLoc⟩ ⟨LocDef⟩ [⟨Edges⟩]

In the first part some constants can be declared. The number of locations and variables must be given.

⟨IConstant⟩ ::= "const" "int" ⟨IConstTag⟩ "=" ⟨Integer⟩ ";"
⟨RConstant⟩ ::= "const" "double" ⟨RConstTag⟩ "=" ⟨Real⟩ ";"
⟨NL⟩ ::= "NbLocations" "=" ⟨Integer⟩ ";" | "NbLocations" "=" ⟨IFormula⟩ ";"
⟨NV⟩ ::= "NbVariables" "=" ⟨Integer⟩ ";" | "NbVariables" "=" ⟨IFormula⟩ ";"

Then set of locations and variables will be declared:

⟨LList⟩ ::= "LocationsList" "=" "{" ⟨LTags⟩ "}" ";"
⟨LTags⟩ ::= ⟨LTag⟩ | ⟨LTags⟩ "," ⟨LTag⟩
⟨VList⟩ ::= "VariablesList" "=" "{" ⟨VTags⟩ "}" ";"
⟨VTags⟩ ::= ⟨VTag⟩ | ⟨VTags⟩ "," ⟨VTag⟩

Then the hasl expression will be introduced:

⟨ExpectExp⟩ ::= "AVG" "(" ⟨F⟩ ")" ";"
⟨F⟩ ::= ⟨H⟩ | ⟨F⟩ "/" ⟨RFormula⟩ | ⟨F⟩ "*" ⟨RFormula⟩ | ⟨F⟩ ⟨ArOp⟩ ⟨F⟩ | "min" "(" ⟨F⟩ "," ⟨F⟩ ")" | "max" "(" ⟨F⟩ "," ⟨F⟩ ")"
⟨H⟩ ::= "Last" "(" ⟨LX⟩ ")" | "Min" "(" ⟨LX⟩ ")" | "Max" "(" ⟨LX⟩ ")" | "Integral" "(" ⟨LX⟩ ")" | "Mean" "(" ⟨LX⟩ ")" | "Var" "(" ⟨LX⟩ ")"
⟨LX⟩ ::= ⟨term⟩ | ⟨term⟩ "+" ⟨term⟩ | ⟨term⟩ "-" ⟨term⟩
⟨term⟩ ::= ⟨VTag⟩ | ⟨Real⟩ "*" ⟨VTag⟩ | "(" ⟨MRFormula⟩ ")" "*" ⟨VTag⟩

The set of initial and final locations will be given:

⟨InitLoc⟩ ::= "InitialLocations" "=" "{" ⟨LTags⟩ "}" ";"
⟨FinalLoc⟩ ::= "FinalLocations" "=" "{" ⟨LTags⟩ "}" ";"

Then, the locations will be completely described. Each location is tagged with <LTag> and satisfies a property on the marking of the Petri net. At each location, the rates of the variables are given. By default rates are set to zero.

⟨LocDef⟩ ::= "Locations" "=" "{" ⟨Ldefs⟩ "}" ";"
⟨Ldefs⟩ ::= ⟨Ldef⟩ | ⟨Ldefs⟩ "," ⟨Ldef⟩
⟨Ldef⟩ ::= "(" ⟨LTag⟩ "," ⟨MLFormula⟩ "," "(" ⟨Vrates⟩ ")" ")"

⟨MLFormula⟩ ::= "TRUE" | ⟨MRFormula⟩ ⟨CompOp⟩ ⟨MRFormula⟩ | ⟨MLFormula⟩ ⟨LogOp⟩ ⟨MLFormula⟩ | "!" "(" ⟨MLFormula⟩ ")"
⟨CompOp⟩ ::= "=" | ">" | "<" | ">=" | "<="
⟨LogOp⟩ ::= "&" | "|"

⟨Vrates⟩ ::= ⟨Vrate⟩ | ⟨Vrates⟩ "," ⟨Vrate⟩
⟨Vrate⟩ ::= ⟨VTag⟩ ":" MRFormula

Finally, the edges will be defined. An edge relies a location source to a location target (<LTag> , <LTag>). Each edge is associated to a set of Petri net transitions <Actions>. If the edge is synchronized with all Petri transitions then <Actions> will take value "ALL". If the edge is not synchronized with the Petri net (i.e an autonomous edge) then <Actions> will take value

"#". Each edge is associated to a set of linear constraints on automaton variable <Constraints>. If the edge is not subject to any constraint then <Constraints> will take value "#". Each edge is also associated to a set of variable updates <Ups>. If no update is required then <Ups> will take value "#".

⟨Edges⟩ ::= "Edges" "=" "{" ⟨Edefs⟩ "}" ";"
⟨Edefs⟩ ::= ⟨Edef⟩ | ⟨Edefs⟩ "," ⟨Edef⟩
⟨Edef⟩ ::= "(" "(" ⟨LTag⟩ "," ⟨LTag⟩ ")" "," ⟨Actions⟩ "," ⟨Constraints⟩ "," ⟨Updates⟩ ")"
⟨Actions⟩ ::= "#" | "{" ⟨PTags⟩ "}" | "ALL" "
" "{" ⟨PTags⟩ "}"
⟨Constraints⟩ ::= "#" | Constraint | Constraint "&" Constraints
⟨Constraint⟩ ::= ⟨LX⟩ "=" ⟨MRFormula⟩ | ⟨LX⟩ ">=" ⟨MRFormula⟩ | ⟨LX⟩ "<=" ⟨MRFormula⟩
⟨Updates⟩ ::= "{" ⟨Ups⟩ "}" ";"
⟨Ups⟩ ::= ⟨Up⟩ "," ⟨Ups⟩
⟨Up⟩ ::= ⟨VTag⟩ "=" ⟨VMRFormula⟩ | ⟨VMRFormula⟩ ::= ⟨VTag⟩ | ⟨MRFormula⟩ | ⟨VMRFormula⟩ ⟨ArOp⟩ ⟨VMRFormula⟩