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:

\[
\begin{align*}
\langle \text{Integer} \rangle & ::= [0-9]+ | [0-9]^* \\
\langle \text{Real} \rangle & ::= ([0-9]+ | [0-9]^*[0-9]+)[(eE)[\pm][0-9]+] \\
\langle \text{Str} \rangle & ::= [a-zA-Z][a-zA-Z_0-9]^*
\end{align*}
\]

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

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

It is useful to define some integer constants \(<\text{IConstant}>\) or real constants \(<\text{RConstant}>\) which can be used by other definitions:

\[
\begin{align*}
\langle \text{IConstant} \rangle & ::= \text{"const" \"int\" } \langle \text{IConstTag} \rangle \text{\"=" } \langle \text{Integer} \rangle \text{\";} \\
\langle \text{RConstant} \rangle & ::= \text{"const" \"double\" } \langle \text{RConstTag} \rangle \text{\"=" } \langle \text{Real} \rangle \text{\";}
\end{align*}
\]

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 \(<\text{RFormula}>\) for real formula. It intervenes numerical values (integer or real) and constants (integer or and real):

\[
\begin{align*}
\langle \text{RFormula} \rangle ::= \langle \text{Real} \rangle | \langle \text{RConstTag} \rangle | \langle \text{IFormula} \rangle | \langle \text{RFormula} \rangle \langle \text{ArOP} \rangle \\
\langle \text{RFormula} \rangle | \langle \langle \text{RFormula} \rangle \text{\"\rangle\rangle} \\
\end{align*}
\]

The second kind is \(<\text{IFormula}>\) 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.
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.

\[
\langle \text{MRFormula} \rangle ::= \langle \text{PTag} \rangle | \langle \text{RFormula} \rangle | \langle \text{MRFormula} \rangle \langle \text{ArOp} \rangle \langle \text{MRFormula} \rangle | "(\langle \text{MRFormula} \rangle )"
\]

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.

\[
\langle \text{MIFormula} \rangle ::= \langle \text{PTag} \rangle | \langle \text{IFormula} \rangle | \langle \text{MIFormula} \rangle \langle \text{ArOpRes} \rangle \langle \text{MIFormula} \rangle | "(\langle \text{MIFormula} \rangle )" | \text{"floor" } "(\langle \text{RFormula} \rangle )"
\]

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

\[
\langle \text{ArOp} \rangle ::= + | - | * | / | ^
\]

\[
\langle \text{ArOpRes} \rangle ::= + | - | * | ^
\]

**GSPN Syntax** The definition of the Petri net consists of:

\[
\langle \text{GSPN} \rangle ::= \{\langle \text{IConstant} \rangle \} \{\langle \text{RConstant} \rangle \} \langle \text{NT} \rangle \langle \text{NP} \rangle \langle \text{PList} \rangle \langle \text{TList} \rangle \langle \text{InitMarking} \rangle \langle \text{TransitionsDef} \rangle \langle \text{InArcs} \rangle \langle \text{OutArcs} \rangle \langle \text{InhibArcs} \rangle
\]

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.

\[
\langle \text{IConstant} \rangle ::= \text{"const" } \text{"int" } \langle \text{IConstTag} \rangle \text{"=" } \langle \text{Integer} \rangle \text{;""
\]

\[
\langle \text{RConstant} \rangle ::= \text{"const" } \text{"double" } \langle \text{RConstTag} \rangle \text{"=" } \langle \text{Real} \rangle \text{;"
\]

\[
\langle \text{NT} \rangle ::= \text{"NbTransitions" } \text{"=" } \langle \text{Integer} \rangle \text{;" | "NbTransitions" } \text{"=" } \langle \text{IFormula} \rangle \text{;"
\]

\[
\langle \text{NP} \rangle ::= \text{"NbPlaces" } \text{"=" } \langle \text{Integer} \rangle \text{;" | "NbPlaces" } \text{"=" } \langle \text{IFormula} \rangle \text{;"
\]

Then, the set of transitions and places must defined:

\[
\langle \text{PList} \rangle ::= \text{"PlacesList" } \text{"=" } [\langle \text{PTags} \rangle ] \text{;"
\]

\[
\langle \text{PTags} \rangle ::= \langle \text{PTag} \rangle | \langle \text{PTags} \rangle \text{"," } \langle \text{PTag} \rangle
\]

\[
\langle \text{TList} \rangle ::= \text{"TransitionsList" } \text{"=" } [\langle \text{TTags} \rangle ] \text{;"
\]

\[
\langle \text{TTags} \rangle ::= \langle \text{TTag} \rangle | \langle \text{TTags} \rangle \text{"," } \langle \text{TTag} \rangle
\]

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

\[
\langle \text{InitMarking} \rangle ::= \text{"Marking" } \text{"=" } [\langle \text{Inits} \rangle ]\text{"
\]

\[
\langle \text{Inits} \rangle ::= \langle \text{Init} \rangle | \langle \text{Init} \rangle \text{"," } \langle \text{Inits} \rangle
\]

\[
\langle \text{Init} \rangle ::= \text{"(\langle \text{PTag} \rangle "," } \langle \text{IFormula} \rangle \text{")"
\]
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.

\[
\text{(TransitionsDef)} ::= \"\text{Transitions}\" \text{ }=\text{ }\{\text{ (Transitions) }\} \text{" }\}; \\
\text{(Transitions)} ::= \{\text{ (Transition) }\} | \{\text{ (Transitions) }\}; \text{ (Transition)} \\
\text{(Transition)} ::= \{\text{ (Exp) }\} | \{\text{ (NonExp) }\}
\]

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.

\[
\text{(Exp)} ::= \text{( TTag) } \text{" }\}; \text{ "EXPOENENTIAL" }\{\text{ (MRFormula)}\} \text{" }\}; \text{ (Priority) } \\
\}; \text{ (Weight) } \text{" }\}; \text{ "Service" } \text{" }\}; \text{ (Memory) } \} | \text{ ( TTag) } \text{" }\}; \text{ "EXPOENENTIAL" }\{\text{(MRFormula)}\} \text{" }\}; \text{ (Priority) }\text{" }\}; \text{ (Weight) } \text{" }\}; \text{ (Service) } \}
\]

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:

\[
\text{(NonExp)} ::= \text{( TTag) } \text{" }\}; \text{ "INFINITE" }\{\text{ (MRFormula)}\} \text{" }\}; \text{ (Priority) }\text{" }\}; \text{ (Weight) } \text{" }\}; \text{ (Memory) } \} | \text{ ( TTag) } \text{" }\}; \text{ "AGEMEMORY" }\{\text{(MRFormula)}\} \text{" }\}; \text{ (Priority) }\text{" }\}; \text{ (Weight) } \text{" }\}; \text{ (Service) } \}
\]

A transition with non exponential distribution can’t be marking dependent parameters. A priority and weight will be given. A service policy will be chosen.

\[
\text{(Dist)} ::= \text{"IMMEDIATE" }\{| \text{ "DETERMINISTIC" }\{\text{ (Real)}\} \} | \text{ "UNIFORM" }\{\text{(Real)\} | \text{ "GAMMA" }\{\text{ (Real)\} | \text{ "TRIANGLE" }\{\text{ (Real)\} | \text{ "GEOMETRIC" }\{\text{ (Real)\} | \text{ "LOGNORMAL" }\{\text{ (Real)\} | \text{ "GEOMETRIC" }\{\text{ (Real)\}
\]

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

\[
\text{(In)} ::= \text{"InArcs" }=\text{ }\{|\text{ (InArcs) }\} \} \}; \\
\text{(InArcs)} ::= \{\text{ (InArc) }\} | \{\text{ (InArcs) }\}; \text{" }\}; \{\text{ (InArcs) }\}
\]

\[
\text{(Out)} ::= \text{"OutArcs" }=\text{ }\{|\text{ (OutArcs) }\} \} \}; \\
\text{(OutArcs)} ::= \{\text{ (OutArc) }\} | \{\text{ (OutArcs) }\}; \text{" }\}; \{\text{ (OutArcs) }\}
\]

\[
\text{(Inhib)} ::= \text{"InhibArcs" }=\text{ }\{|\text{ (InhibArcs) }\} \} \}; \\
\text{(InhibArcs)} ::= \{\text{ (InhibArc) }\} | \{\text{ (InhibArcs) }\}; \text{" }\}; \{\text{ (InhibArcs) }\}
\]

\[
\text{HASL Syntax} \text{ The definition of the HASL formula consists of:} \\
\text{(HASL)} ::= \{\text{ (IConstant) }\} | \{\text{ (RConstant) }\} | \{\text{ (NL) }\} | \{\text{ (NV) }\} | \{\text{ (LList) }\} | \{\text{ (VList) }\} | \{\text{ (Expression) }\} | \{\text{ (InitLoc) }\} | \{\text{ (FinalLoc) }\} | \{\text{ (LocDef) }\} | \{\text{ (Edges) }\}
\]

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) \"\" (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) | (MRF) \"\" (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\" | (MRF) (CompOp) (MRF) | (MLFormula)
(LogOp) (MLFormula) | "\" (MLFormula) \"\"
(CompOp) ::= \"\" | "\" (MLFormula) \"\"
(LogOp) ::= \"\" &\" | "\" |

(Vrates) ::= (Vrate) | (Vrates) \"\" (Vrate)
(Vrate) ::= (VTag) \"\" MRF

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" ";" ";" ";" ";"
(Constraints) ::= "#" | Constraint | Constraint ",&" Constraints
(Constraint) ::= (LX) ";=" (MRFormula) | (LX) ";>=" (MRFormula) | (LX) ";<=" (MRFormula)
(Updates) ::= "{" (Ups) "}" ";"
(Ups) ::= (Up) "," (Ups)
(Up) ::= (VTag) ";=" (VMRFormula) | (VMRFormula) ::= (VTag) | (MRFormula) | (VMRFormula) \\langle V\text{ArOp} \rangle (VMRFormula)