DataModel - Constraints

S. Antonelli

17/08/2019

Contents

1 Constraints

1.1 Variable

1. `L'assegnamento iniziale di una variabile deve riferirsi ad una del modulo`
   $\forall\, m, v, e\ Module(m) \wedge initialAss(m, v, e) \wedge Expression(e) \wedge Variable(v) \Rightarrow modVar(m, v)$

2. `L'assegnamento iniziale di una variabile deve essere coerente con il tipo della variabile`
   $\forall\, m, v, e, t\ Module(m) \wedge initialAss(m, v, e) \wedge Expression(e) \wedge typeExpr(t, v) \wedge Type(t) \wedge Variable(v) \Rightarrow typeExpr(t, e)$

1.2 Expression

1. `La condizione di un'espressione condizionale deve essere booleana`
   $\forall\, e, c, th\ Expression(e) \wedge ConditionalExpression(e) \wedge Expression(val) \wedge cond(c, e, th) \wedge Expression(c) \Rightarrow typeExpr(Boolean, c)$

2. `Il tipo delle espressioni then ed else di un'espressione condizionale deve essere uguale al tipo dell'espressione condizionale`
   $\forall\, e, c, th, el, t\ Expression(e) \wedge ConditionalExpression(e) \wedge Expression(th) \wedge cond(c, e, th) \wedge Expression(c) \wedge else(e, el) \wedge Expression(el) \wedge typeExpr(t, e) \wedge$

$Type(t) \Rightarrow typeExpr(t, th) \wedge typeExpr(t, el)$

3. `Le espressioni coinvolgono solo variabili definite nel modello`
   $\forall\, e, v, m\ Model(m) \wedge Expression(e) \wedge modExpr(m, e) \wedge ExprVar(e) \wedge refers(e, v) \wedge Variable(v) \Rightarrow modVar(m, v)$

4. `Le espressioni coinvolgono solo chiamate a funzioni che sono definite nel modello`
   $\forall\, e, v, m\ Model(m) \wedge Expression(e) \wedge modExpr(m, e) \wedge OperationCall(e) \wedge refers(e, op) \wedge Operation(op) \Rightarrow modOp(m, op)$

1.3 Connection

1. `Variabili coinvolte in connection, devono essere definite nel modulo`

   - $\forall\, m, c, v\ Module(m) \wedge modConn(m, c) \wedge Connection(c) \wedge DirConnection(c) \wedge connInVar(c, v) \wedge Variable(v) \Rightarrow modVar(m, v)$

   - $\forall\, m, c, v\ Module(m) \wedge modConn(m, c) \wedge Connection(c) \wedge DirConnection(c) \wedge connOutVar(c, v) \wedge Variable(v) \Rightarrow modVar(m, v)$

2. `Variabili coinvolte nella stessa connection, non devono essere la stessa variabile`
   $\forall\, m, c, v, v'\ Module(m) \wedge modConn(m, c) \wedge Connection(c) \wedge DirConnection(c) \wedge connInVar(c, v) \wedge Variable(v) \wedge connOutVar(c, v') \wedge Variable(v') \Rightarrow v \neq v'$

3. `Variabili coinvolte nella stessa connection, devono essere dello stesso tipo`
   $\forall\ m, c, v, v', t, t'\ Module(m) \wedge modConn(m, c) \wedge Connection(c) \wedge DirConnection(c) \wedge connInVar(c, v) \wedge Variable(v) \wedge typeExpr(t, v) \wedge Type(t) \wedge connOutVar(c, v') \wedge Variable(v') \Rightarrow typeExpr(t, v')$

1.4 Operation

1. `Il tipo di ritorno di un'operazione deve essere uguale al tipo dell'operazione invocata`
   $\forall\, op, t, o, t'\ OperationCall(op) \wedge typeExpr(t, op) \wedge Type(t) \wedge refers(op, o) \wedge Operation(o) \Rightarrow return(t, o)$

1.5 Event

1. `La trigger condition di un evento deve essere di tipo booleana`
   $\forall\, m, e, c\ Module(m)\ \wedge\ modEvent(m, e)\ \wedge\ Event(e)\ \wedge\ trigger(e, c)\ \wedge\ Expression(c)\ \Rightarrow\ typeExpr(Boolean, c)$

2. `La priority di un evento deve essere numerico`
   $\forall\, m, e, p\ Module(m)\ \wedge\ modEvent(m, e)\ \wedge\ Event(e)\ \wedge\ priority(e, p)\ \wedge\ Expression(p)\ \Rightarrow\ typeExpr(Integer, p)\ \vee\ typeExpr(Real, p)$

3. `Il delay di un evento deve essere numerico`
   $\forall\, m, e, d\ Module(m)\ \wedge\ modEvent(m, e)\ \wedge\ Event(e)\ \wedge\ delay(e, d)\ \wedge\ Expression(d)\ \Rightarrow\ typeExpr(Integer, d)\ \vee\ typeExpr(Real, d)$

1.6 Statement

1. `La condizione di uno statement when deve essere booleana`
   $\forall\, e, th, el, t\ Statement(s) \wedge Block(b) \wedge WhenStatement(s) \wedge Expression(e) \wedge whenCond(s, e, b)\ \Rightarrow\ typeExpr(Boolean, e)$

2. `La condizione di uno statement while deve essere booleana`
   $\forall\, s, e, b\ Statement(s) \wedge WhileStatement(s) \wedge Expression(e) \wedge whileCond(s, e)\ \Rightarrow\ typeExpr(Boolean, e)$

3. `La condizione di uno statement if deve essere booleana`
   $\forall\, s, e, b\ Statement(s)\ \wedge\ IfStatement(s)\ \wedge\ Expression(e)\ \wedge\ Block(b)\ \wedge\ ifCond(s, e, b)\ \Rightarrow\ typeExpr(Boolean, e)$