In Proc. of the *13th European Simulation Symposium, Marseille, 2001, pp. 30-34.*

VARIABLES AND EQUATIONS IN HYBRID SYSTEMS WITH STRUCTURAL CHANGES

D.A. van Beek
Department of Mechanical Engineering
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
E-mail: d.a.v.beek@tue.nl

ABSTRACT

In many models of physical systems, structural changes are common. Such structural changes may cause a variable to change from a differential variable to an algebraic variable, or to a variable that is not defined by an equation at all. Most hybrid modelling languages either restrict the kind of structural changes that may be modelled, or they require several different language elements to model the different kinds of structural changes. This paper proposes a new language element that can be used in combination with conditional equations to model structural changes. The language element is used to declare unknowns in equations, and thus makes a distinction between variables that are unknown in the equations and variables that are not determined by the equations. Examples are given in which unknown declarations are used for modelling steady state initialization, multi-body collision, and higher index systems. Unknown declarations can also be used to clarify the structure of the system of equations, and they can help the modeller detect structurally singular systems of equations.

INTRODUCTION

Many different hybrid modelling and simulation languages exist. Overviews are given in (Mosterman 1999), (Gueguen and Lefebvre 2001), and (Van Beek and Rooda 2000). In hybrid languages, the value of a variable may be determined by equations. These variables are usually referred to as continuous variables. The variables that are not determined by equations are usually referred to as discrete variables. In many models of physical systems, structural changes are common. Such structural changes may cause a variable to change from one category into another. The categories that can be distinguished are the categories of discrete and continuous variables, where the latter category is divided into the subcategories of differential and algebraic variables. Structural changes may also cause the number of equations to change dynamically. Hybrid models are usually initialized by setting the values of the differential variables. Steady state initialization, however, requires a structural change of the equation set immediately after initialization.

Most hybrid modelling languages either restrict the kind of structural changes that may be modelled, or they require several different language elements to model the different kinds of structural changes, e.g. (Barton and Pantelides 1994). In this paper, a single language element is introduced that can be used to model all structural changes that have been treated in this section. The semantics of the language element is treated in an informal way.

BACKGROUND ON DIFFERENTIAL ALGEBRAIC EQUATIONS AND UNKNOWNS

Consider the following general representation of a DAE that is considered on an interval starting at $t = t_0$

$$\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{y}, t) = \mathbf{0}, \tag{1}$$

where $\dot{\mathbf{x}} : I\!R \to I\!R^n$ denotes the n derivatives $\dot{x}_0, \dot{x}_1, \ldots, \dot{x}_{n-1}$, $\mathbf{x} : I\!R \to I\!R^n$ denotes the n differential variables $x_0, x_1, \ldots, x_{n-1}$, $\mathbf{y} : I\!R \to I\!R^m$ denotes the m algebraic variables $y_0, \ldots, y_{m-1}$, $t \in I\!R$ denotes the time, and $\mathbf{f} : ((I\!R \to I\!R^n) \times (I\!R \to I\!R^n) \times (I\!R \to I\!R^m) \times I\!R) \to I\!R^{n+m}$ denotes the set of $n + m$ equations. Mathematically, the solution of differential equations can be divided into two sub-problems that are solved sequentially: the consistent initialization problem (Pantelides 1988), or IP, which aims at solving the equations at a certain point of time $t = t_0$; and the initial value problem, or IVP, which aims at solving the equations for $t \geq t_0$, using the solution of the IP as the starting point.

The IP

The IP can be defined as follows: given information on the initial state $\dot{\mathbf{x}}(t_0), \mathbf{x}(t_0), \mathbf{y}(t_0)$ that is sufficient, in a mathematical sense, to find a unique solution to Equation (1) on an interval starting at $t = t_0$, determine the complete initial state $\dot{\mathbf{x}}(t_0), \mathbf{x}(t_0), \mathbf{y}(t_0)$ corresponding to this unique solution (Brenan et al. 1996). The values $\dot{\mathbf{x}}(t_0), \mathbf{x}(t_0), \mathbf{y}(t_0)$ are referred to as the consistent initial conditions.

The initial conditions $\dot{\mathbf{x}}(t_0), \mathbf{x}(t_0), \mathbf{y}(t_0)$ must satisfy

$$\mathbf{f}(\dot{\mathbf{x}}(t_0), \mathbf{x}(t_0), \mathbf{y}(t_0), t_0) = \mathbf{0}. \tag{2}$$

This is, however, not always enough for consistency of the initial conditions. For certain systems of equations, differentiating a subset of the equations defined by (1) leads to additional equations, also called the hidden constraints, that must also be satisfied by the initial conditions (Pantelides 1988).

The IVP

Given the equation

$$\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{y}, t) = \mathbf{0},$$

combined with the definition

$$\dot{\mathbf{x}}(t) = \lim_{h \to 0} \frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h},$$

determine $\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{y}(t)$ on an interval starting at $t = t_0$ for given consistent initial conditions $\dot{\mathbf{x}}(t_0), \mathbf{x}(t_0), \mathbf{y}(t_0)$.

UNKNOWNS IN THE IP AND IVP

In this section, the IP is further discussed for equations that do not require additional differentiation in order to allow consistent initial conditions to be determined. The variables that are used in the equations of the IP are either known or unknown. In the IP, the values of the unknown variables are determined as a function of the values of the known variables. Therefore, the values of the known variables are not affected by solving the IP. A frequently occurring IP is given by considering $\mathbf{x}(t_0)$ as known, and $\dot{\mathbf{x}}(t_0), \mathbf{y}(t_0)$ as unknown in (2). For ODEs

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t),$$

this IP is trivial: the value of $\dot{\mathbf{x}}(t_0)$ can be calculated by assigning $\mathbf{f}(\mathbf{x}(t_0), t_0)$ to $\dot{\mathbf{x}}(t_0)$. For DAEs, numerical solvers may be needed to solve (2) for $\dot{\mathbf{x}}(t_0), \mathbf{y}(t_0)$. The fact that the value of $\mathbf{x}(t_0)$ remains unchanged when the IP is solved for $\dot{\mathbf{x}}(t_0)$ and $\mathbf{y}(t_0)$, is sometimes referred to as the 'continuity assumption' for $\mathbf{x}$. Another frequently occuring IP is the steady-state IP, which is specified in a general way by

$$\begin{aligned} \mathbf{f}(\dot{\mathbf{x}}(t_0), \mathbf{x}(t_0), \mathbf{y}(t_0), t_0) &= \mathbf{0} \\ \dot{\mathbf{x}}(t_0) &= \mathbf{0}, \end{aligned}$$

with unknowns $\dot{\mathbf{x}}(t_0)$, $\mathbf{x}(t_0)$, and $\mathbf{y}(t_0)$.

In the IVP, the values of the unknown variables are determined as a function of the consistent initial conditions, the time, and the known variables; the values of the known variables are not affected by solving the IVP. The unknown variables for the IVP are usually different from the unknown variables for the IP: the differential variables are always unknown in the IVP, but they can be known in the IP.

Example

Consider a tank with volume V, an inlet feeding a flow of 2, and an outlet with a flow Q. The model of the tank system is:

$$\begin{aligned} \dot{V}(t) &= 2 - Q(t) \\ Q(t) &= \sqrt{V(t)}. \end{aligned}$$

If at time 0, the volume of the tank $V(0)$ is considered as known, the IP is specified by:

$$\begin{aligned} \dot{V}(0) &= 2 - Q(0) \\ Q(0) &= \sqrt{V(0)}, \end{aligned}$$

with $\dot{V}(0)$ and $Q(0)$ as unknowns. When $V(0) = 9$, then $Q(0) = 3$ and $\dot{V}(0) = -1$.

In the case of steady-state initialization, the IP becomes:

$$\begin{aligned} \dot{V}(0) &= 2 - Q(0) \\ Q(0) &= \sqrt{V(0)} \\ \dot{V}(0) &= 0, \end{aligned}$$

where $\dot{V}(0)$, $V(0)$, and $Q(0)$ are the unknowns. This leads to the consistent initial conditions $\dot{V}(0) = 0$, $Q(0) = 2$, and $V(0) = 4$.

For the IVP of the same system, the unknowns are $\dot{V}(t)$, $V(t)$, $Q(t)$, but only two explicit equations instead of three are required:

$$\begin{aligned} \dot{V}(t) &= 2 - Q(t) \\ Q(t) &= \sqrt{V(t)} \end{aligned}$$

since a third, implicit, equation $\dot{V}(t) = \lim_{h \to 0} \frac{V(t+h) - V(t)}{h}$ is also used.

A LANGUAGE ELEMENT FOR THE DECLARATION OF UNKNOWNS

The different kinds of structural changes can be modelled by means of a single language element for the declaration of unknowns. It consists of a list of unknowns between bars, which can be declared wherever equations can be declared. Unknown declarations may also be prefixed to equations. An unknown is a variable of type real, or based on type real (tuples, records, arrays of reals), optionally postfixed with a prime character to denote a derivative.

Informal semantics

A run of a hybrid model is a alternating sequence of discrete and continuous phases, starting with a discrete phase. In both phases, there is a set of unknowns. The variables and derivatives that are in the set of unknowns are considered unknown for the IP and IVP; all other variables are considered known.

In a discrete phase, model time is constant and statements are executed (or actions/events take place). The set of unknowns contains the derivatives that occur in the selected equations, together with the variables that are explicitly declared as unknown in the selected unknown declarations. The selected equations and the set of unknowns define an IP that determines the values of the unknowns as a function of the knowns. The value of an unknown needs to be defined only when the value is actually required in an expression of an executing statement. When no further statements can be executed without advancing model time, the discrete phase terminates and a continuous phase starts.

In a continuous phase, the selected equations and the set of unknowns define an IVP, which defines the values of the unknowns as a function of the consistent initial conditions, the time, and the knowns. The set of unknowns contains the derivatives and the differential variables that occur in the selected equations, together with the variables that are explicitly declared as unknown in the selected unknown declarations. When a time-event or state-event occurs, the continuous phase terminates, and a discrete phase starts.

Derivatives occurring in selected equations, that are by definition present in the set of unknowns, may be explicitly declared as unknown. This does not change the semantics of the model, but it may help to clarify the structure of the system of equations and it may help the modeller to detect structurally singular systems of equations (see Examples).

THE χ LANGUAGE

Unknown declarations are implemented in the newest version of the χ language. An older version of the language is discussed in (Van Beek and Rooda 2000). The χ language has been designed from the start as a hybrid language that can be used for specification, verification, simulation, and real-time control of discrete-event systems, continuous-time systems, and combined discrete-event / continuous-time systems. The language is based on mathematical concepts with well defined semantics (Bos and Kleijn 2000). The discrete-event part of χ is based on Communicating Sequential Processes, the continuous-time part on differential algebraic equations. Processes are parameterized and can be grouped into systems; channels and shared variables are used for inter-process communication and synchronisation. High level data types are available such as arrays, lists and sets along with many associated operators.

The χ simulator is described in (Fábián 1999). It has been successfully applied to a large number of industrial cases, such as an integrated circuit manufacturing plant, a brewery, and a fruit juice blending and packaging plant (Fey 2000). In the sequel, only a minimal subset of the χ language is used. The syntax and operational semantics of the language constructs are explained in an informal way. The models are kept as short as possible, showing only the essentials. This implies that most models are specified as stand-alone processes.

Process syntax

A process may consist of a continuous-time part only (differential algebraic equations), a discrete-event part (statements) only, or a combination of both. The syntax of a process consisting of equations and statements is:

`proc` *name*(*parameter declarations*) =
`|[` *variable declarations*
 `{` *equations* `}`
 `|` *statements*
`]|`

Equation syntax

A somewhat simplified syntax of equations in χ follows below. The syntax is described in extended BNF where $\{X\}$ denotes zero or more repetitions of construct X, and $[X]$ denotes zero or one occurrence of construct X. Literal characters are enclosed in quotes.

```
unknown     ::= variable | variable primechar
unknownDecl ::= '|' unknown {, unknown} '|'
guardedAlt  ::= b '->' Eq | b '->' empty
              | guardedAlt '|' guardedAlt
Eq          ::= [unknownDecl] re '=' re
              | [unknownDecl] '[' guardedAlt ']'
              | unknownDecl
              | Eq,Eq
EqSet       ::= '{' Eq '}'
```

where `b` denotes a boolean expression; `empty` denotes an empty equation; `re` denotes an expression of type `real` (or based on `real`), and `primechar` denotes the prime character.

EXAMPLES

The semantics is illustrated by means of the following example (comments are prefixed by `//`):

```
proc P =
|[ V,x: real:= 0
 , Q: real
 , b: bool:= true
 {      V' = 2 - Q          // Eq 0
 , |Q|  Q  = sqrt (V)       // Eq 1
 , [ b     ->     x' = 1 - x // Eq 2
   | not b -> |x| x  = 2     // Eq 3
   ]
 }
]|
```

Variables `V`, `x`, and `Q` are declared as reals. Function `sqrt` represents the square root function. Variable `b` is a boolean variable, that is initialized to true. When `b` is true, the set of selected equations consist of Equations 0, 1, and 2. The unknowns for the IP are then `V'`, `x'` (by definition), and `Q` (explicitly declared as unknown). This means that the values of `V` and `x` remain unchanged when the consistent initial conditions are determined by solving the IP. The unknowns for the IVP are `V'`, `x'` (by definition), `V`, `x` (by definition), and `Q` (explicitly declared as unknown).

When boolean variable `b` is false, the set of selected equations consist of Equations 0, 1, and 3. Variable `x` is now an algebraic variable. The unknowns for the IP are `V'`, `Q`, and `x`. The unknowns for the IVP are `V'`, `V`, `Q`, and `x`.

Steady state initialization

Consider again the tank system example:

```
proc BO( ref steady: bool
       , ref V: real, ref Q: real
       ) =
|[
 { [     steady -> |V'| V' = 0
                 , |Q|  V' = 2 - Q
                 , |V|  Q  = sqrt(V)
   | not steady -> |V'| V' = 2 - Q
                 , |Q|  Q  = sqrt(V)
   ]
 }
]|
```

Boolean variable `steady` is true when steady-state initialization is used; it is false when the value of differential variable `V` should be unaffected by solving the IP. The keyword `ref` can be prefixed to a process parameter. The meaning is that such a parameter is a shared variable: such a variable can be used in more than one process.

If boolean variable `steady` is true, the set of selected equations and unknown declarations is

```
|V'| V' = 0, |Q| V' = 2 - Q, |V| Q = sqrt(V)
```

with unknowns `V'`, `Q`, and `V`. If boolean variable `steady` is false, the set of selected equations and unknown declarations is

```
|V'| V' = 2 - Q, |Q| Q  = sqrt(V)
```

with `V'` and `Q` unknown for the IP, and `V` known for the IP, so that the value of differential variable `V` remains unchanged when the IP is solved.

A shorter, but functionally equivalent specification is:

```
proc B1( ref steady: bool
       , ref V: real, ref Q: real
       ) =
|[
 { [     steady -> |V|, V' = 0
   | not steady ->
   ]
 , |Q|
 , V' = 2 - Q,
 , Q = sqrt(V)
 }
]|
```

Modelling colliding bodies

Consider a body of mass 1, with applied force 1, moving towards another body of mass 1, and colliding with that body. For a perfectly elastic collision, the energy and impulse conservation laws can be applied. The positions of the bodies are denoted by `x.0` and `x.1`, respectively; `x` is declared as an array of two elements of type `pos` (`x: pos^2`). The velocities of the bodies are modelled by array `v`. Variable `vold` is an array of two elements that contains the velocities of the two bodies just before the collision. Comments are preceded by `//`.

```
type pos = real    // position
,    vel = real    // velocity

proc C =
|[ x: pos^2:= <0.0, 1.0>
 , v: vel^2:= <0.0, 0.0>, vold: vel^2
 , collision: bool:= false
 { v' = < 1.0, 0.0 >
 , x' = v
 , [ collision
     -> |v|
      , v.0^2 + v.1^2 = vold.0^2 + vold.1^2
      , v.0 + v.1     = vold.0  + vold.1
   | not collision
     ->
   ]
 }
| *[ nabla x.0 >= x.1; vold:= v
   ; collision:= true; collision:= false
   ; nabla x.0 < x.1
   ]
]|
```

In the declaration part of the specification, the positions of the bodies are initialized to 0 and 1, respectively; the velocities are initialized to 0, and the boolean variable collision is set to false. The meaning of `nabla x.0 >= x.1`, as the first statement of the repetition `*[ ... ]`, is that the process waits until the value of expression `x.0 >= x.1` becomes true. This is the case when the collision occurs. Subsequently, `vold` is set to `v`. When `collision` becomes true, `v` becomes unknown, and the two equations that specify the conservation of energy and impulse determine the value of `v`. When `collision` becomes false again, `v` changes from an unknown into a known variable for the IP. Therefore, when `collision` has become false, the two additional equations are no longer in the set of selected equations, and the value of `v` is the value that was determined by the IP with the two additional equations.

Higher index systems

Structural regularity of equations with respect to $[\dot{\mathbf{x}}\ \mathbf{y}]^T$ is a necessary requirement to allow consistent initialization without differentiation (Pantelides 1988). The examples show how unknown declarations can be used to assign each derivative and algebraic variable to a unique equation. If this is not possible, structural singularities can be made visible. As an example of a higher index system, take the following PID (proportional integral differential) controller. A horizontal force `F` is applied to a body of mass 1 on a flat surface, without friction. The position and velocity of the body are denoted by `x`, and `v`, respectively. The control objective is to keep the body at a given position `xset`. The control error, integral of the error, and the control output are denoted by `e`, `i`, and `u`, respectively. The model is:

```
type pos   = real
,    vel   = real
,    force = real

proc PID0( kP,kD,KI: real, xset: pos
         , Fin: real -> force
         )=
|[ x,e: pos, v: vel, F: force, i,u: real
 { |u,F|
 , x' = v
 , v' = F - u
 ,  e = x - xset
 , i' = e
 ,  u = kP * e + kD * e' + kI * i
 ,  F = Fin(time)
 }
 | x:= 0.0; v:= 0.0; i:= 0.0; e:= x- xset
]|
```

where `time` denotes the current time of the model, `kP`, `kD`, `kI` are parameters, and parameter `Fin` denotes a function from a real (the time) to a force. The derivatives (`x'`, `v'`, `i'`, `e'`) are unknown by definition. The algebraic variables `u` and `F` are explicitly declared as unknown at the start of the equation set. When initializing `e`, a problem appears: `e` is a differential variable, but its value cannot be initialized independently from `x`. By explicitly declaring all IP unknowns and prefixing them to the equations in which they occur, the structural singularity becomes apparent: the derivatives and algebraic variables cannot each be assigned to a unique equation; the system is an index 2 system, and consistent initialization is only possible after differentiation of equation `e = x - xset`:

```
proc PID1( kP,kD,KI: real, xset: pos
         , Fin: real -> force
         )=
|[ x,e: pos, v: vel, F: force, i,u: real
 { |x'|   x' = v
 , |v'|   v' = F - u
 ,          e = x - xset // Structural singularity!
 , |i'|   i' = e
 , |e',u|  u = kP * e + kD * e' + kI * i
 , |F|     F = Fin(time)
 }
 | x:= 0.0; v:= 0.0; i:= 0.0; e:= x- xset
]|
```

The index can be reduced from 2 to 1 by differentiation of equation `e = x - xset`, which leads to `e' = x'`, and using a substitute equation (Fábián et al. 2001) for `e'`. In this way, `e` is no longer a differential variable and becomes an algebraic variable. The meaning of a substitute equation is that every time the value of the variable on the left hand side is used in the model, the value of the right hand side expression is substituted.

```
proc PID2(kP,kD,KI,xset: real, Fin: real -> real)=
|[ x,e: position, v: vel, F: force, i,u: real
 { |x'| x' = v
 , |v'| v' = F - u
 , |e|   e = x - xset
 , |i'| i' = e
 , |u|   u = kP * e + kD * e' + kI * i
 , |F|   F = Fin(time)
 ,     e' <- x'       // substitute e' by x'
 }
 | x:= 0.0; v:= 0.0; i:= 0.0
]|
```

ACKNOWLEDGMENT

The authors would like to thank Victor Bos and Jeroen Kleijn for their idea to explicitly specify the difference between unknown and known variables in equations.

CONCLUSIONS

The new language element for the declaration of unknowns has been shown to be a versatile language element. It is used to make a distinction between variables that are unknown in the equations and variables that are not determined by the equations. In combination with conditional equations, all kinds of structural changes in the equations can be modelled. In addition, the declaration of unknowns can help to clarify the structure of the system of equations, and it can help the modeller detect structurally singular (high index) systems of equations.

REFERENCES

Barton, P. I. and C. C. Pantelides. 1994. Modeling of Combined Discrete/Continuous Processes. *AIChE 40*(6), 966–979.

Bos, V. and J. J. T. Kleijn. 2000. Formalisation of a Production Systems Modelling Language: the Operational Semantics of χ Core. *Fundamenta Informaticae 41*(4), 367–392.

Brenan, K. E.; S. L. Campbell; and L. R. Petzold. 1996. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM's Classics in Applied Mathematics. Philadelphia: Siam.

Fábián, G. 1999. *A Language and Simulator for Hybrid Systems*. Ph. D. thesis, Eindhoven University of Technology.

Fábián, G.; D. A. van Beek; and J. E. Rooda. 2001. Index Reduction and Discontinuity Handling using Substitute Equations. *Mathematical and Computer Modelling of Dynamical Systems*. To be published.

Fey, J. J. H. 2000. *Design of a Fruit Juice Blending and Packaging Plant*. Ph. D. thesis, Eindhoven University of Technology.

Gueguen, H. and M. A. Lefebvre. 2001. A Comparison of Mixed Specification Formalisms. *APII JESA Journal Europeen des Systemes Automatises 35*(4), 381–394.

Mosterman, Pieter J. 1999. An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages. In *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science 1569, 165–177. Springer.

Pantelides, C. C. 1988. The Consistent Initialization of Differential-Algebraic Systems. *SIAM J. Sci. Stat. Comput. 9*(2), 213–231.

Van Beek, D. A. and J. E. Rooda. 2000. Languages and Applications in Hybrid Modelling and Simulation: Positioning of Chi. *Control Engineering Practice 8*(1), 81–91.