In *Les Systèmes Dynamiques Hybrides / Hybrid Dynamical Systems—Proc. of 3rd International Conference on Automation of Mixed Processes*, Reims, 1998, pp. 415-420.

SPECIFICATION OF DISCONTINUITIES IN HYBRID MODELS

SPECIFICATION DES DISCONTINUITES DANS LES MODELES HYBRIDES

D.A. VAN BEEK*, J.E. RODA**

Eindhoven University of Technology, Dept. Mech. Eng.
POB 513, 5600 MB Eindhoven, The Netherlands
** Tel: +31-(0)40-2472892, E-mail: vanbeek@wtb.tue.nl*
URL: http://asterix.urc.tue.nl/~vanbeek
*** Tel: +31-(0)40-2474553, E-mail: rooda@wtb.tue.nl*

Abstract: Two types of discontinuities are distinguished: D and D' discontinuities. The facilities for discontinuity specification in the combined continuous-time / discrete-event χ language are treated. Although the language consists of a relatively small number of language elements, discontinuities can be elegantly specified in many ways. This is illustrated by means of examples. D discontinuities cannot be handled by the differential algebraic equation solver used in the χ simulator. By means of techniques such as substitution, variables that cause the discontinuity can be removed.

Résumé: Deux types de discontinuités se distinguent: les discontinuités D et les discontinuités D'. Les mécanismes mis en place par le langage hybride χ pour spécifier les discontinuités sont présentés. Bien que ce langage comporte un nombre limité de primitives, il fournit des mécanismes élégants et variés pour la spécification des discontinuités. Le 'résolveur' d'équations différentielles algébriques employé dans le simulateur n'est pas capable de traiter les discontinuités D. Des techniques telle que la substitution peuvent être employées pour éliminer les variables engendrant ces discontinuités.

1. INTRODUCTION

An important aspect of modelling hybrid systems is the specification of discontinuities. Physical systems normally do not exhibit discontinuous behaviour. Models of such systems are abstractions. Detail is omitted in order to reduce the complexity of the models. In this way, discontinuities can be introduced into the models. Many languages that have facilities for specification of discontinuities are continuous-time languages to which constructs for discontinuity specification have been added. Two examples are ACSL [MG 76] and Dymola [Elm 94]. The χ language is based on a different design philosophy. It consists of two parts: a part for specification of continuous-time systems, and a part for specification of discrete-event systems. By using combinations of the discrete-event and continuous-time language constructs, different types of discontinuities can be easily specified. This paper defines different types of discontinuities and treats the facilities for discontinuity specification of the χ language. The paper also gives insight into how discontinuities affect the operation of a simulator, and treats how discontinuities can be successfully handled.

2. DEFINITION OF DISCONTINUITIES

First, the definition of a discontinuous function is given. We write $\lim_{x \uparrow x_d} f(x)$ as $f(x_d^-)$, and $\lim_{x \downarrow x_d} f(x)$ as $f(x_d^+)$. A function $f(x)$ is discontinuous if a value x_d exists for which $f(x_d^-)$ and $f(x_d^+)$ exist, and $f(x_d^-) \neq f(x_d^+)$. In [CEOT 93], such a function $f(x)$ is referred to as a D-function, whereas a continuous function $g(x)$ with a discontinuous derivative $h(x) = \frac{\partial g(x)}{\partial x}$ is referred to as a DD-function. In this paper, we use the term D' function instead of DD-function. In a similar way, D'' functions (and so on) could be defined, but they are of little practical importance. Second, two kinds of variables are distinguished: continuous and discrete variables. The value of a continuous variable is determined mainly by equations, whereas the value of a discrete variable is determined by assignments.

The value of a variable can be represented by a function defined on time. Consider a continuous variable c, the value of which is represented by function $c(t)$ (where t represents time). Variable c is D-continuous or D'-continuous if $c(t)$ is a D function or D' function, respectively. The function $d(t)$ that represents the value

of a discrete variable d is always a D function (unless it is constant), because discrete variables change at discrete points of time only.

Discontinuities are relevant for models with continuous variables. Such a model has a D or D' discontinuity if a time point t_d and a (D- or D'-) continuous variable c exist for which $c(t_d^-) \neq c(t_d^+)$ or $c'(t_d^-) \neq c'(t_d^+)$, respectively. The discontinuity occurs at $t = t_d$ when $c(t)$ changes from $c(t_d^-)$ to $c(t_d^+)$, or $c'(t)$ changes from $c'(t_d^-)$ to $c'(t_d^+)$. A model is also said to have a $D^{(')}$ discontinuity if it has a continuous variable that can be represented as a $D^{(')}$ function of another continuous variable (see Section 4.2).

3. INTRODUCTION TO THE χ LANGUAGE

3.1 *Design*

The χ language has only one language construct that has been designed especially for discontinuity handling: guarded equations, which are treated in the next section. All other language elements are general purpose constructs that are required for continuous-time and discrete-event systems specification. The language has been designed from the start as a hybrid language that can be used for modelling, simulation and control of discrete-event systems, continuous-time systems and combined discrete-event / continuous-time systems.

A small number of flexible, orthogonal (elements can be freely combined and have no overlap) discrete-event and continuous-time χ language constructs allow different types of discontinuities to be specified in an elegant way. Interaction between the continuous-time and discrete-event part plays a central role in discontinuity specification. This is a different approach than the one taken in the gPROMS language for example, where the continuous-time part has been designed for physical systems specification, and the discrete-event part for operating procedure modelling. As a result, the continuous-time part of gPROMS has two language elements for physicochemical discontinuity specification: one for specification of 'reversible' discontinuities, and one for specification of 'irreversible' and 'asymmetric and reversible' discontinuities [BP 94].

3.2 *Syntax and semantics*

In this paper, a small subset of the language is treated. We do not treat the language elements for modelling parallel processes that interact by means of message passing and synchronization. For these and other aspects of the χ language, we refer the reader to [BRG 96,BGR 97].

A process may consist of a continuous-time part only (*DAEs*: differential algebraic equations), a discrete-event part only, or a combination of both.

proc *name*(*parameter declarations*) =
⟦ *variable declarations* ; *initialization*
| *DAEs* | *discrete-event statements*
⟧

All data types and variables are declared as either continuous or discrete. The value of a discrete variable is determined by assignments (e.g. $n := 0$). Between two subsequent assignments, the variable retains its value. The value of a continuous variable, on the other hand, is determined by equations. An assignment to a continuous variable (e.g. $v ::= 0$) determines its value for the current point of time only. Some discrete data types are predefined such as bool (boolean), int (integer) and real. Since all continuous variables are assumed to be of type real, they are defined by specifying their units only. For example, the declaration v : [m/s] defines a continuous variable v.

Differential algebraic equations are separated by commas: $DAE_1, DAE_2, \ldots, DAE_n$. A DAE can be a normal equation or a *guarded equation*. The latter is used when the set of equations depends on the state of the system. The syntax is $[\, b_1 \longrightarrow DAEs_1 \,[]\, \ldots \,[]\, b_n \longrightarrow DAEs_n \,]$. $DAEs_i$ $(1 \leq i \leq n)$ represents one or more DAEs. The boolean expression b_i denotes a *guard*. At any time, at least one of the guards must be open (true), so that the DAEs $DAEs_i$ associated with an open guard can be selected.

The discrete-event part of χ is a CSP-like real-time concurrent programming language for which we refer to [MFRN 95]. *Time passing* is denoted by $\Delta\, t$, where t is an expression of type real. A process executing this statement is blocked until the time is increased by t time-units. *Repetition* of statement S is denoted by $*[\ S\ ]$.

The χ *selection* statement is a guarded command $[\, b_1 \longrightarrow S_1 \,[]\, \ldots \,[]\, b_n \longrightarrow S_n \,]$. The boolean expression b_i $(1 \leq i \leq n)$ denotes a *guard*, which is open if b_i evaluates to true and is otherwise closed. After evaluation of the guards, one of the statements S_i associated with an open guard b_i is executed.

By means of the *state event* statement $\nabla\, r$, the discrete-event part of a process can synchronize with the continuous-time part of a process. Execution of $\nabla\, r$, where r is a relation involving at least one continuous variable, causes the process to be blocked until the relation becomes true.

4. DISCONTINUITY SPECIFICATION IN χ

Discontinuities can be specified in χ by means of:

(1) Assignments to discrete variables occurring in equations
(2) Guarded equations using continuous variables in the guards
(3) Functions

(4) Assignments to differential variables.

4.1 *Assignments to discrete variables occurring in equations*

A simple tank level control system is used as an example. The level of the liquid in the vessel is kept close to the set point by switching the valve on and off. Liquid flows out of the tank due to gravity. By means of statements in the discrete-event part, hysteresis can be modelled (see process TCD_1).

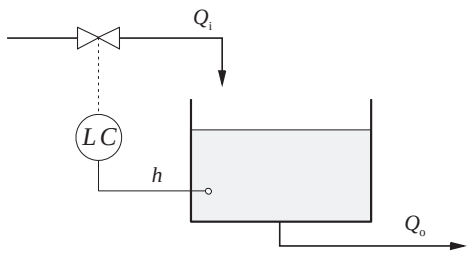

Fig. 1. The tank level control system.

D′ discontinuities

The specification of the tank level process TCD_1 follows below (ρ and g are constants). The discrete variable n gets the values 1 and 0 in the discrete-event part. As a result of this, the equation $Ah' = nQ_{\text{set}} - Q_o$ changes from $Ah' = Q_{\text{set}} - Q_o$ to $Ah' = -Q_o$. Therefore, variable h is D'-continuous. This way of specifying discontinuities leads to short and clear models. In the specification below, the incoming flow is switched on when $h < h_{\text{set}} - h_{\text{hys}}$ and off when $h > h_{\text{set}}$. The hysteresis thus equals h_{hys}.

proc $TCD_1(A, h_0, h_{\text{set}}, h_{\text{hys}}, k, Q_{\text{set}} : \text{real}) =$
⟦ h : [m], Q_o : [m^3/s], n : int
; $h ::= h_0$; $n := 0$
| $Ah' = nQ_{\text{set}} - Q_o$
, $Q_o = k\sqrt{\rho g h}$
| ∗[$\nabla h < h_{\text{set}} - h_{\text{hys}}$; $n := 1$
; $\nabla h > h_{\text{set}}$; $n := 0$
]
⟧

Since the only values used for variable n are 1 and 0, a boolean variable can also be used. Such a boolean variable requires the use of a guarded equation as shown below. The boolean variable b is true when the valve is open and false when it is closed. This way of specifying discontinuities is more flexible, because several equations can be specified after the arrows of the guarded equation (e.g. see process *TOV* in Section 4.2).

proc $TCD_2(A, h_0, h_{\text{set}}, h_{\text{hys}}, k, Q_{\text{set}} : \text{real}) =$
⟦ h : [m], Q_o : [m^3/s], b : bool
; $h ::= h_0$; $b :=$ false
| [$b \longrightarrow Ah' = Q_{\text{set}} - Q_o$
[] $\neg b \longrightarrow Ah' = -Q_o$
]
, $Q_o = k\sqrt{\rho g h}$
| ∗[$\nabla h < h_{\text{set}} - h_{\text{hys}}$; $b :=$ true
; $\nabla h > h_{\text{set}}$; $b :=$ false
]
⟧

D discontinuities

D discontinuities can be introduced in the specifications by introducing a continuous variable Q_i for the incoming flow. This gives three continuous variables (h : [m], Q_i, Q_o : [m^3/s]). The equation $Ah' = nQ_{\text{set}} - Q_o$ of process TCD_1 is then replaced by two equations:

$Ah' = Q_i - Q_o$
, $Q_i = nQ_{\text{set}}$

Variable Q_i is D-continuous, because it changes from 0 to Q_{set} when n changes from 0 to 1.

Process TCD_2 can be changed in a similar way by introducing the continuous variable Q_i, and replacing the guarded equation for Ah' by the following two equations:

$Ah' = Q_i - Q_o$
, [$b \longrightarrow Q_i = Q_{\text{set}}$
[] $\neg b \longrightarrow Q_i = 0$
]

4.2 *Guarded equations using continuous variables in the guards*

In this case, the discontinuity is specified in the continuous-time part of the language. Discrete-event statements are not required. The specification of hysteresis in this way is, however, not possible.

D′ discontinuities

In this example, the on/off valve used in the previous section is replaced by a valve that can be adjusted linearly between fully open and fully closed. The controller is now a continuous-time controller instead of a discrete-event controller. The model of the complete system is specified below. The equation for Q_o has been removed by substitution.

proc $TCPC(A, h_0, h_{\text{set}}, k, K_p, Q_{\text{set}} : \text{real}) =$
⟦ h : [m], Q_i : [m^3/s], u : [−]
; $h ::= h_0$
| $Ah' = Q_i - k\sqrt{\rho g h}$
, $u = K_p(h_{\text{set}} - h)$
, [$u < 0$ $\longrightarrow Q_i = 0$
[] $0 \le u \le 1 \longrightarrow Q_i = uQ_{\text{set}}$
[] $u > 1$ $\longrightarrow Q_i = Q_{\text{set}}$
]
⟧

The tank is represented by $Ah' = Q_\mathrm{i} - k\sqrt{\rho g h}$, and the controller by $u = K_\mathrm{p}(h_\mathrm{set} - h)$. The valve is represented by the guarded equation

$$
\begin{array}{ll}
[\ u < 0 & \longrightarrow\ Q_\mathrm{i} = 0 \\
[]\ 0 \le u \le 1 & \longrightarrow\ Q_\mathrm{i} = u\,Q_\mathrm{set} \\
[]\ u > 1 & \longrightarrow\ Q_\mathrm{i} = Q_\mathrm{set} \\
\ ]
\end{array}
$$

In this example, continuous variable Q_i can be represented as a D' function of u.

D discontinuities

Specification of D discontinuities is explained by means of a tank, represented in Fig. 2, that flows over when the level h of the liquid in the tank is higher than the maximum level h_max. The χ specification follows below. Variable Q_x is D-continuous, and it can be represented as a D function of h.

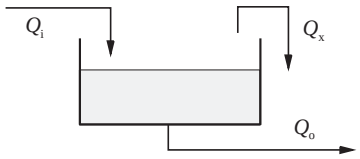

Fig. 2. Tank with overflow.

$$
\begin{array}{l}
\mathsf{proc}\ TOV(A, h_0, h_\mathrm{max}, Q_\mathrm{i}, k : \mathrm{real}) = \\
[\![\ h : [\mathrm{m}],\ Q_\mathrm{o}, Q_\mathrm{x} : [\mathrm{m}^3/\mathrm{s}] \\
;\ h ::= h_0 \\
|\ [\ h < h_\mathrm{max} \vee Q_\mathrm{i} < Q_\mathrm{o} \longrightarrow Ah' = Q_\mathrm{i} - Q_\mathrm{o} \\
\qquad\qquad\qquad\qquad\quad ,\ Q_\mathrm{x} = 0 \\
\quad []\ h \ge h_\mathrm{max} \wedge Q_\mathrm{i} \ge Q_\mathrm{o} \longrightarrow Ah' = 0 \\
\qquad\qquad\qquad\qquad\quad ,\ Q_\mathrm{x} = Q_\mathrm{i} - Q_\mathrm{o} \\
\quad] \\
,\ Q_\mathrm{o} = k\sqrt{\rho g h} \\
]\!]
\end{array}
$$

4.3 *Functions*

Discontinuities can also be present in explicitly defined χ functions. Consider for example the valve from process *TCPC* in Section 4.2. This valve can also be modelled by means of a (D') χ function. The function is shown below.

$$
\begin{array}{l}
\mathsf{func}\ \mathrm{clip}(x_\mathrm{min}, x, x_\mathrm{max} : \mathrm{real}) \to \mathrm{real} = \\
[\![\ [\ x \le x_\mathrm{min} \longrightarrow \uparrow x_\mathrm{min} \\
\quad []\ x_\mathrm{min} \le x \le x_\mathrm{max} \longrightarrow \uparrow x \\
\quad []\ x \ge x_\mathrm{max} \longrightarrow \uparrow x_\mathrm{max} \\
\quad] \\
]\!]
\end{array}
$$

Statement $\uparrow x$ means: return the value of x. Using the function clip, the equations of process *TCPC* can be simplified to

$$
\begin{array}{l}
\quad Ah' = Q_\mathrm{i} - k\sqrt{\rho g h} \\
,\quad u = K_\mathrm{p}(h_\mathrm{set} - h) \\
,\ Q_\mathrm{i} = \mathrm{clip}(0, u, 1) \cdot Q_\mathrm{set}
\end{array}
$$

The clip function simplifies the process equations. It models the saturation phenomenon that often occurs in models. The function needs to be defined only once and can then be used in different processes.

4.4 *Assignments to differential variables*

Modelling is specifying a system in an abstract way. The level of abstraction is determined by the amount of detail the modeller wants to include in the model. Consider for example a tank. Filling and emptying of the tank can be described by means of a differential equation. If the modeller prefers to specify the filling of the tank in a more abstract way with less detail, assignments to differential variables can be used. In the following specification, a tank is repeatedly filled and emptied.

$$
\begin{array}{l}
\mathsf{proc}\ Tk(V_\mathrm{set}, Q_\mathrm{setin}, Q_\mathrm{setout} : \mathrm{real}) = \\
[\![\ V : [\mathrm{m}^3],\ n : \mathrm{int} \\
;\ V ::= 0;\ n := 0 \\
|\ V' = -n\,Q_\mathrm{setout} \\
|\ *[\ V ::= V_\mathrm{set};\ \Delta\, V/Q_\mathrm{setin} \\
\quad ;\ n := 1;\ \nabla\, V = 0;\ n := 0 \\
\quad] \\
]\!]
\end{array}
$$

Filling of the tank is modelled by means of a discontinuity: the volume is instantly increased to V_set ($V ::= V_\mathrm{set}$). Subsequently, time passing of the filling operation is modelled in the delta statement $\Delta\, V/Q_\mathrm{setin}$. Reversing the order of the two statements is also possible ($\Delta\, V/Q_\mathrm{setin}\,;\ V ::= V_\mathrm{set}$). Emptying of the tank is modelled using the differential equation $V' = -n\,Q_\mathrm{setout}$.

4.5 *Combinations*

A somewhat more complex example, that combines assignments to discrete variables occurring in equations and assignments to differential variables (Sections 4.1 and 4.4), deals with modelling of dry friction as shown in Fig. 3.

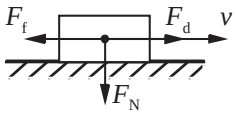

Fig. 3. Dry friction.

A driving force F_d ($F_\mathrm{d} = \sin 0.25\pi\tau$, where τ is the current time) is applied to a body on a flat surface with frictional force F_f. When the body is moving with positive velocity v, the frictional force is given by $F_\mathrm{f} = \mu F_\mathrm{N}$, where $F_\mathrm{N} = mg$. When the velocity of the body is 0, the frictional force neutralizes the applied driving force. If the driving force becomes bigger than $\mu_0 F_\mathrm{N}$, the body suddenly starts moving according to $F_\mathrm{d} - F_\mathrm{f} = mv'$,

where $F_f = \mu F_N$ ($\mu < \mu_0$). In process *Friction*$_1$, defined below, discrete variable s represents the state of the process; it can have the values NEG, STOP, and POS. These values correspond with negative, zero, and positive velocities respectively of the body. As a result of the different equations for the three states, variable F_f is D-continuous.

```
proc Friction1(ε, μ, μ0, m : real) =
⟦ Ff, Fd, FN : [N], x : [m], v : [m/s]
, s : {NEG, STOP, POS}
; x ::= −2; v ::= 0; s := STOP
| Fd = sin 0.25πτ, FN = mg
, v' = (Fd − Ff)/m, x' = v
, [ s = NEG  ⟶ Ff = −μFN
  [] s = STOP ⟶ Ff = Fd
  [] s = POS  ⟶ Ff = μFN
  ]
| ∗[ s = STOP; ∇ Fd > μ0FN   ⟶ s := POS
                              ; v ::= ε
   [] s = STOP; ∇ Fd < −μ0FN ⟶ s := NEG
                              ; v ::= −ε
   [] s ≠ STOP; ∇ v = 0      ⟶ s := STOP
   ]
⟧
```

The discrete-event part of process *Friction*$_1$ consists of a selective waiting statement ([... [] ... [] ...], e.g. see [BGR 97]) that is repeated forever ($*$). If boolean expression $s =$ STOP is true and state event $\nabla F_d > \mu_0 F_N$ succeeds, then the state switches to POS ($s :=$ POS), causing a different equation to be active in the continuous-time part. The velocity v is then assigned a very small value ε in order to prevent the state event $\nabla v = 0$ from occurring immediately. Now that the state equals POS, boolean expression $s \neq$ STOP is true. This means that state event $\nabla v = 0$ is awaited. When v becomes equal to 0, the state changes to STOP again.

Process *Friction*$_2$, that follows below, is a more detailed model. It models that a small amount of energy is required to switch from state STOP to state POS or NEG. For this purpose, a fourth state TRY is introduced. If $|F_d|$ becomes bigger than $\mu_0 F_N$, the state switches to TRY. In this state, the frictional force equals $\mu_0 F_N$. If the driving force causes the velocity of the body to become bigger than ε (a small user defined value, e.g. 10^{-5}), the state switches to POS. If, however, the driving force falls back below $\mu_0 F_N$, the state switches back to STOP. Fig. 4 shows the results of a 10 second simulation run for the process *Friction*$_1$ and *Friction*$_2$. The difference between the results of the two processes is too small to be shown graphically.

```
proc Friction2(ε, μ, μ0, m : real) =
⟦ Ff, Fd, FN : [N], x : [m], v : [m/s]
, s : {NEG, STOP, TRY, POS}
; x ::= −2; v ::= 0; s := STOP
| Fd = sin 0.25πτ, FN = mg
, v' = (Fd − Ff)/m, x' = v
, [ s = NEG  ⟶ Ff = −μFN
  [] s = STOP ⟶ Ff = Fd
  [] s = TRY  ⟶ Ff = sign(Fd) · μ0FN
  [] s = POS  ⟶ Ff = μFN
  ]
| ∗[ s = STOP; ∇ |Fd| > μ0FN
      ⟶ s := TRY
      ; [ ∇ v > ε          ⟶ s := POS
        [] ∇ v < ε         ⟶ s := NEG
        [] ∇ |Fd| ≤ μ0FN   ⟶ s := STOP; v ::= 0
        ]
   [] s ≠ STOP; ∇ v = 0 ⟶ s := STOP
   ]
⟧
```

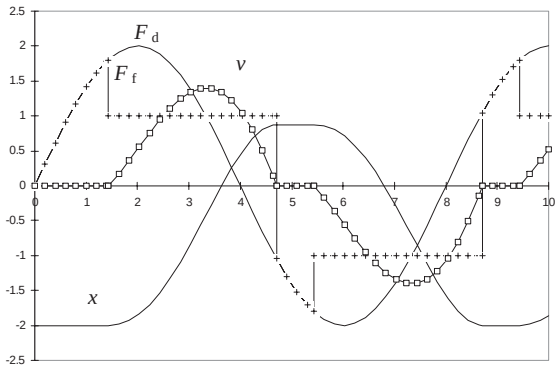

Fig. 4. Friction simulation ($\mu = 0.1$, $\mu_0 = 0.18$, $m = 1$, $\varepsilon = 10^{-5}$, $g = 10$).

5. DISCONTINUITY HANDLING IN THE χ SIMULATOR

Simulation of a hybrid model is a repetition of two alternating phases: the 'discrete-event phase' which involves execution of statements at the current discrete-event time point, and the 'continuous-time phase' which involves solving of equations while time is advanced to the next discrete-event time point. At the end of a discrete-event time point, when no more statements can be executed, a non linear equation solver is called. This solver calculates a consistent initial state for the beginning of the subsequent continuous-time phase. This entails, among other things, calculation of the values of the algebraic variables. In the continuous-time phase, a differential algebraic equation solver (DASSL, [Pet 83]) calculates the continuous variables as a function of time until the next discrete-event time point. Discontinuities can occur either in the discrete-event phase or in the continuous-time phase.

Discontinuities (D' and D) that occur in the discrete-event phase are caused by assignments to discrete or differential variables (Sections 4.1, and 4.4). In this case, the non linear equation solver, called at the end of the discrete-event phase, calculates a new consistent initial state for the continuous-time phase.

A discontinuity in the continuous-time phase is caused by a guarded equation (e.g. [$h < h_{\max} \longrightarrow Q_x = 0$ [] $h \geq h_{\max} \longrightarrow Q_x = Q_i - Q_o$]) in which a different set of equations becomes valid, or by a function (Section 4.3). The discontinuity is a result of guards ($h < h_{\max}$ and $h \geq h_{\max}$) changing from true to false or vice versa, because a continuous variable (h) crosses a predefined boundary ($h_{\max}$). D' discontinuities in the continuous-time phase (Sections 4.2 and 4.3) can be handled by the DASSL differential algebraic equation solver. Approaching a discontinuity, it reduces the step size until the exact time point of the discontinuity is found. After the discontinuity, the step size is gradually increased. More efficient techniques, that use the root finding algorithm of the solver to quickly find the discontinuity, are possible, but these are not treated here. D discontinuities in the continuous-time phase (Section 4.2) cannot be so easily handled. The DASSL solver, like most other differential equation solvers, cannot handle D discontinuities. Therefore, these D discontinuities must first be removed. How this can be done is treated in the next section.

5.1 *Removing D discontinuities*

D discontinuities can be removed by removing the D-continuous variable by substitution. Consider, for example, process *TOV* in Section 4.2. Variable Q_x should not be given to the DASSL solver. At any time that the value of Q_x is needed, the value of a substitution expression can be used. For this purpose, equations

$$
\begin{array}{ll}
[\ h < h_{\max} \vee Q_i < Q_o \longrightarrow & Ah' = Q_i - Q_o \\
\quad , & Q_x = 0 \\
[]\ h \geq h_{\max} \wedge Q_i \geq Q_o \longrightarrow & Ah' = 0 \\
\quad , & Q_x = Q_i - Q_o \\
]
\end{array}
$$

are rewritten as

$$
\begin{array}{l}
Ah' = Q_i - Q_o - Q_x \\
, \ Q_x \leftarrow [\ h < h_{\max} \vee Q_i < Q_o \longrightarrow 0 \\
\qquad [] \ h \geq h_{\max} \wedge Q_i \geq Q_o \longrightarrow Q_i - Q_o \\
\qquad]
\end{array}
$$

The use of a left arrow instead of the equal sign indicates a substitution equation. The meaning of the substitution equation for Q_x is that whenever the value of Q_x is needed, 0 is substituted when $h < h_{\max} \vee Q_i < Q_o$, and $Q_i - Q_o$ is substituted when $h \geq h_{\max} \wedge Q_i \geq Q_o$. This means that the only equations that are given to the DASSL solver are $Ah' = Q_i - Q_o - Q_x$ and $Q_o = k\sqrt{\rho g h}$, where h and Q_o are the two continuous variables. Q_i is a constant parameter, and Q_x is substituted by either 0 or $Q_i - Q_o$, depending on the value of h and the value of Q_i relative to Q_o.

Currently, the modeller needs to identify substitution variables by means of the $\leftarrow$ symbol. In future, we want to let the χ simulator determine substitution variables by means of symbolic analysis and manipulation of the equations, so that the modeller no longer needs to do this.

6. CONCLUSIONS

A small number of flexible, orthogonal discrete-event and continuous-time χ language constructs allow different types of discontinuities to be specified in an elegant way. Discontinuities that occur in the discrete-event phase are easily handled by calculation of a new initial state at the end of the discrete-event phase. In the continuous-time phase, the DASSL differential algebraic equation solver used in the χ simulator handles D' discontinuities, but cannot handle D discontinuities. By means of techniques such as substitution, variables that cause the discontinuity can be removed.

ACKNOWLEDGEMENT

We thank Georgina Fábián for helpful comments, and for the idea of substitution of continuous variables.

REFERENCES

[BGR 97] D.A. van Beek, S.H.F. Gordijn, J.E. Rooda. "Integrating continuous-time and discrete-event concepts in modelling and simulation of manufacturing machines." *Simulation Practice and Theory*, **5**, 653–669.

[BP 94] P.I. Barton, C.C. Pantelides. "Modeling of combined discrete/continuous processes." *AIChE*, **40**(6), 966–979.

[BRG 96] D.A. van Beek, J.E. Rooda, S.H.F. Gordijn. "Hybrid modelling in discrete-event control system design." In: *CESA'96 IMACS Multiconference, Symposium on Discrete Events and Manufacturing Systems*, pp. 596–601.

[CEOT 93] F.E. Cellier, H. Elmqvist, M. Otter, J.H. Taylor. "Guidelines for modeling and simulation of hybrid systems." In: *IFAC 12th Triennial Wordl Congress*, pp. 1219–1225.

[Elm 94] H. Elmqvist. *Dymola – Dynamic Modeling Language – User's Manual*. Dynasim AB, Lund, Sweden.

[MFRN 95] J.M. van de Mortel-Fronczak, J.E. Rooda, N.J.M. van den Nieuwelaar. "Specification of a flexible manufacturing system using concurrent programming." *Concurrent Engineering: Research and Applications*, **3**(3), 187–194.

[MG 76] E.E.L. Mitchell, J.S. Gauthier. "Advanced continuous simulation language (ACSL)." *Simulation*, **26**(3), 72–78.

[Pet 83] L.R. Petzold. "A description of DASSL: A differential/algebraic system solver." *Scientific Computing*, pp. 65–68.