The HI-Maude Tool

Muhammad Fadlisyah and Peter Csaba Ölveczky

University of Oslo

Abstract. In complex hybrid systems, different components may influence each others' *continuous* behaviors. HI-Maude is a rewriting-logic-based tool that supports an object-oriented modeling methodology in which it is sufficient to specify the continuous dynamics of *single* (physical component and physical interaction) objects in such interacting hybrid systems. HI-Maude supports simulation and model checking for a number of numerical approximations of the continuous behaviors, based on adaptations of the Euler and the Runga-Kutta methods.

1 Introduction

Many nontrivial hybrid systems consist of several components that may influence each others' *continuous* behaviors. The continuous behavior of such systems is typically very hard to define. Consider a system consisting of a cup of hot coffee in a room. The coffee will continuously become cooler *and* the room temperature will increase due to heat transfer from the coffee to the room. Although the continuous behaviors of the single components and the heat flow between them are well known and can be easily defined (see Fig. 1), it is very challenging to define the continuous behavior of the entire system "explicitly" in one shot, which is what current formal models of hybrid systems require.

In addition to making it practically impossible to define the continuous behaviors of nontrivial systems, such as the one discussed in Section 4, existing formalisms do not support an object-oriented specification methodology of continuously interacting hybrid systems, since the continuous behavior must be redefined for each new configuration of objects, e.g., if we have *three* cups of coffee in the room. Existing methodologies also do not support central object-oriented features such as dynamic creation and deletion of objects.

HI-Maude is a rewriting-logic-based formal tool for hybrid systems that supports an object-oriented modeling methodology in which both the physical components *and* their physical *interactions* are modeled explicitly. For example, heat flows from the coffee and the cup to the room through heat *convection*, and heat flows between the coffee and the cup through heat *conduction*. In HI-Maude, one can define the continuous dynamics of *single* physical component objects and single interaction objects. HI-Maude then computes the continuous dynamics of the entire system. This enables object-oriented modeling, since both the discrete and the continuous dynamics are defined at the class level, and since the dynamic creation/deletion of physical components is supported. For example, to add another cup of coffee, one could just add (possibly dynamically) a new coffee object, a new cup object, and three new interaction objects (for the convection between the new cup and the room and between the new coffee and the room, and for the conduction between the new coffee and the new cup) to the state.

To analyze hybrid systems—and HI-Maude targets complex systems whose continuous dynamics may be defined by differential equations that are not analytically solvable—HI-Maude uses an adaptation of different numerical methods (the Euler method and Runge-Kutta methods of different order) to give approximate solutions to coupled ordinary differential equations. These approximations are then used in HI-Maude simulation, reachability analysis, and linear temporal logic model checking. Since the numerical methods only approximate the real continuous behaviors, HI-Maude analyses are in general not sound and complete.

The HI-Maude tool, together with examples and documentation, is available at `http://folk.uio.no/mohamf/HI-Maude`. The paper [2] describes the adaptation of the numerical methods to the effort/flow modeling approach and compares the accuracy and execution times of the different methods in HI-Maude; the paper [3] presents HI-Maude and its semantic foundations in more detail; and [4] describes the case study summarized in Section 4.

2 Formal Modeling in HI-Maude

The HI-Maude modeling methodology is based on the the *effort/flow* method [5], in which a physical system is modeled as a network of *physical entities* and *physical interactions* between the entities. This approach is applicable to different systems. In mechanical translation systems, the pair of effort and flow variables are force and velocity; in mechanical rotation systems, torque and angular velocity; in electrical systems, voltage and current; in fluidic systems, pressure and volume flow rate; and in thermal systems, temperature and heat flow rate.

A *physical entity* is described by an *effort* value, a set of *attribute* values, and the entity's *continuous dynamics* (see Fig. 1, left). The effort variable represents a physical quantity, such as temperature, that evolves continuously, where the time derivative $\dot{e}$ of the effort e is a function of both the entity's attribute values *and* the flows of connected interactions (i.e., $\dot{e} = f(\mathit{atts}, \sum \mathit{flows})$).

An *interaction* between two physical entities is described by a *flow*, a set of *attribute* values, and a *continuous dynamics*. The flow value's evolution over time is specified as function of the interaction's attributes and the efforts of the connected entities (i.e., $\mathit{flow} = g(\mathit{atts}, \mathit{effort}_1, \mathit{effort}_2)$). A *one-sided interaction* represents an interaction between a physical entity and its environment.

Figure 1 illustrates the effort/flow methodology on our coffee example, where the flow variable ($\dot{Q}$) denotes the heat flow rate. The effort variables T_R, T_C, and T_K of the room, the coffee, and the cup are their temperatures. The attributes denote parameters such as the mass and surface area of the coffee. An immersion heater that can be dynamically added to or removed from the system adds constant heat to the coffee. The system also exhibits discrete behaviors; e.g., the heater is turned off and on to keep the coffee temperature between 70° and 80°.

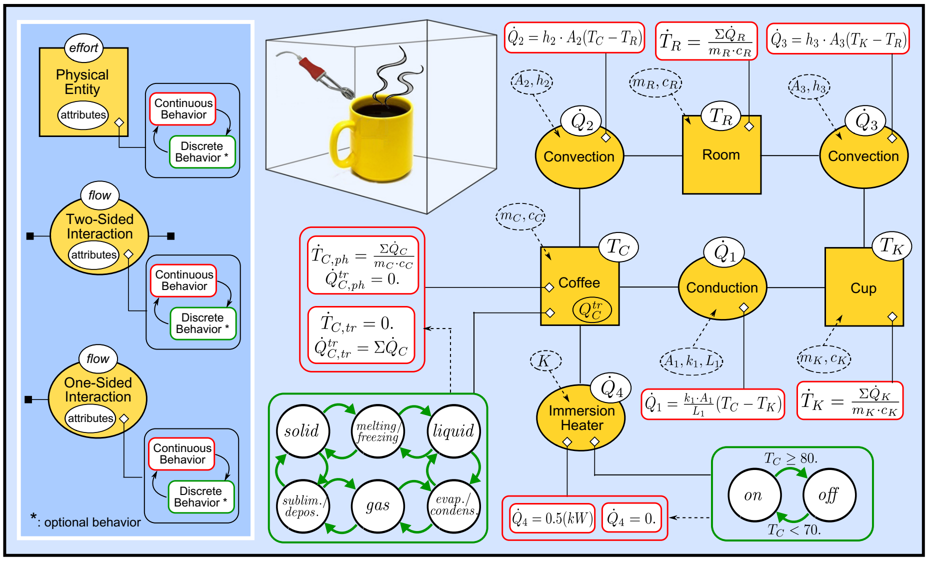

Fig. 1. Physical system components and their interaction in a simple thermal system

Since HI-Maude extends Maude [1], a *membership equational logic* theory (Σ, E), with Σ a signature[1] and E a set of *conditional equations*, specifies the system's state space as an algebraic data type. Instantaneous transitions are specified by conditional *rewrite rules* `crl` t `=>` t' `if` *cond*, where t and t' are two Σ-terms. A declaration `class` C `|` att_1 `:` s_1`, ... ,` att_n `:` s_n declares a *class* C with attributes att_1 to att_n of sorts s_1 to s_n. An *object* of class C is represented as a term $< O : C \mid att_1 : val_1, ..., att_n : val_n >$, where O, of sort `Oid`, is the object's *identifier*, and where val_1 to val_n are the values of the attributes att_1 to att_n. The state is a term of sort `Configuration` denoting a *multiset* of objects and messages. A *subclass* inherits the attributes and rules of its superclasses.

In HI-Maude, physical entities and (one-sided and two-sided) interactions are defined as object instances of user-defined subclasses of the built-in classes

```
class PhysicalEntity | effort : Float .
class TwoSidedInteraction | flow : Float, entity1 : Oid, entity2 : Oid .
class OneSidedInteraction | flow : Float, entity : Oid .
```

The entity attributes denote the physical entities involved in the interaction. The user must define the time derivative `effortDyn(`$object, \sum\dot{Q}$`)` of the effort variable of *object* for each physical entity; the sum $\sum\dot{Q}$ of the flows to/from the entity is provided by the tool. For example, if $\dot{e} = f(atts, \sum flows)$, we define `effortDyn(< O : C |` *atts* `>, X) =` f`(`*atts*`,X)`. We specify `flowDyn`$(object, e_1, e_2)$ and `flowDyn`$(object, e_1)$ to define the continuous dynamics of two-sided (resp. one-sided) *interaction objects*; the effort values e_1 and e_2 are given by HI-Maude.

[1] i.e., Σ is a set of declarations of *sorts*, *subsorts*, and *function symbols*.

Discrete transitions are modeled as rewrite rules, and `timeCanAdvance`(s) must be `false` in states s in which a rule must be applied.

To model the system in Fig. 1, we first define a class `ThermalEntity` (with attributes denoting the heat capacity and the mass of the entity), whose objects model thermal entities, such as the cup, the coffee, and the room:

```
class ThermalEntity | heatCap : Float, mass : Float .
subclass ThermalEntity < PhysicalEntity .
```

The `effort` attribute of the superclass denotes the temperature; its continuous dynamics is defined in the same way for each entity (e.g., $\dot{T}_R = \frac{\sum \dot{Q}_R}{m_R \cdot c_R}$):[2]

```
eq effortDyn(< O : ThermalEntity | mass : M, heatCap : C >, X) = X / (M * C) .
```

The heat flow by *convection* through the surface of the coffee or cup is given by the temperatures of the entities, the *area* of the surface (A), and the convection coefficient h. The definition of its dynamics is straight-forward from Fig. 1:

```
class Convection | area : Float, convCoeff : Float .
subclass Convection < TwoSidedInteraction .
eq flowDyn(< O : Convection | convCoeff : CC, area : A >, E1, E2) = CC * A * (E1 - E2).
```

The heater is a one-sided interaction (whose flow is 0 when the `status` is `off` and is 500 when the `status` is `on`; it also monitors the temperature of the coffee):

```
class Heater | status : OnOff, temp : Float.   subclass Heater < OneSidedInteraction.
eq flowDyn(< O : Heater | status : S >, E) = if S == on then 500.0 else 0.0 fi.
```

A rewrite rule turns off the heater when the temperature of the coffee has reached 80 degrees. `timeCanAdvance` forces the timely application of this rule:

```
crl < H : Heater | status : on, temp : T >  =>  < H : Heater | status : off >  if T >= 80.0.
ceq timeCanAdvance(< H : Heater | status : on, temp : T >) = false  if T >= 80.0 .
```

3 Formal Analysis in HI-Maude

HI-Maude provides a range of formal analyses, including: (i) simulating *one* behavior from a given initial state; (ii) checking whether a state matching a state pattern is reachable from the initial state (possibly within a given time interval); (iii) finding the shortest/longest time needed to reach an (un)desired state; and (iv) checking whether all possible behaviors from the initial state (possibly up to some duration) satisfy a linear temporal logic property. In all such analysis, HI-Maude advances time in small time increments and approximates the values of the continuous variables at each "visited" point in time. We have adapted the following numerical methods to our effort/flow framework: the *Euler*, the *Runge-Kutta 2nd order* (RK2), and the *Runge-Kutta 4th order* (RK4) methods.

[2] We follow the Maude convention that variables are written with (only) capital letters, and do not show the variable declarations.

In any analysis command, the user selects the numerical approximation technique and the time increment used in the approximation. For example, HI-Maude's hybrid *rewrite* command is used to simulate one behavior of the system from a given initial state *initState* up to duration *timeLimit*:

```
(hrew initState in time ∼ timeLimit using numMethod stepsize stepSize .)
```

$\sim$ is either '`<=`' or '`<`'; *numMethod* $\in \{\texttt{euler}, \texttt{rk2}, \texttt{rk4}\}$; and *stepSize* is the time increment used in the approximations.

Using HI-Maude. The following module defines an initial state `c1` for the coffee example, consisting of one room, one heater, one coffee, and one cup object (each with temperature 20°), and the interaction objects in Fig. 1:

```
(homod COFFEE-SYSTEM is  including HYBRID-LIB .  including TIMED-MODEL-CHECKER .
...
eq c1 =
{< coffee : WaterEntity | effort : 20.0, mass : m_C, heatCap : c_C, phase : liquid, ... >
 < cup : ThermalEntity  | effort : 20.0, mass : m_K, heatCap : c_K, ... >
 < room : ThermalEntity | effort : 20.0, mass : m_R, heatCap : c_R, ... >
 < condCK : Conduction  | flow : 0.0, entity1 : coffee, entity2 : cup, thermCond : k_1,
                          thickness : L_1, area : A_1, ... >
 < convCR : Convection  | flow : 0.0, entity1 : coffee, entity2 : room, area : A_2,
                          convCoeff : h_2 ... >
 < convKR : Convection  | flow : 0.0, entity1 : cup, entity2 : room, area : A_3, ... >
 < immerHeater : Heater | flow : 0.0, entity : coffee, state : off, ... >} .
endhom)
```

HI-Maude simulation shows that after 5 minutes, the temperatures of the coffee, cup, and room are 79.47°, 74.43°, and 20.17°, respectively:

```
Maude> (hrew c1 in time <= 300 using euler stepsize 1.0 .)

Result ClockedSystem :
  {< coffee : WaterEntity | effort : 7.9470872797477682e+1, phase : liquid, ... >
   < cup : ThermalEntity  | effort : 7.4430660426346876e+1, ... >
   < room : ThermalEntity | effort : 2.0171582578947586e+1, ... > ... } in time 300
```

HI-Maude's *hybrid find earliest* command can then be used to find out how quickly the coffee temperature can reach 80°:

```
Maude> (hfind earliest
          c1 =>* {C:Configuration  < coffee : WaterEntity | effort : T:Float >}
           such that (T:Float >= 80.0) using euler stepsize 1.0 .)

Result: {< coffee : WaterEntity | effort : 8.00470...e+1, ... > ... } in time 277
```

Finally, we define a state proposition `temp-ok` to hold if the temperature of the coffee temperature is between 69.95° and 80.05°:

```
eq {REST  < coffee : WaterEntity | effort : T >} |= temp-ok = T >= 69.95 and T <= 80.05.
```

We can then model check (for all behaviors up to 30 minutes) that once an ok coffee temperature has been reached, it will remain in this interval:

```
Maude> (hmc c1 |=t [](temp-ok -> [] temp-ok) in time <= 1800 using euler stepsize 1.0.)

Result Bool :  true
```

4 Case Study: The Sauna World Championships

The winner of the Sauna World Championships is the contestant who can stay the longest in a 110°C sauna where half a liter of water is poured onto the sauna rocks every thirty seconds. The 2010 event ended in a tragedy when the two finalists collapsed with severe burn injuries after about six minutes; one of them died the next day. The cause of this tragedy is still under investigation.

Instead of experimenting with humans in different saunas to investigate the cause of this unsolved accident, one might use computers to analyze possible causes of the accident. However, that is a tricky task for reasons that include:

- The continuously changing temperature of the skin and of the "core" of the human body are different. Since problems may be caused by severe hyperthermia (body temperature above 40.6°C) and second degree burn (skin temperature above 55°C), we must take both values into account.
- The system also has instantaneous transitions (e.g., release of half a liter of water in the sauna) and nondeterministic behaviors (others leave the sauna).

We have defined a HI-Maude model of the human thermoregulatory system according to accepted physiological facts and models [4], where the body core and the skin are two main components. Heat flows between them through blood flows where the diameter of the blood vessels changes continuously. The main forms of heat exchange between skin and environment are by conduction/convection, radiation, and evaporation of sweat; between core and environment heat flows mainly through respiration. For the sauna, we model the heating rocks, their specific heat capacity, the pouring of water on the rocks, the heater, etc.

Our HI-Maude analyses (see [4]) show that even the average person should endure 12 minutes in the sauna before the onset of major injuries. Our results seem consistent with what we know about how long both professionals and amateurs can endure in the sauna. HI-Maude analysis showed that any of the following scenarios could explain the still unsolved tragedy that could cause major injuries to a five-time world champion in around 6 minutes:

- The initial temperature of the heating rocks is 250°C.
- The sauna temperature of 210°C.
- The initial humidity is very high (39 liters of water vapor instead of 10 liters).

References

1. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude. LNCS, vol. 4350. Springer, Heidelberg (2007)
2. Fadlisyah, M., Ölveczky, P.C., Ábrahám, E.: Formal modeling and analysis of hybrid systems in rewriting logic using higher order numerical methods and discrete-event detection. In: Proc. CSSE 2011. IEEE (2011)
3. Fadlisyah, M., Ölveczky, P.C., Ábrahám, E.: Object-oriented formal modeling and analysis of interacting hybrid systems in HI-Maude. In: Barthe, G., Pardo, A., Schneider, G. (eds.) SEFM 2011. LNCS, vol. 7041, pp. 415–430. Springer, Heidelberg (2011)
4. Fadlisyah, M., Ölveczky, P.C., Ábrahám, E.: Formal modeling and analysis of human body exposure to extreme heat in HI-Maude. In: Durán, F. (ed.) WRLA 2012. LNCS, vol. 7571, pp. 139–161. Springer, Heidelberg (2012)
5. Wellstead, P.E.: Introduction to physical system modelling. Academic Press (1979)