

Formal modeling and analysis of interacting hybrid systems in HI-Maude: What happened at the 2010 Sauna World Championships?



Muhammad Fadlisyah^a, Peter Csaba Ölveczky^{a,*}, Erika Ábrahám^b

^a University of Oslo, Norway

^b RWTH Aachen University, Germany

HIGHLIGHTS

- Provides a formal rewriting logic model of the human thermoregulatory system.
- Proposes a formal model for discomfort thresholds and voluntary behaviors.
- Defines a formal model of the sauna used in the Sauna World Championships.
- Analyzes possible causes of the accident at the 2010 Sauna World Championships.

ARTICLE INFO

Article history:

Received 22 January 2013

Received in revised form 31 March 2014

Accepted 3 June 2014

Available online 10 July 2014

Keywords:

Simulation

Rewriting logic

Hybrid systems

Object-oriented specification

Human thermoregulatory system

ABSTRACT

In this paper we use HI-Maude to model and analyze the human thermoregulatory system and the effect of extreme heat exposure to the human body. The case study is motivated by the 2010 Sauna World Championships, which ended in a tragedy when the last two finalists were severely burnt in surprisingly short time (one of them died the next day). HI-Maude is a rewriting-logic-based formal modeling language and analysis tool for complex hybrid systems whose components influence each others' continuous dynamics. One distinguishing feature of HI-Maude is that the user only needs to describe the continuous dynamics of *single* components and interactions, instead of having to explicitly define the continuous dynamics of the entire system. HI-Maude analyses are based on numerical approximations of the system's continuous behaviors. We use HI-Maude to analyze how long the human body can survive when experiencing extreme conditions such as those encountered in the Sauna World Championships.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

August 7, 2010. The final of the 2010 Sauna World Championships in Heinola, Finland, pits the national hero, five-time world champion Timo “The Great” Kaukonen—sponsored by a sauna company and practicing in a mobile 140 °C sauna three times a day—against the Russian Vladimir Ladyzhensky. Although the winning time is typically between 12 and 18 minutes, on this beautiful summer day both finalists collapse after about six minutes. Ladyzhensky dies the next day; Kaukonen wakes up from a medically induced coma six weeks after the event, his respiratory system scorched, 70% of his skin burnt, and his kidneys failed as well. What happened? The cause of this tragedy is still unknown.

* Corresponding author.

July 31, 2001. The Minnesota Vikings professional football team holds its preseason training camp in Mankato, Minn. Suddenly, one of the team's stars, Pro Bowler Korey Stringer, collapses of heat stroke and dies 15 hours later [1]. In addition to understanding when the danger of heat stroke sets in, an interesting question posed by this tragedy is why Stringer did not leave the field before the tragedy; that is, will a human experience significant pain or discomfort before succumbing to heat stroke?

There is a great interest in studying how long human beings can endure extreme heat or cold stress, or, more generally, how humans can stay comfortable and safe when firefighting, flying large passenger planes, traveling in space, floating in cold water, parachuting in the “stratosphere,” or just practicing sports.¹ The best way to study how long humans can survive in extreme conditions is probably to experiment on human subjects. A large number of medical experiments on “prisoners of war” during World War II studied how long humans could survive in ice cold water, in the cold Eastern Front, and in other extreme conditions (see, e.g., [3]). However, it is not considered appropriate to experiment on humans in order to study the body's reaction to extreme stress, for example to understand what happened in Heinola in 2010. Although animals can sometimes replace humans in such experiments, they often function quite differently than humans, and experimentation on animals must take the growing animal rights sentiment into account.

Computer-based simulation and analysis can be a cheap and useful way to study the human body's reaction to stimuli. The human thermoregulatory system tries to keep the person at a comfortable temperature even in difficult environments. Understanding this system is crucial to understand how our body will respond to hostile and extreme environments. Inspired by the above two cases, we are interested in using automated analysis to study three questions: (i) How long can a person endure in different extreme situations, such as in the sauna used in the Sauna World Championships, before risking significant injuries? (ii) What could have happened at the 2010 Sauna World Championships, or, more precisely, what conditions of the sauna could explain the events in 2010? (iii) Will a person participating in events such as a sauna competition or an NFL training camp *feel* significant thermal discomfort before the onset of significant injury/death?

Defining useful computer models of the human thermoregulatory system is a challenging task. We need to model as closely as possible the complex continuous dynamics of the various parts (skin, core, blood, etc.) and, in particular, the complex continuous physical interaction between these parts, and between the body and its environment. We may also want to model behavioral, and therefore nondeterministic, aspects of the thermoregulatory system. Finally, we must be able to account for changing configurations, such as when a human jumps out of an oppressively hot sauna and into the cold snow; in this case, the thermal interactions change instantly, and the continuous dynamics of the entire system must be recomputed.

In this paper we use the rewriting-logic-based HI-Maude language and tool [4,5] to model the human thermoregulatory system according to established medical/physiological models and to analyze it under extreme conditions. Since our investigation is partially motivated by the 2010 Sauna World Championships, we also present a fairly detailed model of the thermodynamics of a sauna and use the HI-Maude tool to address the three questions above.

HI-Maude is a recent extension of Real-Time Maude [6] to support the object-oriented formal modeling, simulation, and further formal analysis of *hybrid systems* with combined discrete and continuous behaviors. Since we target large and complex hybrid systems, these systems tend to have multiple physical entities that interact and influence each other's continuous behavior. For a thermal system example, consider the proverbial cup of hot coffee in a room which interacts with the room through different kinds of heat transfer, leading to a decrease of the coffee's temperature and to a slight increase of the room's temperature. One distinguishing feature of HI-Maude is the *modularity* and *compositionality* of the specification of the system's continuous dynamics. Non-compositional specification of the *whole* system is very hard, as it involves combining the ordinary differential equations (ODEs) that specify the dynamics of its components; it also requires redefining the system's continuous dynamics for each new configuration of interacting physical components. To achieve the desired modularity and compositionality, HI-Maude offers an object-oriented modeling methodology [7] that allows us to specify the continuous dynamics of *single physical entities* (such as the cup of coffee and the room) and of *single physical interactions* (such as thermal conduction and convection). Not only does this decomposition make it easier to specify the continuous dynamics of a system of interacting physical components, but it also means that the specification does not need to be redefined for each new configuration of physical entities. If we want to add a cup of coffee to the room, we just add a new coffee object and appropriate physical interaction objects to the state.

To analyze the system, whose continuous dynamics is typically defined by ordinary differential equations that are not analytically solvable, HI-Maude uses adaptations of different numerical methods (the Euler method and Runge–Kutta methods of different order) to give fairly precise approximate solutions to coupled ordinary differential equations. These approximations are then used in HI-Maude simulation, reachability analysis, and linear temporal logic model checking.

We (three computer scientists) follow as much as possible established medical facts and models when defining our own models. We had to combine a number of different sources and models to define our model, since a single integrated model of the human body that can be used to analyze the sauna accident does not seem to exist. The reference [8] gives an overview of some approaches to model the human thermoregulatory system. We choose the two-node Gagge model [9] as the basic model of the human body, since much scientific and engineering research on human thermoregulation is based on this model. For the formulas used to model the physiological aspects of the human thermoregulatory system, our main

¹ Heat stroke is the third leading cause of death among athletes in the U.S. [2].

sources are [10,11]. We use [12] as the main source for some physics-related equations for modeling some aspects of interaction of human body with the environment, and use [11,13] as main sources for modeling the behavioral aspect of the human thermoregulatory system. Our main sources on how to model experimental subjects in different physiological conditions and degrees of preparedness for the Sauna World Championships are [14–16,10].

We have tried our best to model the sauna as closely as possible to the one used in the Sauna World Championships. Since we have not found any official descriptions about the sauna room in the event, we have had to rely on information gathered from related stories, photographs, and videos available on the web, and from the related technical specification of the equipments (e.g., the heater) and physical properties of the material used (e.g., the heat capacity of the kind of stones used for the heating). We used some information from [17] for some physical properties of the environment.

This paper is a significant extension of our paper [18] that includes many more details about our model and its analyses. In particular, modeling and analyzing thermal discomfort (issue (iii) above) were not considered at all in [18]. Neither did we in [18] include any details about thermal interactions such as shivering, evaporation, respiration, or convection. However, even in a much longer paper, we can still only provide a sampler of our model. The entire executable formal HI-Maude model, the analysis commands, a long report, and the HI-Maude tool itself, are available at <http://folk.uio.no/mohamf/HI-Maude>.

Section 2 introduces the effort/flow-based modeling methodology upon which the HI-Maude modeling methodology and analysis tool described in Section 3 is based. Section 4 gives an overview of the human thermoregulatory system. Section 5 presents some parts of our model of the human thermoregulatory system, and Section 6 presents our model of the Heinola sauna. Section 7 uses HI-Maude to analyze how long humans can stay safely in saunas and tries to understand what could have happened in 2010. Section 8 briefly discusses related work and Section 9 gives some concluding remarks.

2. Effort/flow modeling of interacting hybrid systems

HI-Maude is based on the modeling methodology introduced in [7], which adapts the *effort/flow* method [19] to model a physical system as a network of *physical entities* and *physical interactions* between the entities. This approach is applicable to different areas of physical 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; in thermal systems, temperature and heat flow rate.

As shown in Fig. 1 (top left), a *physical entity* is described by a real-valued *effort* value, a set of *attribute* values, and the entity's *continuous dynamics*. The effort variable represents a dynamic physical quantity, such as temperature, that evolves over time as given by the continuous dynamics in the form of an ordinary differential equation (ODE), where its time derivative is a function of both the entity's attribute values and the flows of connected interactions (i.e., the time derivative \dot{e} of the effort e can be described by an equation of the form $\dot{e} = f(\sum \text{flows}, \text{atts})$).

A *two-sided interaction* between two physical entities is described by a real-valued *flow*, a set of *attribute* values, and a *continuous dynamics*. The flow value describes the dynamic interaction between two entities, whose evolution over time is specified by the continuous dynamics as an equation with the flow variable on the left-hand side and an expression possibly referring to the interaction's attributes and the efforts of the connected entities on the right-hand side (i.e., $\text{flow} = g(\text{effort}_1, \text{effort}_2, \text{atts})$). A *one-sided interaction* represents an interaction of a physical entity with its environment. The system may also exhibit discrete dynamics, for representing, e.g., the changes of physical states of the system components, explicit control behaviors, communication, or other phenomena.

Fig. 1 illustrates our modeling methodology on a much simplified *thermal* system consisting of a woman working out at a gym. There are two *physical entities* of interest for thermal reasoning: the human body and the training room, each with the temperature (T_{bd} and T_{tr} , respectively) as its effort variable. The body produces heat, and the heat production increases as the exercise gets harder. The body releases heat to the room through the skin (e.g., by convection), and through sweating (heat is released from the body as the sweat evaporates). These heat transfers are represented as *two-sided interactions* where the flow variable (\dot{Q}) denotes the heat flow rate of the interactions. For example, in Fig. 1, heat flows by convection from the human body to the training room through the “Heat Exchange Through Skin” physical interaction, and this heat flow rate \dot{Q}_{skn} equals $h_{CNV} \cdot A_{skn} \cdot (T_{bd} - T_{tr})$, where h_{CNV} is the convection coefficient and A_{skn} is the area of the skin. If the training room is warmer than the body, then this heat flow from body to room is negative, and hence heat flows from room to body.

The *one-sided interaction* is used to model the heat production inside the human body through metabolism, and also to model the system which handles heating and cooling of the gym. For example, when the heating/cooling system is turned off, the one-sided interaction labeled “Heating, Ventilation, and Air Conditioning” provides no heat flow (given by \dot{Q}_{hvac}) from the heating system to the room, whereas the heat flow from this system to the room is $C_{HEAT} \cdot A_{hvac}$ when the heating/cooling system is in state *heating*.

The change in temperature of an entity like a human body (\dot{T}_{bd}) is then given by the sum of the flows into the body (which may well be a negative number) divided by the product of the mass m_{bd} and the heat capacity c_{bd} of the body.

Beside continuous dynamics, there are some physical phenomena which are suitably modeled as discrete dynamics, e.g., the changes of activity during the training (from running, to walking, to resting), activation and deactivation of sweating, turning on/off the heater/cooler, and so on.

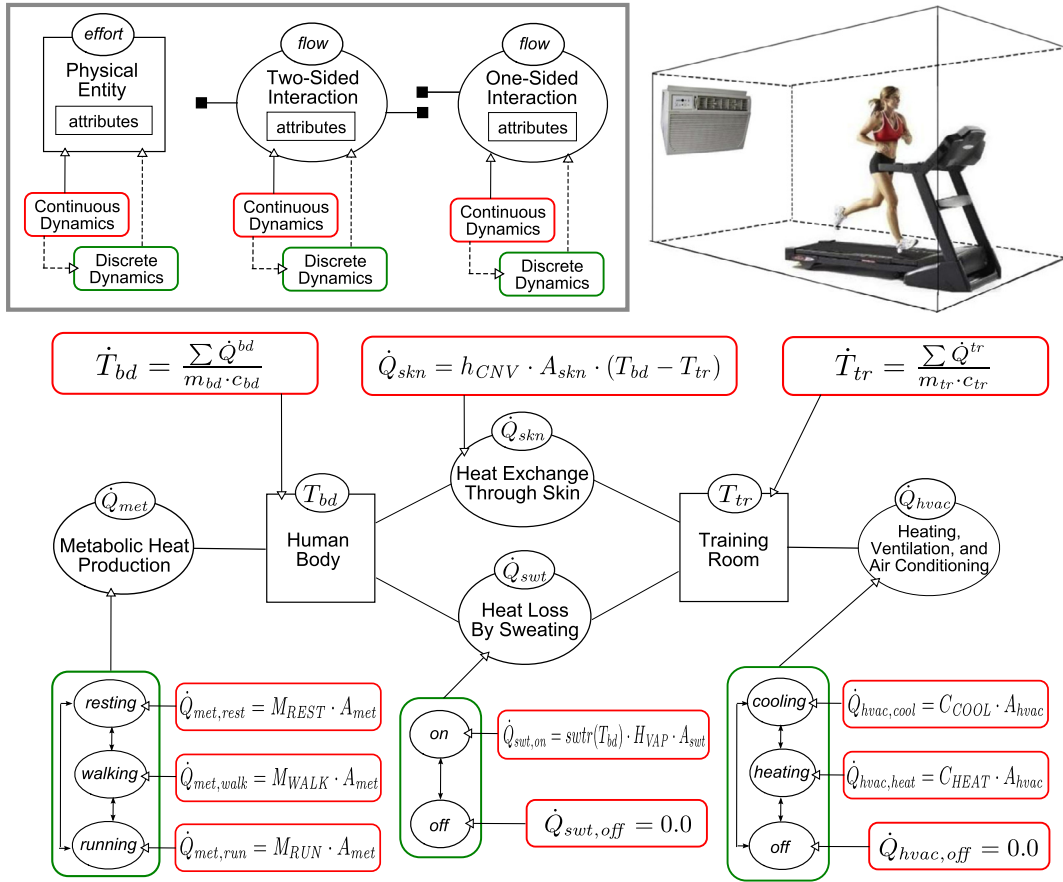


Fig. 1. Modeling the human thermoregulatory system using the effort/flow method.

3. The HI-Maude tool

HI-Maude [4,5] is a language and analysis tool that extends Maude [20] and its real-time extension Real-Time Maude [21] to support the object-oriented effort/flow-based modeling and approximation-based formal analysis of interacting hybrid systems. Since HI-Maude targets complex systems, and therefore does not restrict to *linear* ODEs for describing the continuous dynamics of a component/interaction, the continuous dynamics of a system is in general not analytically solvable. HI-Maude therefore uses numerical techniques to *approximate* the continuous behaviors by advancing time in small discrete time increments, and approximating the values of the continuous variables at each “visited” point in time.

This execution is based on adaptations of the *Euler* [7], the *Runge–Kutta 2nd order* (RK2), and the *Runge–Kutta 4th order* (RK4) numerical approximation methods [22] to the effort/flow framework. The papers [7,22] show the execution times and the relative errors for the different numerical methods on an example that does have an analytical solution, i.e., when we can compute the solved form of the differential equations. RK4 is the most sophisticated approximation algorithm of these, and Euler is the least sophisticated one. On the other hand, Euler is the computationally fastest, whereas RK4 is the slowest. The experiments in [7,22] indicate, for the one example, that RK2 is a good compromise between precision and efficiency. The “step size” (or “time increment”) between each visited point in time is the parameter of these approximation methods; the smaller the step size, the more accurate the numerical approximation, and the longer the computation time.

Once the dynamics of the *single* physical entity and interaction components has been defined, HI-Maude

1. automatically defines the continuous dynamics of the entire systems, and
2. provides (Real-Time Maude-based) simulation and model checking commands, where the desired built-in approximation algorithm and the desired time increments used by the approximations are additional parameters of the commands.

HI-Maude is implemented in Maude as an extension of the Real-Time Maude tool, and is available at <http://folk.uio.no/mohamf/HI-Maude/>.

3.1. Preliminaries: rewriting logic and Maude

Since HI-Maude specifications extend Maude specifications, we first recall specification in Maude.

A *membership equational logic* (MEL) [23] *signature* is a triple $\Sigma = (K, \Omega, S)$, with K a set of *kinds*, $\Omega = \{\Omega_{w,k}\}_{(w,k) \in K^* \times K}$ a many-kinded algebraic signature, and $S = \{S_k\}_{k \in K}$ a K -kinded family of disjoint sets of sorts. The kind of a sort s is denoted by $[s]$. A MEL algebra A contains a set A_k for each kind k , a function $A_f : A_{k_1} \times \dots \times A_{k_n} \rightarrow A_k$ for each operator $f \in \Omega_{k_1 \dots k_n, k}$, and a subset $A_s \subseteq A_k$ for each sort $s \in S_k$. $T_{\Sigma, k}$ and $T_{\Sigma}(X)_k$ denote, respectively, the set of ground Σ -terms with kind k and of Σ -terms with kind over the set X of kinded variables.

A MEL *theory* is a pair (Σ, E) , where Σ is a MEL signature, and E is a set of conditional equations of the form $(\forall X) t = t'$ **if** *cond* and conditional memberships of the form $(\forall X) t : s$ **if** *cond*, for $t, t' \in T_{\Sigma}(X)_k, s \in S_k$, the latter stating that t is a term of sort s , provided the condition holds; the condition *cond* is a conjunction of individual equations $t_i = u_i$ and individual memberships $w_j : s_j$. In Maude, a conjunct in such a condition may also consist of just a single term t'' of kind $[B_{0 \circ 0 1}]$, in which case it abbreviates the equation $t'' = \text{true}$. Furthermore, in Maude an individual equation in the condition *cond* may also be a *matching equation* $t_l := u_l$, which is mathematically interpreted as an ordinary equation. However, operationally, the new variables occurring in the term t_l become instantiated by matching the term t_l against the canonical form of the instance of u_l (see [20] for further explanations). Order-sorted notation $s_1 < s_2$ can be used to abbreviate the conditional membership $(\forall x : [s_1]) x : s_2$ **if** $x : s_1$. Similarly, an operator declaration $f : s_1 \times \dots \times s_n \rightarrow s$ corresponds to declaring f at the kind level and giving the membership axiom $(\forall x_1 : k_1, \dots, x_n : k_n) f(x_1, \dots, x_n) : s$ **if** $\bigwedge_{1 \leq i \leq n} x_i : s_i$, where $[s_i] = k_i$.

A Maude module specifies a *rewrite theory* [24,25] of the form $(\Sigma, E \cup A, R)$, where $(\Sigma, E \cup A)$ is a membership equational logic theory with A a set of equational axioms such as associativity, commutativity, and identity, so that equational deduction is performed *modulo* the axioms A . The theory $(\Sigma, E \cup A)$ specifies the system's state space as an algebraic data type. R is a collection of *labeled conditional rewrite rules* specifying the system's local transitions, each of which has the form²

$$[l] : t \longrightarrow t' \text{ if } \bigwedge_{i=1}^m u_i = v_i \wedge \bigwedge_{j=1}^n w_j : s_j,$$

where l is a *label*. Intuitively, such a rule specifies a *one-step transition* from a substitution instance of t in a given context to the corresponding substitution instance of t' in the same context, *provided* the condition holds; that is, the substitution instances of the equalities $u_i = v_i$ and memberships $w_j : s_j$ follow from $E \cup A$. The rules are implicitly universally quantified by the variables appearing in the Σ -terms t, t', u_i, v_i , and w_j . The rules are applied *modulo* the equations $E \cup A$.³

We briefly summarize the syntax of Maude. Operators are introduced with the `op` keyword: `op f : s1 ... sn -> s`. They can have user-definable syntax, with underbars '`_`' marking the argument positions, and are declared with the sorts of their arguments and the sort of their result. Some operators can have equational *attributes*, such as `assoc`, `comm`, and `id`, stating, for example, that the operator is *associative* and *commutative* and has a certain *identity* element. Such attributes are then used by the Maude engine to match terms *modulo* the declared axioms. An operator can also be declared to be a *constructor* (`ctor`) that defines the carrier of a sort. Unconditional and conditional equations, memberships, and rewrite rules are introduced with the keywords `eq`, `ceq`, `mb`, `cmb`, `rl`, and `crl`, respectively. The mathematical variables in such statements are either explicitly declared with the keywords `var` and `vars`, or can be introduced on the fly in a statement without being declared previously, in which case they have the form `var : sort`.

In object-oriented Maude modules one can declare *classes* and *subclasses*. A class declaration

```
class C | att1 : s1, ... , attn : sn
```

declares an object class C with attributes att_1 to att_n of sorts s_1 to s_n . An *object* of class C in a given state is represented as a term

```
< O : C | att1 : val1, ... , attn : valn >
```

of the built-in sort `Object`, where O is the object's name or identifier, and where val_1 to val_n are the current values of the attributes att_1 to att_n and have sorts s_1 to s_n . Objects can interact with each other in a variety of ways, including the sending of messages. A message is a term of the built-in sort `Msg`, where the declaration

```
msg m : p1 ... pn -> Msg
```

² In general, the condition of such rules may not only contain equations $u_i = v_i$ and memberships $w_j : s_j$, but also rewrites $q_k \longrightarrow q'_k$; however, the specification in this paper does not use this extra generality. Furthermore, the individual equations, memberships, and rewrites in the condition may be intermixed.

³ Operationally, a term is reduced to its E -normal form modulo A before any rewrite rule is applied in Maude. Under the coherence assumption [26] this is a complete strategy to achieve the effect of rewriting in $E \cup A$ -equivalence classes.

defines the syntax of the message (m) and the sorts ($p_1 \dots p_n$) of its parameters. In a concurrent object-oriented system, the state, which is usually called a *configuration*, is a term of the built-in sort `Configuration`. It has the structure of a *multiset* made up of objects and messages. Multiset union for configurations is denoted by a juxtaposition operator (empty syntax) that is declared associative and commutative and having the `none` multiset as its identity element, so that rewriting is *multiset rewriting* supported directly in Maude. The dynamic behavior of object systems is axiomatized by specifying each of its concurrent transition patterns by a rewrite rule. For example, the configuration fragment on the left-hand side of the rule

```
r1 [1] :  m(O,w)
         < O : C | a1 : 0, a2 : O', a3 : z >
         =>
         < O : C | a1 : w, a2 : O', a3 : z >
         m'(O')
```

contains a message m , with parameters O and w , and an object O of class C . The message $m(O, w)$ does not occur in the right-hand side of this rule, and can be considered to have been *removed* from the state by the rule. Likewise, the message $m'(O')$ only occurs in the configuration on the right-hand side of the rule, and is thus *generated* by the rule. The above rule, therefore, defines a parametrized family of transitions (one for each substitution instance) in which a message $m(O, w)$ is read and consumed by an object O of class C , with the effect of altering the attribute $a1$ of the object and of sending a new message $m'(O')$. By convention, attributes, such as $a3$ in our example, whose values do not change and do not affect the next state of other attributes need not be mentioned in a rule or an equation. Attributes like $a2$ whose values influence the next state of other attributes or the values in messages, but are themselves unchanged, may be omitted from right-hand sides of rules/equations. Of course, rules do not need to be triggered by messages; indeed, most of the rules in our model do not contain messages, but only a number of objects in the left-hand and right-hand sides. If there is more than one object in the rule's left-hand side, then the objects participating in an application of such a rule can be seen to “synchronize” and change “in lockstep.”

A *subclass* inherits all the attributes, equations, and rules of its superclasses,⁴ and multiple inheritance is supported [27].

3.2. Modeling in HI-Maude

HI-Maude extends Maude and Real-Time Maude to support the effort/flow-based object-oriented modeling of interacting hybrid systems by providing a set of built-in classes for defining physical entities and (one-sided and two-sided) physical interactions. Concrete physical entities and interactions must then be defined as object instances of user-defined subclasses of these built-in classes.

Before introducing these built-in classes, we would like to mention that although Maude provides a built-in data type for the *unbounded* rational numbers, and HI-Maude originally used these rationals for the effort and flow values, it quickly became apparent that it is inconvenient to use the rationals, because: (i) it is hard to read large rational numbers; and (ii) the *size* of the rational numbers gets very large (both the numerator and the denominator are large numbers), which slows down the execution. HI-Maude therefore uses the Maude's built-in IEEE-754 floating-point numbers to represent the effort and flow values.

Subclasses of the built-in class `PhysicalEntity` should be used to define physical entities:

```
class PhysicalEntity | effort : Float .
```

Sometimes a physical entity needs additional continuous variables whose dynamics are time-derivative functions. The tool therefore provides the classes `PhysicalEntityAck`, where k denotes the number of additional continuous variables⁵:

```
class PhysicalEntityAck | contvar1 : Float , ... , contvark : Float .
subclass PhysicalEntityAck1 ... PhysicalEntityAckn < PhysicalEntity .
```

The `contvar` attributes denote the additional continuous variables. In our case study, the main effort attribute of the core of the human body is its temperature; however, we also need to analyze the amount of water in the body to be able to analyze dehydration properties. Since the sweating rate is a time-derivative function of the body water content, the body core must be a subclass of `PhysicalEntityAck1`, where the additional continuous attribute `contvar1` denotes the amount of water in the body (see Section 5).

Objects of the classes `TwoSidedInteraction` and `OneSidedInteraction` should be used to model two-sided and one-sided physical interactions, respectively:

⁴ The attributes, equations, and rules of a class cannot be redefined by its subclasses, but subclasses may introduce additional attributes, equations, and rules.

⁵ The tool currently provides the entity class with two additional continuous variables.

```

class PhysicalInteraction | flow : Float, contdyntype : ContDynType .
class TwoSidedInteraction | entity1 : Oid, entity2 : Oid .
class OneSidedInteraction | entity : Oid .
subclass TwoSidedInteraction OneSidedInteraction < PhysicalInteraction .

```

The `contdyntype` attribute denotes the type of continuous dynamics specified for the interaction. The `entity1` and `entity2` attributes denote the two physical entities involved in the two-sided interaction. The `entity` attribute of the class `OneSidedInteraction` denotes the entity interacting with the environment.

HI-Maude requires the user to define the continuous dynamics by defining the function `effortDyn` for each physical entity:

```
op effortDyn : Object Float -> Float .
```

The first argument of `effortDyn` is the entity object itself, which contains all attributes (including the effort variable); the second argument is the sum of the values of the flows to/from the entity, and is computed by the tool. $\text{effortDyn}(\text{object}, \sum Q)$ therefore defines the time derivative of the effort variable e of the object. That is, if $\dot{e} = f(\sum \text{flows}, \text{atts})$, we define $\text{effortDyn}(\langle O : C \mid \text{atts} \rangle, X) = f(X, \text{atts})$.

For the physical entities with additional continuous variable(s), the functions `contvar1Dyn` define the continuous dynamics of the additional continuous variables:

```
ops contvar1Dyn ... contvarnDyn : Object Configuration -> Float .
```

The first argument of the function is the entity object itself; the second argument is the entire multiset of objects in the system.

The function `flowDyn` defines the continuous dynamics of the physical interactions. To define the continuous dynamics of, respectively, two-sided interactions and one-sided interactions, the following formats are used:

```
op flowDyn : Object Float Float -> Float .
op flowDyn : Object Float -> Float .
```

The first argument of the function is the interaction object itself. The second (and third) arguments are the effort variable values of the interacting physical entiti(es).

Sometimes attribute values from objects that are not directly related to the interaction must be used to define the continuous dynamics of an interaction, in which case the following function is used:

```
op flowDyn : Object Configuration -> Float .
```

The first argument of the function is the entity object itself; the second argument is the entire multiset of objects in the system.

Discrete transitions are modeled as ordinary rewrite rules; in particular, they should be understood as *instantaneous* rewrite rules that are assumed to model change that can be seen to take zero time. HI-Maude does not support any kind of “tick” rewrite rules to advance time; time advance is taken care of by the tool. To ensure that a rewrite rule is applied in a timely manner, HI-Maude provides the function

```
op timeCanAdvance : Configuration -> Bool .
```

so that if the user does *not* want time to advance when an object is in a certain state, (s)he must define `timeCanAdvance` to be `false` for those object states.

3.3. Formal analysis in HI-Maude

HI-Maude provides extensions of Maude’s (and Real-Time Maude’s) simulation and model checking features by allowing the user to select the numerical approximation technique used to approximate the continuous behaviors and the time increment used in the approximation.

HI-Maude’s hybrid *rewrite* command is used to simulate one behavior of the system from a given initial state `initState` up to a certain duration `timeLimit`:

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

where \sim is either `<=` or `<`; $\text{numMethod} \in \{\text{euler}, \text{rk2}, \text{rk4}\}$ is the numerical method used to approximate the continuous behaviors; and `stepSize` is the time increment used in the approximation of the continuous behaviors.

HI-Maude’s search command searches for (possibly up to n) states that are matched by a search pattern with a substitution that satisfies an (optional) condition and that can be reached from an initial state within a given time interval:

```
(hsearch [[n]] initState =>* searchPattern [such that cond] in time ~ timeLimit
  using numMethod stepsize stepSize .)
```

where $\sim \in \{<, <=, >, >=\}$, and *cond* is a condition on the variables in the search pattern. The following “find earliest” command finds the shortest time needed to reach a state matching a (possibly conditional) state pattern:

```
(hfind earliest init =>* pattern[suchthat cond] using numMethod stepsize stepSize .)
```

Finally, HI-Maude extends Maude’s *linear temporal logic model checker* to check whether each behavior (possibly “up to a certain time,” as explained in [21]) satisfies a temporal logic formula. *State propositions*, possibly parametrized, should be declared as operators of sort *Prop*, and their semantics should be given by equations of the form

```
eq {statePattern} |= prop = b . and ceq {statePattern} |= prop = b if cond .
```

for *b* a term of sort *Bool*, which defines the state proposition *prop* to hold in exactly those states $\{t\}$ where $\{t\} \models prop$ evaluates to *true*. A temporal logic *formula* is constructed by state propositions and temporal logic operators such as *True*, *False*, \sim (negation), \wedge , \vee , \rightarrow (implication), $[]$ (“always”), $<>$ (“eventually”), \cup (“until”), and \bar{w} (“weak until”). The time-bounded hybrid model checking command is written with syntax

```
(hmc initState |=t formula in time ~ timeLimit using numMethod stepsize stepSize .)
```

and checks whether the temporal logic formula *formula* holds in all behaviors—relative to the approximation of the continuous behaviors as explained below—up to duration *timeLimit* from the initial state *initState*.

3.3.1. Soundness and completeness of HI-Maude analyses

There is a trade-off between expressiveness and analytic power for hybrid systems. Model checkers and reachability analysis tools deal with very restricted fragments of hybrid systems, such as initialized rectangular hybrid automata (see [28] for a survey on decidable fragments of hybrid automata), to ensure that key properties are decidable. On the other hand, simulation tools have much more expressive modeling languages, but only provide simulation capabilities. HI-Maude is as expressive as simulation tools, yet provides reachability and LTL model checking analysis in addition to simulation. The price to pay is that reachability and satisfaction of LTL properties are in general no longer decidable.

HI-Maude only analyzes those behaviors that are possible with the selected time increment and numerical method used to approximate the continuous behaviors. Therefore, the results of search and model checking in HI-Maude may not be correct because of: (i) the approximation errors due to the use of numerical approximations, (ii) round-off errors due to the use of floating-point numbers, and (iii) the “sampling” approach to continuous time means that some behaviors may be missed.

4. The human thermoregulatory system

The human body needs to maintain a body temperature of around 37 °C to function normally. The metabolic heat production within the body is the only internal factor affecting body temperature of a healthy person. But abnormalities in brain function, chemical substances, and infectious diseases can alter the body temperature by affecting the temperature-regulating mechanisms of the body. The environment surrounding the body affects the body temperature by heat loss or gain through physical processes such as radiation (e.g., sun bathing), evaporation (of, e.g., sweat), convection (e.g., standing in the wind), and conduction (e.g., holding a piece of ice). *Hyperthermia* and *hypothermia* occur, respectively, when the body temperature increases, resp. decreases, significantly beyond normal [16].

Physiological and *behavioral* thermoregulation responds to changing environments in an attempt to ensure human survival and comfort. The primary control center of physiological thermoregulation is located in a portion of the brain called the hypothalamus. The hypothalamus enables mechanisms to support heat loss from the body when the body temperature is increasing above normal levels that include: increasing the diameter of blood vessels to let more blood flow underneath the skin (*vasodilation*), which promotes heat loss by radiation, convection, and conduction; and increasing sweat production, which promotes heat loss by evaporation. When the body temperature is decreasing, the hypothalamus enables the following mechanisms to reduce heat loss and increase heat production: decreasing the diameter of blood vessels to let less blood flow underneath the skin (*vasoconstriction*), and stimulating the skeletal muscles to cause shivering, which increases heat production by the body. *Behavioral* response to heat or cold stress include taking off some clothes or switch on a fan when the temperature is felt to be too hot, and putting on some more clothes or moving closer to fireplace when it feels too cold. The behavioral thermoregulation is related to a part of the brain called the cerebral cortex, which has the role in motor control of the body. (See Fig. 2.)

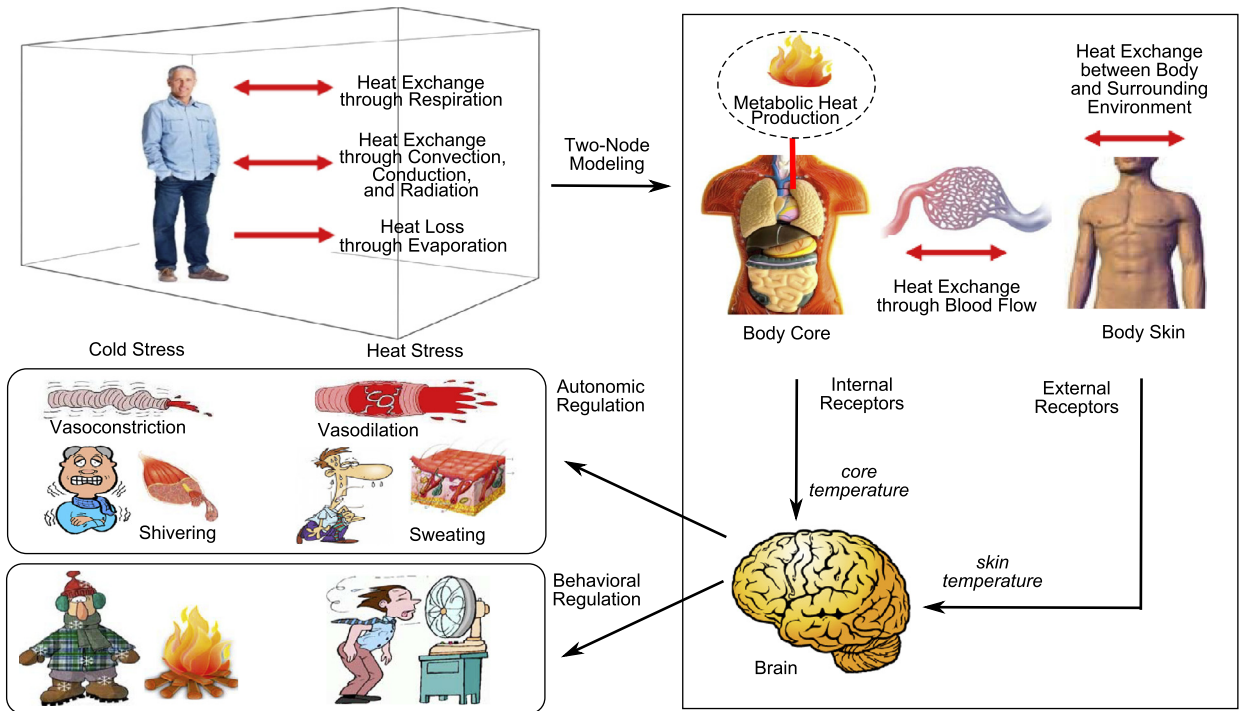


Fig. 2. Human thermoregulatory system.

The two-node modeling approach There are a number of approaches to model the human body and its thermoregulatory system, from considering the body to be one piece and up to more refined models where the human body is seen as consisting of a number of segments/parts.

In the well-established two-node Gagge model [9] the human body is considered as a cylinder with two concentric layers where the inner layer is the central core, and the outer layer is the skin shell. Heat exchange between the body and the environment takes place continuously at the skin surface. Heat generated inside the body is transferred to the skin surface through blood flow. From the skin, heat is transferred to the environment by convection, conduction, radiation, and sweat evaporation. There is also a direct thermal interaction between the body core and the environment through respiration. Heat in excess of that which can be dissipated is stored in the tissue, resulting in a rise of body temperature.

5. Modeling the human thermoregulatory system

This section presents our formal HI-Maude model of the human thermoregulatory system. Due to the size of the model, we can only show a small part of it; the entire executable model is available at <http://folk.uio.no/mohamf/HI-Maude/>, and is described in more detail in [29].

Using the two-node modeling approach to reason about the human thermoregulatory system, we model the body core, the body skin, and the surroundings as *thermal entities*, and the heat flow among these entities as *thermal interactions*, as shown in Fig. 3. Heat flows between the core and the skin through blood vessels, and between the body and the environment through respiration. Heat flows between the skin and the environment through convection, radiation, and evaporation. The heat production inside the body through metabolic processes and the heat production by muscles through shivering are represented as one-sided thermal interactions.

The human thermoregulatory system can to a certain extent be regarded as a *control system*, in which the brain (hypothalamus and cortex) gets information about the body and skin temperature from sensors and then give commands to effectors/actuators (in this case to the muscles to start/stop shivering, to the skin to start/stop sweating, and to the blood vessels to dilate or constrict the blood vessels). This control system part is highlighted in Fig. 3 by using shaded boxes and circles.

5.1. Thermal entities and interactions

It follows from the well known law $\Delta \dot{Q} = m \cdot c \cdot \Delta T$ of thermodynamics that the change of temperature of a thermal entity with mass m and specific heat capacity c is given by $\dot{T} = \frac{\sum \dot{Q}}{m \cdot c}$, where $\sum \dot{Q}$ is the total amount of heat lost or gained

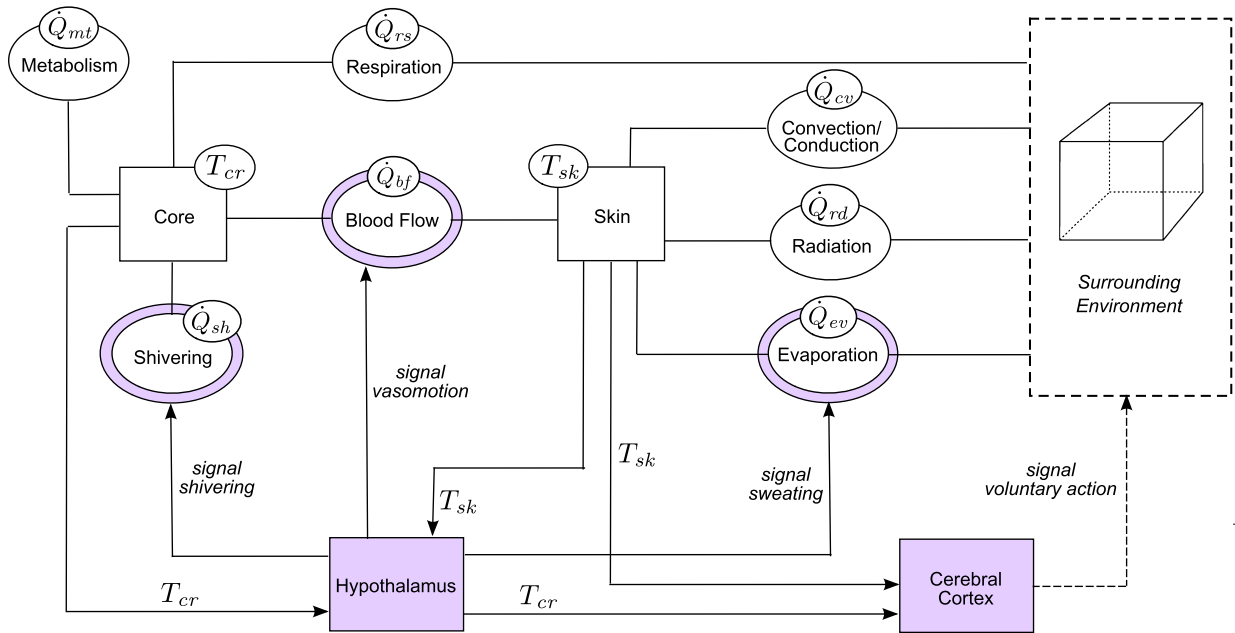


Fig. 3. Effort/flow model of the human thermoregulatory system.

per time unit. We therefore model a thermal entity, whose effort attribute denotes its temperature, by extending the built-in class `PhysicalEntity` with the entity's heat capacity and mass:

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

The continuous dynamics of the effort variable of the entity is defined as⁶

```
eq effortDyn(<TE : ThermalEntity | mass : MASS, heatCap : HC>, SF) = SF / (MASS * HC) .
```

A thermal interaction between two thermal entities is a two-sided interaction, with an additional attribute that denotes the area of the flow. Similarly, the source of heat flow to a thermal entity is a one-sided interaction:

```
class ThermalInteraction | area : Float .      subclass ThermalInteraction < TwoSidedInteraction .
class ThermalFlowSource | area : Float .      subclass ThermalFlowSource < OneSidedInteraction .
```

5.2. The body core

Since we need to add another continuous variable, denoting the amount of water in a person, to the body core (in addition to its temperature), we define the body core as a subclass of both `ThermalEntity` and `PhysicalEntityAC1`, where the new continuous attribute `contvar1` denotes the amount of water in the person. The body core component is defined by extending these classes with the entity's core state, body water state, the initial amount of water in the body, and some factor values for sweating, blood flow, and respiration:

```
class CoreHumanBody | coreState : CoreStateType,      bWaterState : BWaterStateType,
                    bWaterInit : Float,              sweatRateFactor : Float,
                    bloodFlowRateFactor : Float,     respiRateFactor : Float .
subclass CoreHumanBody < ThermalEntity PhysicalEntityAC1 .
```

We define the temperature-related (normal, mild/moderate/severe hyperthermia or hypothermia, or dead) and body-water-related (degrees of dehydration) states of the core:

⁶ In this paper we follow the Maude convention that variables are written with (only) capital letters, and do not show the variable declarations.

```

sorts CoreStateType BWaterStateType .
ops normal mildHyperthermia modHyperthermia sevHyperthermia
    mildHypothermia modHypothermia sevHypothermia dead : -> CoreStateType [ctor] .
ops normal modDehydration sevDehydration dead : -> BWaterStateType [ctor] .

```

The continuous dynamics of the body water of the component is defined as expected: the amount of water lost by the core equals the sweat *rate* of its SKIN times the area of its skin, times -1 (since water is *lost*). The interactions BloodFlow and Evaporation are introduced in Section 5.4; in the following rule the blood flow object is only used to associate the given body core to its skin:

```

eq contVar1Dyn(< CORE : CoreHumanBody | >,
    < BLF : BloodFlow | entity1 : CORE, entity2 : SKIN >
    < EVAP : Evaporation | entity1 : SKIN, sweatRate : SWTR, area : A >
    REST) = SWTR * -1.0 * A .

```

Changes in the body condition caused by temperature changes are represented as discrete events. For example, the core state changes from *mild hyperthermia* to *moderate hyperthermia* if the core temperature exceeds 38.9°C:

```

crl [normal-to-modhyperthermia] :
    < CORE : CoreHumanBody | effort : TEMP, coreState : mildHyperthermia >
=>
    < CORE : CoreHumanBody | coreState : modHyperthermia >
if TEMP > 38.9 .

```

The core experiences *severe hyperthermia* if the core temperature exceeds 40.6°C; this causes sweating to stop:

```

crl [modhyperthermia-to-sevhyperthermia] :
    < CORE : CoreHumanBody | effort : TEMP, coreState : modHyperthermia >
    < BLF : BloodFlow | entity1 : CORE, entity2 : SKIN >
    < EVAP : Evaporation | entity1 : SKIN >
=>
    < CORE : CoreHumanBody | coreState : sevHyperthermia >    < BLF : BloodFlow | >
    < EVAP : Evaporation | state : off >
if TEMP > 40.6 .

```

If the core temperature exceeds 44.0°C, the person reaches the *death* zone. The direct effect of this state is that the metabolic and respiratory processes stop:

```

crl [sevhyperthermia-to-death] :
    < CORE : CoreHumanBody | effort : TEMP, coreState : sevHyperthermia >
    < MET : Metabol | entity : CORE >
    < RESP : Respiration | entity1 : CORE >
=>
    < CORE : CoreHumanBody | coreState : dead >
    < MET : Metabol | state : stop >
    < RESP : Respiration | state : off >
if TEMP > 44.0 .

```

The treatment of the core temperature related to cold stress is similar.

To ensure that the above rules are applied in a timely manner, we use the built-in `timeCanAdvance` function to define, for each core state, when time can advance *without* a rule having to be taken. For example:

```

eq timeCanAdvance(<CORE : CoreHumanBody | effort : TEMP, coreState : sevHyperthermia >)
= TEMP > 40.6 and TEMP <= 44.0 .

```

Changes in the volume of water in the body may cause discrete changes in the body core (see Fig. 4). For example, the body water state changes to *severe dehydration* if the body has lost more than 10% of its initial amount of water:

```

crl [moddehydration-to-sevdehydration] :
    < CORE : CoreHumanBody | bWaterState : modDehydration,
                                contVar1 : WATERCUR, bWaterInit : WATERINIT >
=>
    < CORE : CoreHumanBody | bWaterState : sevDehydration >
if bWaterLoss(WATERCUR, WATERINIT) >= 0.1 .

```

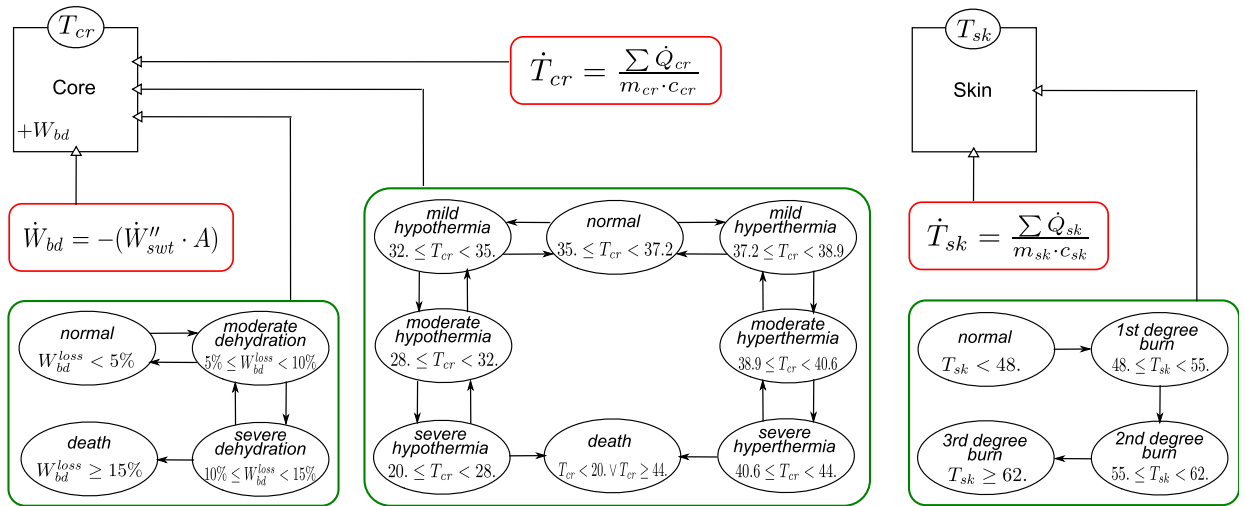


Fig. 4. The dynamics of the core and the skin.

where the function `bWaterLoss`, denoting the fraction of water the body has lost, is defined as expected:

```
op bWaterLoss : Float Float -> Float .
eq bWaterLoss(WATERCUR, WATERINIT) = (WATERINIT - WATERCUR) / WATERINIT .
```

If the loss of body water exceeds 15%, the person reaches the *death* zone; in particular, sweating, metabolism, and respiration all stop:

```
cr1 [sevdehydration-to-death] :
  < CORE : CoreHumanBody | bWaterState : sevDehydration,
    contVar1 : WATERCUR, bWaterInit : WATERINIT >
  < BLF : BloodFlow | entity1 : CORE, entity2 : SKIN >
  < SWEAT : Evaporation | entity1 : SKIN >
  < MET : Metabol | entity : CORE >
  < RESP : RespiHeatFlow | entity1 : CORE >
=>
  < CORE : CoreHumanBody | bWaterState : dead >
  < BLF : BloodFlow | state : off >
  < SWEAT : Evaporation | state : off >
  < MET : Metabol | state : stop >
  < RESP : RespiHeatFlow | state : off >
if bWaterLoss(WATERCUR, WATERINIT) >= 0.15 .
```

As always, the function `timeCanAdvance` must be defined to ensure that time progress stops when there is a change in the hydration state of a person.

5.3. The skin

The skin component is defined by extending the class `ThermalEntity` with an attribute for different degrees of burn injuries (normal or first/second/third degree burn):

```
class SkinHumanBody | skinState : SkinStateType .
subclass SkinHumanBody < ThermalEntity .

sort SkinStateType .
ops normal firstDBurn secondDBurn thirdDBurn : -> SkinStateType [ctor] .
```

Change in skin temperature may change the state of the skin, e.g., to *second degree burn* if the skin temperature exceeds 55.0°C; this causes evaporation to stop:

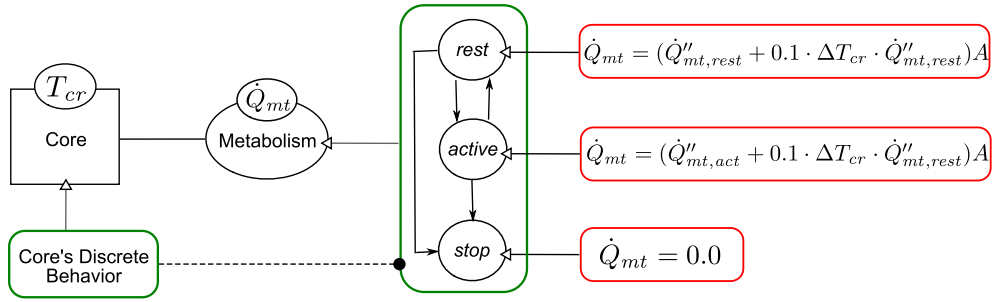


Fig. 5. The dynamics of heat production from the body metabolism.

```

crl [1st-degree-burn-to-2nd-degree] :
  < SKIN : SkinHumanBody | effort : TEMP, skinState : firstDBurn >
  < SWEAT : Evaporation | entity1 : SKIN >
=>
  < SKIN : SkinHumanBody | skinState : secondDBurn >
  < SWEAT : Evaporation | state : off >
if TEMP >= 55.0 .

```

The skin experiences *third degree burn* if its temperature exceeds 62.0°C.

5.4. Thermal interactions of the human body

This section gives an overview of the formal specification of the thermal interactions in the human body and between the body and its environment; these interactions are represented by the ovals in Fig. 3.

5.4.1. Metabolic heat production

The amount of metabolic heat produced by the human body is mainly dependent on the activity being performed. The basal metabolic rate is a function of mass, height, gender, and the body temperature, since it changes by about 10% per 1°C in body temperature [30]. The model of heat production by metabolism is shown in Fig. 5. Three discrete states represent states of the body when a person is at rest, doing some activity, or is dead. Metabolic heat production components are ThermalFlowSource components extended with attributes for the state and the heat production rates at rest and active conditions:

```

class Metabol | state : MetState, hfrRest : Float, hfrActive : Float .
subclass Metabol < ThermalFlowSource .

sort MetState .
ops rest active stop : -> MetState [ctor] .

```

The formal definition of the flowDyn function for the metabolism is somewhat tricky, since, as shown in Fig. 5, the flow is also a function of not only the temperature of the core, but also of the change ΔT_{cr} of the core's temperature. We therefore use the knowledge that the effP attribute is automatically added to the built-in PhysicalEntity component to denote the *previous* value of the effort variable (without this insider knowledge the user could of course add such an attribute himself):

```

eq flowDyn(< MET:Metabol | state:rest, hfrRest:HFREST, entity:CORE, area:A >,
  < CORE:CoreHumanBody | effort:TEMPCUR, effP:TEMPPREV > REST)
= (HFREST + ((TEMPCUR - TEMPPREV) * 0.1 * HFREST)) * A .

eq flowDyn(< MET:Metabol | state:active, hfrRest:HFREST, hfrActive:HFACT, entity:CORE, area:A >,
  < CORE:CoreHumanBody | effort:TEMPCUR, effP:TEMPPREV > REST)
= (HFACT + ((TEMPCUR - TEMPPREV) * 0.1 * HFREST)) * A .

eq flowDyn(< MET:Metabol | state:stop >, CONFIG) = 0.0 .

```

5.4.2. Heat production by shivering

The model of heat production by shivering is shown in Fig. 6. Two discrete states, with different continuous dynamics, represent whether shivering is active or not. The activation and deactivation of shivering is performed by the hypothalamus, which also determines the rate of shivering heat production. However, the discrete behavior of the core may determine

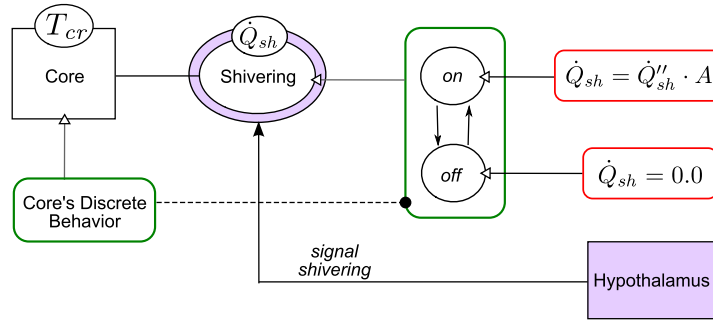


Fig. 6. The dynamics of heat production from the shivering through the muscles.

the discrete behavior of the shivering component; e.g., when the core's state is severe hypothermia or dead, there is no shivering.

The shivering heat production component is defined by extending the class `ThermalFlowSource` with attributes for discrete state and shivering rate:

```
class Shivering | state : OnOff, shiverRate : Float .
subclass Shivering < ThermalFlowSource .
```

The continuous dynamics of the flow variable is defined as follows:

```
eq flowDyn(< SHV : Shivering | state : on, shiverRate : SHVR, area : A >, EFF) = SHVR * A .
eq flowDyn(< SHV : Shivering | state : off >, EFF) = 0.0 .
```

Three instantaneous rules model the discrete behaviors of the shivering component. The shivering receives a signal from the hypothalamus to activate or deactivate the shivering process. For example, when it receives an 'on' signal, the shivering process is activated if the core is not severely hypothermic or dead:

```
rl [activate-shivering] :
  signalShivering(SHIVER, on, SHVR)
  < SHIVER : Shivering | entity : CORE >
  < CORE : CoreHumanBody | coreState : CRST >
=>
  < SHIVER : Shivering | state : if CRST /= sevHypothermia and CRST /= dead
    then on else off fi, shiverRate : SHVR >
  < CORE : CoreHumanBody | > .
```

5.4.3. Heat exchange through blood flow

The heat flow rate between the core and the skin through the blood vessels per square meter of body area can be computed using the equation $\dot{Q}_{bf} = (K + \dot{m}_{bl} \cdot c_{bl})(T_{cr} - T_{sk})$, where K is thermal conductivity between core and skin, \dot{m}_{bl} is the blood flow rate and c_{bl} is blood specific heat capacity [10]. The model of heat exchange through blood flow is shown in Fig. 7. Two discrete states represent whether the blood flow is active or not, since blood stops flowing when a person is dead (since the heart cannot pump blood anymore). The equation below for state `off` shows that even though the blood stops flowing, heat transfer through the blood can still happen as long as there is the temperature difference between the core and the skin. Increasing or decreasing blood flow through vasodilation or vasoconstriction is managed by the hypothalamus which determines the value of the blood flow rate.

The blood flow component is defined by extending the class `ThermalInteraction` with attributes for the blood flow rate, the blood thermal conductivity, and the blood heat capacity:

```
class BloodFlow | state : OnOff, conduct : Float, heatCap : Float, bloodFlowRate : Float .
subclass BloodFlow < ThermalInteraction .
```

The continuous dynamics of the flow variable is defined in the usual way:

```
eq flowDyn(< BLF : BloodFlow | state : on, conduct : COND, heatCap : HC, bloodFlowRate : BFR, area : A >,
  TEMPCR, TEMPSK) = (COND + HC * BFR) * (TEMPCR - TEMPSK) * A .

eq flowDyn(< BLF : BloodFlow | state : off, conduct : COND, area : A >, TEMPCR, TEMPSK)
  = COND * (TEMPCR - TEMPSK) * A .
```

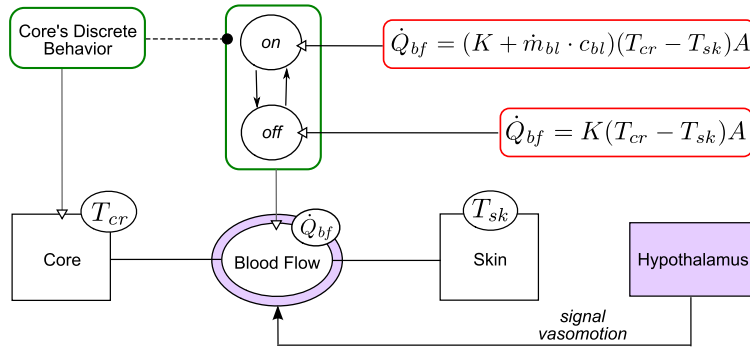


Fig. 7. The dynamics of heat exchange through blood flow.

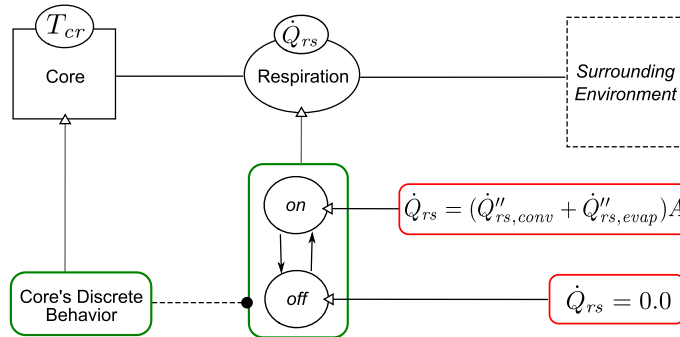


Fig. 8. The dynamics of heat exchange through respiration.

An instantaneous rule defines the behavior of the blood flow component when it receives a signal (message) from the hypothalamus containing the value of the desired blood flow rate:

```

r1 [vaso] :
  signalVaso(BLF, BLFR)
  < BLF : BloodFlow | entity1 : CORE >
  < CORE : CoreHumanBody | coreState:CRST, bWaterState:BWST >
=>
  < BLF:BloodFlow | bloodFlowRate:if CRST!=dead and BWST!=deaththen BLFRelse 0.0 fi >
  < CORE : CoreHumanBody | > .

```

5.4.4. Heat exchange through respiration

During respiration, heat transfer occurs through convection and evaporation of heat and water vapor from the respiratory tract to the inhaled air. Air is inhaled at ambient condition and exhaled at a temperature only slightly cooler than the core temperature. There is an approach to compute the respiratory heat flow through some approximations to determine the value of convection C_{res} and evaporation E_{res} by $C_{res} = 0.0014 \cdot M(T_{ex} - T_a)$ and $E_{res} = 0.0173 \cdot M(P_{ex} - P_a)$, respectively, where T_{ex} is exhaled air temperature, T_a is air temperature, P_{ex} is water vapor pressure at exhaled air temperature, and P_a is water vapor pressure [31,32]. The model of heat exchange through respiration is shown in Fig. 8. As expected, the class `RespiHeatFlow` is a subclass of `ThermalInteraction` and its formal specification is given in our report [29].

5.4.5. Heat exchange through convection

The human body gains or loses heat from/to the surrounding air through convection. For a person wearing clothes, the heat exchange passes first through from the skin surface to the outer clothing surface, and then from the outer clothing surface to the environment. Therefore, the heat flow rate computation needs to take into account some properties related to clothing. The heat flow rate of the convection can be computed using the equation $\dot{Q}_{cv} = \left(\frac{T_{sk} - T_{am}}{R_{cl} + 1.0 / (f_{cl} \cdot h_{cv})} \right)$ for each square meter of body area, where R_{cl} is the thermal resistance of clothing, f_{cl} is the clothing area factor, and h_{cv} is the convective heat transfer coefficient [11]. For a nude person, R_{cl} is 0.0 and f_{cl} is 1.0. The model of heat exchange through convection is shown in Fig. 9. The convection component is defined by extending `ThermalInteraction` with attributes for clothing resistance, clothing area factor, relative velocity, and air pressure:

```

class Convection | relVelocity:Float, clothResist:Float, clothArea:Float, airPressure:Float .
subclass Convection < ThermalInteraction .

```

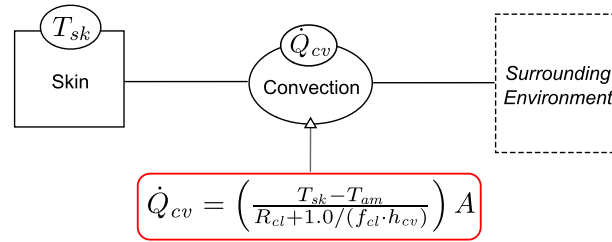


Fig. 9. The dynamics of heat exchange from convection.

The relative velocity and air/gas pressure in the room are needed to compute the convection coefficient used in the heat flow rate equation. To get a modular specification, we assume that each such room has two functions

```
ops gasPressure gasVelocity : Object -> Float [frozen (1)] .
```

which give the air/gas pressure and the velocity of the air/gas in the room (or the outside environment). Section 6 shows the definition of these functions for our sauna room.

```
ceq flowDyn(< CONV : Convection | entity1 : SKIN, entity2 : SROOM, area : A,
            clothResist : RCL, clothArea : ACL >,
  < SKIN : SkinHumanBody | effort : TEMPSK >
  < ROOM : ThermalEntity | effort : TEMPAM >
  REST) =
  ((TEMPSK - TEMPAM) / (RCL + 1.0 / (ACL * convectionCoeff(PRES, VEL)))) * A
if PRES := gasPressure(< ROOM : ThermalEntity | >)
  /\ VEL := gasVelocity(< ROOM : ThermalEntity | >) .
```

5.4.6. Heat exchange by radiation

The heat flow rate by radiation also depends on the clothing, and is defined by equation $\dot{Q}_{rd} = h_{rd} \cdot f_{rd}(T_{cl} - T_{rd})$ for each square meter of the body area, where h_{rd} is the radiative heat transfer coefficient, f_{rd} is the view (or posture) factor, T_{cl} is the cloth temperature, and T_{rd} is the radiant temperature. Again, the corresponding class `Radiation` is a subclass of `ThermalInteraction` and is described in our report [29].

5.4.7. Heat loss by evaporation

The heat loss from the body through evaporation is the combination of heat loss from sweating and heat loss from water diffusion through the skin. Sweating is controlled by the hypothalamus, whereas diffusion is not. The rate of sweating heat loss per square meter of body area is computed using the equation $\dot{Q}_{ev,swt}'' = \dot{m}_{swt} \cdot h_{vap}$, where \dot{m}_{swt} is the rate of sweat generation, and h_{vap} is the specific heat of vaporization. Meanwhile, the rate of skin diffusion heat loss per square meter of body area is computed using the equation

$$\dot{Q}_{ev,dif}'' = 0.06 \cdot \left[\left(\frac{p_{sk} - p_{am}}{R_{cl} + 1.0 / (f_{cl} \cdot h_e)} \right) - \dot{Q}_{ev,swt}'' \right]$$

where p_{sk} and p_{am} are the water vapor pressure at skin and in ambient air, respectively, R_{cl} is the heat transfer resistance of clothing layer, f_{cl} is the clothing area factor, and h_e is the evaporative heat transfer coefficient (the equation is derived from equations in [31]).

The model of heat loss through evaporation is shown in Fig. 10. When sweating is inactive, the heat loss is only from diffusion. Activation and deactivation of sweating is controlled by the hypothalamus.

The evaporation component is defined by extending `ThermalInteraction` with the following attributes: state for representing active and inactive condition; sweating rate and specific heat of vaporization for computing the heat loss rate from sweating; relative velocity, clothing resistance and factor, air pressure, and latent heat evaporation for computing heat loss rate from skin diffusion:

```
class Evaporation | state : OnOff,          heatVap : Float,          sweatRate : Float,
  relVelocity : Float,  clothResist : Float,  clothArea : Float,
  airPressure : Float,  latentHeatEvap : Float .
subclass Evaporation < ThermalInteraction .
```

From a formal modeling perspective, the definition of the continuous dynamics of the flow variable of the evaporation component is similar to that for the convection component and is therefore not shown.

When the evaporation component receives a signal from the hypothalamus to *activate* sweating, sweating is activated if the core is not severely hyperthermic or dead, and the skin is not burnt:

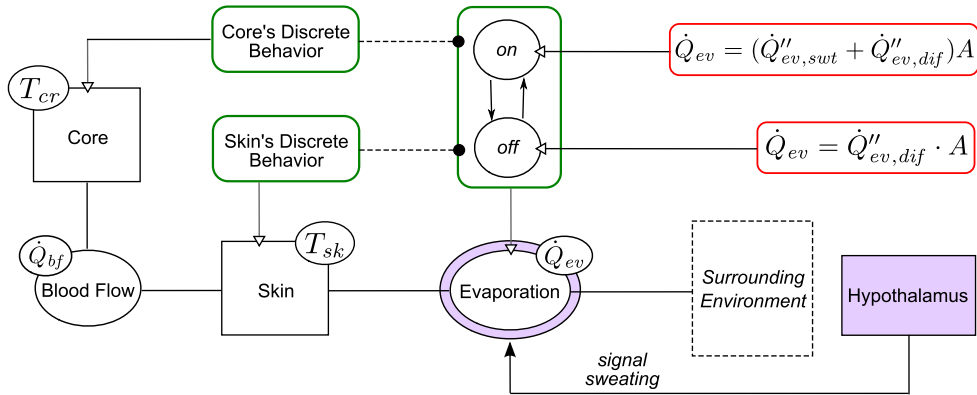


Fig. 10. The dynamics of heat loss from evaporation through the skin.

```

rl [activate-sweating] :
  signalSweating(SWEAT, on, SWTR)
  < SWEAT : Evaporation | entity1 : SKIN >
  < SKIN : SkinHumanBody | skinState : SKST >
  < BLF : BloodFlow | entity1 : CORE, entity2 : SKIN >
  < CORE : CoreHumanBody | coreState : CRST, bWaterState : BWST, sweatRateFactor : SWRF >
=>
  if CRST /= sevHyperthermia and CRST /= dead and SKST /= secondDBurn
    and SKST /= thirdDBurn and BWST /= sevDehydration and BWST /= dead
  then < SWEAT : Evaporation | state : on, sweatRate : SWRF * SWTR >
  else < SWEAT : Evaporation | sweatRate : 0.0 > fi
  < SKIN : SkinHumanBody | > < BLF : BloodFlow | >
  < CORE : CoreHumanBody | > .

```

5.5. The controllers

To model the regulatory process in the human body, we have defined a control system with *controllers*, *sensors*, and *actuators*. A sensor is connected to a component in a physical system to monitor the value of some variable and to periodically send the value to one or more controllers. A controller receives information from one or more sensors, and performs the controlling actions by sending messages/signals to one or more actuators.

A control system approach to human homeostasis, including thermoregulation, is quite common to explain how the brain controls our body. Basically, the controllers (hypothalamus and cortex) periodically receive a message from the receptors in Fig. 3 about the current temperature of the body core (message `tempValCore`) and the skin (message `tempValSkin`). We abstract from the small signaling delays in the body.⁷ Since we use a message/signal-passing infrastructure, we therefore add the equation

```
eq timeCanAdvance(MSG) = false .
```

(where `MSG` is a variable of the sort `MSG` denoting messages) to ensure that messages are read without delay.

5.5.1. Sensors

We add a `Sensor` object to each component being “sensed.” This object should ensure that the corresponding sensed value is sent to its controller(s) at regular intervals (every time unit in our case study). Such sensor objects are object instances of the following class `Sensor`:

```
class Sensor | entity : Oid, timer : Int, period : Int, status : CtrlType .
```

The `entity` attribute denotes the physical system component whose variable is being observed/measured; `timer` and `period` are used to enable the controlling system periodically; and `status` denotes the current status of the sensor component (running, or inactive).

⁷ Another reason for abstracting from the small signaling delays is that we have not found good models for estimating such delays, especially given the fact the signals travel different distances (what is the distance between the brain and the “skin”?). Indeed, this “lumped model,” which abstracts from the signaling delays in the body, seems to be quite common.

We use the function `computeAfterEF` to decrease the timer value of a sensor; if the sensor's status is inactive, no change is needed (`computeAfterEF` ensures that in each time step the computation related to the controlling system is performed after the computation of physical system components):

```
eq computeAfterEF(< SEN : Sensor | timer : TMR, status : running > REST) =
  < SEN : Sensor | timer : TMR - 1 > computeAfterEF(REST) .
```

```
ceq computeAfterEF(< SEN : Sensor | status : inactive > REST) =
  < SEN : Sensor | > computeAfterEF(REST) .
```

To ensure that time cannot advance when a sensor must send its values, the function `timeCanAdvance` is false when the timer value of a running sensor is 0:

```
eq timeCanAdvance(< SEN : Sensor | status : SENSTA, timer : TMR >) =
  (SENSTA == inactive) or TMR /= 0 .
```

Essentially, the sensor object works as a timer of the component to which it belongs; when the timer is 0, that object must reset the timer and send its (updated) continuous value to the corresponding controller.

For example, when the timer of the `Sensor` object associated to the `SkinHumanBody` expires, the skin component must send the (now updated) value of the temperature of the skin to the *hypothalamus* and the *cortex* components. We therefore first define the sensor object of the skin, which also contains the identifiers of the hypothalamus and the cortex objects:

```
class SkinSensor | cortex : Oid, hypothalamus : Oid .
subclass SkinSensor < Sensor .
```

When the `SkinSensor` timer is 0, the skin component sends a `tempValSkin` message to the cortex and the hypothalamus, and resets the timer:

```
rl [sendSkinTemp] :
  < SKINSENSOR : SkinSensor | entity : SKIN, timer : 0, status : running, period : P,
    hypothalamus : HYPOTHAL, cortex : CORTEX >
  < SKIN : SkinHumanBody | effort : TEMP >
=>
  < SKINSENSOR : SkinSensor | timer : P >
  < SKIN : SkinHumanBody | >
  tempValSkin(HYPOTHAL, TEMP)
  tempValSkin(CORTEX, TEMP) .
```

The same infrastructure is not needed for controllers and actuators, since time does not advance when there are messages in the system, so that “controllers” and “actuators” will treat their messages in a timely way.

5.5.2. The hypothalamus

The *hypothalamus* component is defined by extending the controller class with attributes for temperature set points⁸ for the core and the skin. The hypothalamus component is triggered when it receives the core and skin temperature values. If the hypothalamus is not inactive, it uses these values to compute the appropriate messages/signals to send to the *sweating*, *shivering*, and *blood flow* components, and to send the core temperature value to the *cortex*. When the hypothalamus is inactive, no signal is sent except for a message to the cortex stating that it is inactive:

```
class Hypothalamus | status : OnOff, core : Oid, setPointCore : Float, setPointSkin : Float .
subclass Hypothalamus < Controller .
```

```
rl [hypo-manages-involuntary-thermoreg] :
  tempValCore(HYPOTHAL, TEMPCR)
  tempValSkin(HYPOTHAL, TEMPSK)
  < HYPOTHAL : Hypothalamus | status : S, core : CORE, setPointCore : SPCR, setPointSkin : SPSK >
  < SWEAT : Evaporation | entity1 : SKIN >
  < SHIVER : Shivering | entity1 : CORE >
  < BLF : BloodFlow | entity1 : CORE, entity2 : SKIN >
```

⁸ Set points are the desired core/skin temperatures that we want to maintain; the set point for the body core can be higher than 37°C during, e.g., illness.

```

< CORTEX : Cortex | dataProvider : HYPOTHAL >
=>
< HYPOTHAL : Hypothalamus | > < SWEAT : EvapSkinSweating | >
< SHIVER : Shivering | > < BLF : BloodFlow | > < CORTEX : Cortex | >
if S == on then
  signalVaso(BLF, bloodFlowRate(TEMPCR, TEMPSK, SPCR, SPSK))
  makeSignalSweating(SWEAT, TEMPPCR, TEMPSK, SPCR, SPSK)
  makeSignalShivering(SHIVER, TEMPPCR, TEMPSK, SPCR, SPSK)
  tempValCoreFromHypothalamus(CORTEX, TEMPPCR)
else hypothalamusInactive(CORTEX) fi .

```

Some of the “messages” above are functions which generate the appropriate message. For example, when the body temperature is considered too hot, a signal containing the `on` value and the sweat rate value is sent to the *sweating* component. If the body temperature is considered fine, only the `off` value is sent. The treatment is similar for messages to the *shivering* component:

```
ops makeSignalSweating makeSignalShivering : Oid Float Float Float Float -> Msg .
```

```
msg signalVaso : Oid Float -> Msg .
msgs signalSweating signalShivering : Oid OnOff Float -> Msg .
msg hypothalamusInactive : Oid -> Msg .
```

```
ceq makeSignalSweating(SWEAT, TEMPPCR, TEMPSK, SPCR, SPSK) =
  signalSweating(SWEAT, if SWTR /= 0.0 then on else off fi, SWTR)
  if SWTR := sweatRate(TEMPPCR, TEMPSK, SPCR, SPSK) .

ceq makeSignalShivering(SHIVER, TEMPPCR, TEMPSK, SPCR, SPSK) =
  signalShivering(SHIVER, if SHVR /= 0.0 then on else off fi, SHVR)
  if SHVR := shiverRate(TEMPPCR, TEMPSK, SPCR, SPSK) .

```

The function `bloodFlowRate`, governing vasodilation and vasoconstriction, is defined according to the equation of the blood flow rate for the two-node model as $\dot{m}_{bl} = \frac{1}{3600} [6.3 + \frac{200 \cdot WSIG_{cr}}{1 + 0.5 \cdot CSIG_{sk}}]$, where $WSIG_{cr}$ is the effector controlling signal for vasodilation, and $CSIG_{sk}$ is the effector controlling signal for vasoconstriction [10].

The equation for shivering heat production rate for the two-node model is $\dot{Q}_{sh}'' = 19.4 \cdot CSIG_{sk} \cdot CSIG_{cr}$ (in W/m^2 ; we multiply by 10^{-3} since we use kilowatts):

```
op shiverRate : Float Float Float Float -> Float .
eq shiverRate(TEMPPCR, TEMPSK, SPCR, SPSK) =
  19.4 * cSigSK(TEMPSK, SPSK) * cSigCR(TEMPPCR, SPCR) * 0.001 .

```

The function computing the sweat rate is based on [10] where it is defined as $\dot{m}_{sw} = 4.7 \cdot 10^{-5} \cdot WSIG_{body} \cdot \exp(\frac{WSIG_{sk}}{10.7})$:

```
op sweatRate : Float Float Float Float -> Float .
eq sweatRate(TEMPPCR, TEMPSK, SPCR, SPSK) =
  4.7 * 0.00001 * wSigBody(TEMPPCR, TEMPSK, SPCR, SPSK) * exp(wSigSK(TEMPSK, SPSK)) .

```

5.5.3. The cerebral cortex

Up to this point, we have modeled the involuntary actions of the human thermoregulatory system; this part should therefore be deterministic. We now turn to formalizing the cerebral *cortex* part of the brain that manages the behavioral (voluntary) actions of the thermoregulatory system. In particular, this voluntary part introduces nondeterminism in the model (at least at the abstraction level of this paper; we leave the question of free will/biological determinacy, etc., for future work): when its gets uncomfortable, a person with weak willpower, or one just having a bad day, could try to escape the discomfort, whereas another person may endure the discomfort for some purpose. This part is also novel compared to the previous ones in that we must formalize notions such as the comfort level and the tolerance level of a person.

The cortex component therefore has an attribute that represents thermal sensation/comfort value, a reference to the body core, as well as a measure for the tolerance limit of the person:

```
class Cortex | status : OnOff, tsensVal : Float, tsensToLimit : Float, metaBol : Oid,
  body : Oid .

```

The cortex is activated when it receives signals containing the skin and the core temperature values. The component computes the thermal comfort/sensation value, modeling the comfort level felt by the body. The empirical equation used in

this case study is based on skin temperature, core temperature, and metabolic rate values. We have not found a description of how to obtain the metabolic rate value for this purpose in literature on the human thermoregulatory system. We therefore assume that the cortex has access to the person's metabolic heat production component (attribute `metaBol`). If the computed (dis)comfort is within the tolerance limit, no voluntary action is performed. If, on the other hand, the computed discomfort value is greater than the threshold for heat discomfort, a voluntary action could be performed. In our case study, this action can be to exit into a cooler place. We formalize this nondeterminism by two rewrite rules that are both enabled when it is getting unpleasant: the rule `cortex-tsens-too-hot-do-something` models a quick escape to a more comfortable location (or doing something else, like turning on a fan despite soaring costs of electric power), and the rule `cortex-tsens-too-hot-do-nothing` models that no action is taken:

```
cr1 [cortex-tsens-too-hot-do-something] :
  tempValSkin(CORTEX, TEMPSK)
  tempValCoreFromHypothalamus(CORTEX, TEMPCR)
  < CORTEX : Cortex | status: on, tsensTolLimit : TSENSTOL, metaBol : MET, body : CORE >
  < MET : Metabol | flow : METAB, area : A >
=>
  < CORTEX : Cortex | tsensVal : TSENS >          < MET : Metabol | >
  exitToCooler(CORE)          --- do something!
if TSENS := tsens(TEMPSK, TEMPCR, METAB / A)
  /\ TSENS >= TSENSTOL .
```

Since it is a voluntary action, the cortex may decide *not* to go to the cooler room:

```
cr1 [cortex-tsens-too-hot-do-nothing] :
  tempValSkin(CORTEX, TEMPSK)
  tempValCoreFromHypothalamus(CORTEX, TEMPCR)
  < CORTEX : Cortex | tsensTolLimit : TSENSTOL, metaBol : MET >
  < MET : Metabol | flow : METAB, area : A >
=>
  < CORTEX : Cortex | tsensVal : TSENS >          < MET : Metabol | >
if TSENS := tsens(TEMPSK, TEMPCR, METAB / A)
  /\ TSENS >= TSENSTOL .
```

The computation estimating the thermal comfort or sensation is based on the model described in [31] that uses empirical expressions to predict thermal sensation (*TSENS*). The indices are based on 11-point numerical scales, where positive values represent the warm side of neutral comfort, and negative values represent the cool side. The indices are 0: comfortable; 1: slightly uncomfortable but acceptable; 2: uncomfortable and unpleasant; 3: very uncomfortable; 4: limited tolerance; and 5: intolerable. The *TSENS* value is computed by the equation

$$TSENS = \begin{cases} 0.4685(t_{body} - s_{ec}) & \text{if } t_{body} < s_{ec} \\ 4.7 \cdot E_e \frac{(t_{body} - s_{ec})}{(s_{eh} - s_{ec})} & \text{if } s_{ec} \leq t_{body} \leq s_{eh} \\ 4.7 \cdot E_e + 0.4685(t_{body} - s_{eh}) & \text{if } s_{eh} < t_{body} \end{cases}$$

where t_{body} is mean body temperature, s_{ec} and s_{eh} are cold and hot set points representing the lower and upper limits for the zone of evaporative regulation, respectively, and E_e is the evaporative efficiency. The function `tsens` is defined according to the above equation.

The cold and hot set points of the zone of evaporative regulation, s_{ec} and s_{eh} , respectively, are computed by $s_{ec} = \frac{0.194}{58.15}(M - W) + 36.301$ and $s_{eh} = \frac{0.347}{58.15}(M - W) + 36.669$, where $M - W$ is the net rate of internal heat production (W represents the heat loss from the work done by the body to the environment).

6. Modeling participation in the Sauna World Championships

In this section we formally model in HI-Maude the kind of sauna used in the 2010 Sauna World Championships, its thermal connections, as well as an outdoors environment and its thermal interactions. We do not have any official information about the sauna used in the Sauna World Championships, but after researching literature, product descriptions, etc., we have obtained information that we use to model the sauna.

6.1. The 2010 Sauna World Championships

The Sauna World Championships were an annual event held in Heinola, Finland. The winner is the contestant who can stay the longest in an oppressively hot and humid sauna. Before the torturing game starts, the sauna is pre-heated to 110°C (warmer than the boiling point for blood). To make the conditions even worse, every thirty seconds half a liter of water



Fig. 11. From the 2010 Sauna World Championships in Heinola, Finland, with the finalist who died the day after the event on the left and the surviving finalist in the middle.

is poured onto the hot sauna rocks which are the heat source of the sauna. The intense vapor from this water increases the humidity of the sauna, which makes it more difficult for the participants' sweat to evaporate. The world record of 18 minutes and 15 seconds was set in the 2008 championships. The championship in 2010 ended in a tragedy. The two last finalists collapsed with severe burn injuries after about six minutes. One of them died a day later, and the other, a five-time champion, survived after two months in coma, with serious damage to his skin, lungs and kidneys. They both were conscious but unable to get out the sauna on their own. The cause of this tragedy is still under investigation. The temperature in the sauna was similar to those in previous years and the times of the competitors were about the same, until this terrible final round.

For more information about the event, we refer the reader to 11-time National Sportswriter of the Year Rick Reilly's book "Sports from Hell: My Two-Year Search for the World's Dumbest Competition" [33] and his article about the 2010 event, "The point of no return" [34].

6.2. Modeling the sauna

The sauna is modeled as a room which uses special rocks for providing heat to the room, as shown in Fig. 12. The rocks are heated by a heater, which is connected to a control system that manages the temperature of the room. Some amount of water is poured on the rocks periodically. The action of exiting the sauna is modeled by dynamically changing the thermal interactions between the human body and the sauna to interactions between the human and the outside environment.

6.2.1. Thermal entities

The thermal entity for the sauna room is represented differently compared to other thermal entity components. For example, there are two attributes for the mass: for the dry air and the water vapor, since we want to model the effect of pouring water on the rocks:

```
class SaunaRoom | massDryAir: Float,      spHeatAir: Float,      massWaterVap: Float,
                  spHeatWater: Float,    relHumid: Float,      timer: Int,           period: Int,
                  gcDryAir: Float,       gcWaterVap: Float,    relVelocity: Float,   vol: Float .
subclass SaunaRoom < PhysicalEntity .
```

The attributes `massDryAir` and `massWaterVap` specify the mass of dry air and of water vapor, respectively; `spHeatAir` and `spHeatWater` specify the specific heat of air and water, respectively; `relHumid` keeps the value of relative humidity of the room; `timer` is used for triggering periodical events; `vol` denotes the volume of the sauna room; `relVelocity` is the relative velocity of the air in the sauna; and `gcDryAir` and `gcWaterVap` are the gas constants of dry air and water vapor, respectively.

The continuous dynamics of the effort variable of the sauna is defined as

```
ceq effortDyn(< ROOM : SaunaRoom | massDryAir : MASSDA,      spHeatAir : SHDA,
```

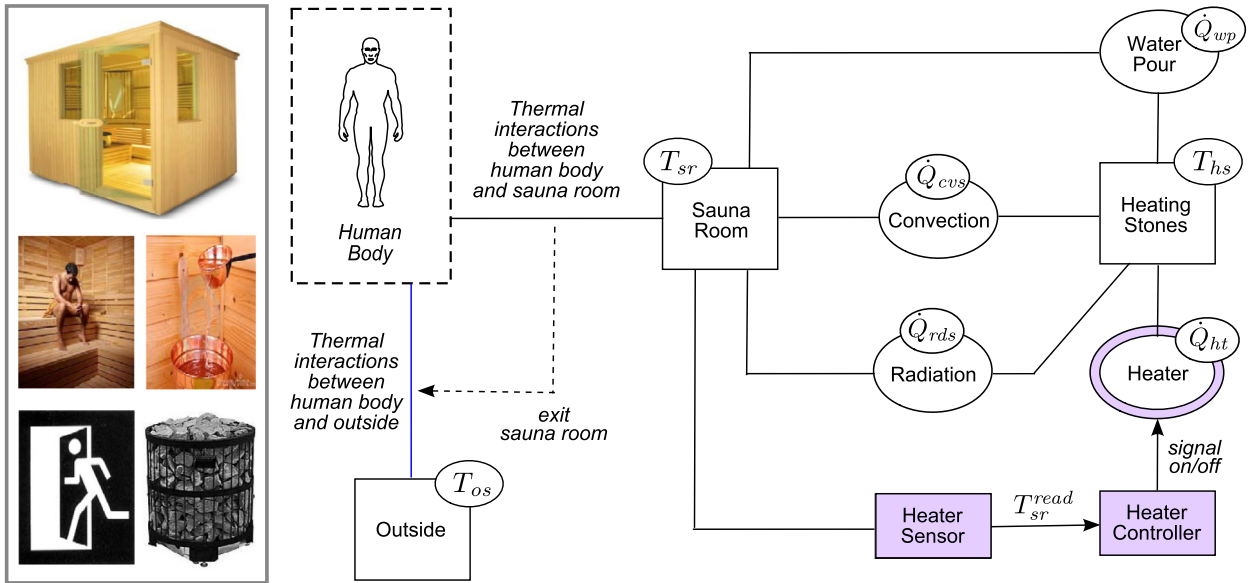


Fig. 12. The sauna.

```

massWaterVap : MASSWV, spHeatWater : SHWV >, SF)
= SF / ((MASSDA + MASSWV) * specHeatHumid(SHDA, SHWV, HUMSPEC))
if HUMSPEC := humidRatio2(MASSDA, MASSWV) .

```

The dynamics above is based on the basic equation for thermal entities. However, the fact that the mass of the room is a combination of the mass of the dry air and of the water vapor must be taken into account in the computation. The function `specHeatHumid` computes the specific heat of a mix of dry air and water vapor. The function `humidRatio` computes the humidity ratio of the mix.

The *sauna rocks* component is defined as a simple thermal entity:

```

class SaunaRocks .      subclass SaunaRocks < ThermalEntity .

```

Heat transfer from the rocks to the sauna occurs through convection and radiation. Unlike for the human body, we use basic forms of convection and radiation:

```

class ConvectionBasic | convectCoeff : Float .
class RadiationBasic | emmissiv : Float .
subclass ConvectionBasic RadiationBasic < ThermalInteraction .

eq flowDyn(< CONV : ConvectionBasic | convectCoeff : COEFF, area : A >, TEMP1, TEMP2)
= COEFF * (TEMP1 - TEMP2) * A .

eq flowDyn(< RAD : RadiationBasic | emmissiv : EMMI, area : A >, TEMP1, TEMP2)
= EMMI * stefBoltzConst * A * ((TEMP1 ^ 4.0) - (TEMP2 ^ 4.0)) .

```

We must also define the functions `gasPressure` and `gasVelocity` that give the gas pressure and the velocity of the air in the sauna as follows; these functions are used by any outside entity that interacts thermally with the heater through convection (such as the human specified in Section 5):

```

eq gasPressure(< SAUNA : SaunaRoom | effort : TEMP,      massDryAir : MASSDA,      vol : VOL,
               cgDryAir : CGDA,      massWaterVap : MASSWV,      cgWaterVap : CGWV >)
= gasPressure(MASSDA, CGDA, TEMP, VOL) + gasPressure(MASSWV, CGWV, TEMP, VOL) .
eq gasVelocity(< SAUNA : SaunaRoom | relVelocity : VEL >) = VEL .

```

where `gasPressure(MASSDA, CGDA, TEMP, VOL)` and `gasPressure(MASSWV, CGWV, TEMP, VOL)` compute the gas pressure of the air and vapor, respectively.

6.2.2. Pouring water

We only consider the thermal effect of pouring water on the rocks, and make some simplifying assumptions: the effect of pouring water is a heat loss for the heating rocks and a heat gain for the sauna room; all the water is vaporized; and the vaporization process is instantaneous. The *water pouring* component is defined as a thermal interaction between the heating rocks and the sauna room component:

```
class WaterPouringHeatLoss | mass : Float,      temp : Float,      heatCap : Float,
                             heatEvap : Float,  timer : Int,      period : Int .
subclass WaterPouringHeatLoss < PhysicalInteraction .
```

The attribute `mass` specifies the amount of water poured; `temp` defines the temperature of the water; `heatCap` and `heatEvap` represent the heat capacity and the latent heat evaporation of the water, respectively; and `timer` and the `period` are used to periodically trigger the water pouring.

The flow variable of the pouring water component represents the rate of heat flow from the heating rocks to the sauna room. Two definitions are needed for this continuous variable: one for the event where the water is poured, and one for the period of time between two water pouring events.

```
ceq flowDyn(< WPHL : WaterPouringHeatLoss | mass: MASS, temp: TEMP, heatCap: HC, timer: TMR,
            heatEvap: HEVAP >, EFF1, EFF2) = HEATLOSS
  if TMR == 0 /\
    HEATLOSS := MASS * HC * (100.0 - TEMP) + MASS * HEVAP * factorPourWater(EFF1, TEMP) .

ceq flowDyn(< WPHL : WaterPouringHeatLoss | timer: TMR >, EFF1, EFF2) = 0.0 if TMR /= 0 .
```

When the water is poured, assuming that the water is vaporized at once, the heat transferred from the rocks to the room is the sum of the heat needed to increase the water to the boiling point and the heat needed to change the water from the liquid to vapor. The function `factorPourWater` takes into account the current temperature of the heating rocks to compute the evaporation of the water (lower temperature of the heating rocks means less vaporization, then less heat transferred from the rocks to the room). When the water is not poured, there is no heat transferred from the heating rocks to the sauna room by vaporization.

The water is poured periodically. When the associated timer expires (i.e., becomes 0), water is poured on the rocks, and the vapor content of the sauna room increases by the amount of water poured (half a liter):

```
rl [add-watervap-to-sauna] :
  < ROOM : SaunaRoom | timer : 0,      massWaterVap : MASSWV,      period : PER >
=>
  < ROOM : SaunaRoom | timer : PER,    massWaterVap : MASSWV + 0.5 > .
```

The function `timeCanAdvance` must be used to ensure that the rule is applied at the moment when the timer expires:

```
eq timeCanAdvance(< ROOM : SaunaRoom | timer : TMR >) = TMR > 0 .
```

The other effect of pouring water to the heating rocks is an increase of relative humidity of the sauna room. The function `computeAfterEF` (which is a built-in function that is computed right after the new effort and flow values for a time increment have been computed) is used to update the value of the relative humidity, and of the timer (which should be synchronized with the timer of the pouring water component):

```
ceq computeAfterEF(< ROOM : SaunaRoom | timer : TMR,      effort : TEMPAM,      vol : VOL,
                  massWaterVap : MASSWV, cgWaterVap : CGWV,
                  massDryAir : MASSDA,  cgDryAir : CGDA > REST) =
  = < ROOM : SaunaRoom | timer : TMR - 1, relHumid : relativeHumidity(PRESTOT, PRESSAT, HUMSPEC) >
  computeAfterEF(REST)
  if TMR > 0 /\
    PRESWV := gasPressure(MASSWV, CGWV, TEMPAM, VOL)
    PRESDA := gasPressure(MASSDA, CGDA, TEMPAM, VOL)
    PRESTOT := PRESWV + PRESDA
    PRESSAT := waterVaporPressure(TEMPAM)
    HUMSPEC := humidRatio(MASSDA, MASSWV) .
```

6.2.3. The sauna heating system

The heating system considered here is an automatic control system which manages to keep the temperature of the sauna room at a specified value. The specification, given in detail in [29], is fairly standard (in the context of this paper!) and is not further explained.

6.2.4. Exiting the sauna

When a person in the sauna feels that the heat is becoming unbearable, he *may* leave the sauna if he is still able to do that (for example, a person suffering from heat stroke may not be able to move). When the person leaves the sauna, his body does not interact with the sauna room anymore, but with the environment outside the room. In our model, it means that the thermal interactions between the human body entities and the sauna must be redirected to other entities representing the outside environment. In particular, in Heinola the sauna opens directly to the outside, *not* to another room.

The “outside” component is defined as follows:

```
class Outside | relHumid : Float, airPressure : Float, humidRat : Float, velocityRel : Float .
subclass Outside < PhysicalEntity .
```

The attribute `relHumid` defines the relative humidity of the outside environment; `airPressure` is the outside air pressure; `humidRat` specifies the humidity ratio of air, i.e., the ratio between actual mass of water vapor present in moist air to the mass of dry air; and `velocityRel` is the relative velocity of air in the outside environment.

We assume that a contestant going outside will not change the outside temperature; i.e., the outside environment is assumed to have *constant* temperature. This is represented in the continuous dynamics of the effort variable of the outside component, whose time derivative is 0:

```
eq effortDyn(< OUT : Outside | >, SF) = 0.0 .
```

Likewise, the definition of the convection needs the functions `gasPressure` and `gasVelocity` for the outside as well:

```
eq gasPressure(< OUT : Outside | airPressure : PRESSURE >) = PRESSURE .
eq gasVelocity(< OUT : Outside | velocityRel : REL >) = REL .
```

Exiting the sauna is defined by rewrite rules which are activated when the signal from the cortex is received by the core. A person inside the sauna can open the door if his body is still able to do conscious motoric actions. For example, a person having severe hyperthermia or second degree burn is considered unable to move toward and to open the door for exiting the sauna. The following rule models the person opening the sauna door and exiting the sauna. For simplicity, we assume that this is an instantaneous action, and that no significant amount of cool air therefore enters the sauna. Exiting the sauna is modeled as expected by a rule that “redirects” all thermal connections between the person and the sauna to instead go between the person and the outdoors:

```
cr1 [exit-sauna-1] :
  exitToCooler(CORE, CORTEX)
  < CORE : CoreHumanBody | coreState : CRST, bWaterState : BWST >
  < BLF : BloodFlow | entity1 : CORE, entity2 : SKIN >
  < SKIN : SkinHumanBody | skinState : SKST >
  < SAUNA : SaunaRoom | > < OUT : Outside | >
  < RESP : RespiHeatFlow | entity1 : CORE, entity2 : SAUNA >
  < EVAP : Evaporation | entity1 : SKIN, entity2 : SAUNA >
  < CONV : Convection | entity1 : SKIN, entity2 : SAUNA >
  < RAD : Radiation | entity1 : SKIN, entity2 : SAUNA >
=>
  < CORE : CoreHumanBody | > < BLF : BloodFlow | > < SKIN : SkinHumanBody | >
  < SAUNA : SaunaRoom | > < OUT : Outside | >
  < RESP : RespiHeatFlow | entity2 : OUT > < EVAP : Evaporation | entity2 : OUT >
  < CONV : Convection | entity2 : OUT > < RAD : Radiation | entity2 : OUT >
  if CRST /= sevHyperthermia and CRST /= dead and BWST /= dead
  and SKST /= secondDBurn and SKST /= thirdDBurn .
```

Otherwise, the person is unable to open the door and exit when the `exitToCooler` signal from the cortex arrives; this is modeled by the expected rewrite rule that does nothing and is not shown.

7. HI-Maude analysis

This section shows how we can use HI-Maude and our model to analyze the ability of the human body to survive extreme conditions similar to those in the Sauna World Championships. In particular, we are interested in investigating three main issues:

1. How long can people in different states of health/practice stay in different kinds of saunas before problems will occur?

2. What happened at the 2010 Sauna World Championships? We propose some hypotheses to explain that accident and use HI-Maude to analyze those hypotheses.
3. To organize any kind of activity where heat (or cold) stress could be a problem—be it a sauna competition, an NFL training camp in late summer, or parachuting from the “stratosphere”—it is of course absolutely crucial that a person, even in excellent shape, feels significantly uncomfortable before heat-related problems may occur.

We are interested in analyzing these questions not only for normal people, but also for weak/sick people and for very fit people with considerable practice in the event. For example, the three-time champion Timo “The Great” Kaukonen used to travel with a mobile sauna where he practiced in 140 °C heat daily; we may also assume that an NFL Pro Bowler like Korey Stringer was in better shape than most “normal people.” An analysis of the 2010 accident that does not take this into account would be misleading.

In Section 7.1 we define the various parameters for different kinds of persons and for different settings of the sauna used in the Sauna World Championships. The physical parameters of the “normal” human body are chosen to be as close as possible to those of an average person. As already mentioned, we do not have any official information about the sauna environment used in the Sauna World Championships, but after researching literature, news, product descriptions, etc., we have obtained information that we use to define the parameter values. The physical parameters of the sauna are chosen to be as close as possible to existing sauna products.

In Section 7.2 we use HI-Maude simulations and further analysis to analyze how long different kinds of persons can stay in different kinds of saunas before various health problems occur.

In Section 7.3 we then propose some hypotheses that could explain the cause of the 2010 accident, and use HI-Maude analyses to analyze these hypotheses. Furthermore, we use HI-Maude analyses to also *find* the appropriate values of the parameters (temperature/humidity, etc.) of our hypotheses that could explain the death and the serious injury in the 2010 event.

Finally, in Section 7.4 we investigate whether persons will experience significant discomfort before the onset of heat-related injuries.

The experiments have been performed on a computer with an Intel Xeon 2.27 GHz. The analyses are carried out using the different numerical approximation methods with time increment one. The executable formal models, as well as the simulation and analysis commands described below, are available at <http://folk.uio.no/mohamf/HI-Maude/>.

7.1. Physical parameters

This section shows the values of different physical parameters used in our experiments. For the human body, we use the following values:

Parameter	Unit	Initial value
body mass	kg	70
core mass	kg	80% of body mass
skin mass	kg	20% of body mass
specific heat (core and skin)	kJ/(kg °C)	3.5
body water	liter	60% of body mass
core temperature	°C	37
skin temperature	°C	34
blood conductivity	kW/(m ² K)	0.00528
blood specific heat	kJ/(kg °C)	4.187
metabolic heat production at rest	kW/m ²	0.04
metabolic heat production when sitting	kW/m ²	0.06
clothing resistance (nude person)	m ² °C/kW	90.0
clothing area factor (nude person)	–	1.0
view factor (seated person)	–	0.7
latent heat evaporation (sweating)	kJ/kg	2455.0

The size of the area of the body is needed to compute the heat transfer between a person and his/her environment. There are two common approaches to compute the size of the area of the body:

1. By considering the human body as a single cylinder, the body area can be computed as long as the body height and the body diameter are known.
2. Using the *Dubois* function, which uses the body mass and the body height as parameters.

In this work, we use the Dubois function, which is a well established approximation in the field of human body modeling [10]:

```
op bodyAreaDubois : Float Float -> Float .
eq bodyAreaDubois(MASS, HEIGHT) = 0.202 * (MASS ^ 0.425) * (HEIGHT ^ 0.725) .
```

The survival of a person in extreme conditions depends on the fitness of his/her body. The fitness level may be influenced by age, experience, level of training, health condition, etc. For example, a trained athlete can have a sweating rate four times as large as the average person. A person with anhidrosis (the inability to sweat normally) may have much lower sweating rate than the average person. To model and simulate the effect of different fitness levels, we model three persons in different states of fitness (“Def” means the default value of the parameter, which is given in the table above):

	Ordinary	Trained, experienced	Weak, sick
Sweat rate	Def	4.0 * Def	0.5 * Def
Respiration rate	Def	0.5 * Def	2.0 * Def
Body water	Def	Def + 1.0	Def – 1.0
Temperature set points	Def	Def	Def + 2.0 (both core and skin)

To model a trained and experienced person, we:

- multiply the sweat rate by four, which increases the sweat production;
- decrease the respiration rate to model the ability to breath efficiently (less inspired air means less heat goes into the body); and
- adding body water (by drinking) will support heat loss from sweating.⁹

To model a sick and weak person:

- the sweat rate is set to half the standard value, which decreases the sweat production;
- the respiration rate is increased to model the inability to breath efficiently (more inspired air means more heat goes into the body);
- body water is reduced, to model, e.g., an excessive urination which is a symptom of some health problems; and
- the temperature set points are increased to model a person with high fever.

To model the sauna room we use the following values, which are based on some sauna cabin products:

Parameter	Unit	Initial value
room volume	m ³	35.0
air density	kg/m ³	1.0
dry air mass	kg	room volume * air density
dry air specific heat	kJ/(kg °C)	1.006
gas constant dry air	kJ/(kg °C)	0.28704
water vapor mass	kg	10.0
water vapor specific heat	kJ/(kg °C)	1.9
gas constant water vapor	kJ/(kg °C)	0.4615
air relative velocity	m/s	1.5
room temperature	°C	110.0
room door size	m ²	1.6

The following physical parameters of the heating system used in the experiments are based on real values found in product descriptions of sauna heating systems:

Parameter	Unit	Initial value
heater capacity	kW	16.5
sauna rocks mass	kg	200.0
sauna rocks specific heat (olivine diabase)	kJ/(kg K)	0.86
sauna rocks temperature	°C	110.0

The physical parameters of the water pouring are shown in the following table:

Parameter	Unit	Initial value
poured water mass	liter	0.5
water specific heat	kJ/(kg K)	4.2
water heat evaporation		2260.0
poured water temperature	°C	25.0

⁹ Kaukonen drank three to four gallons of water every day.

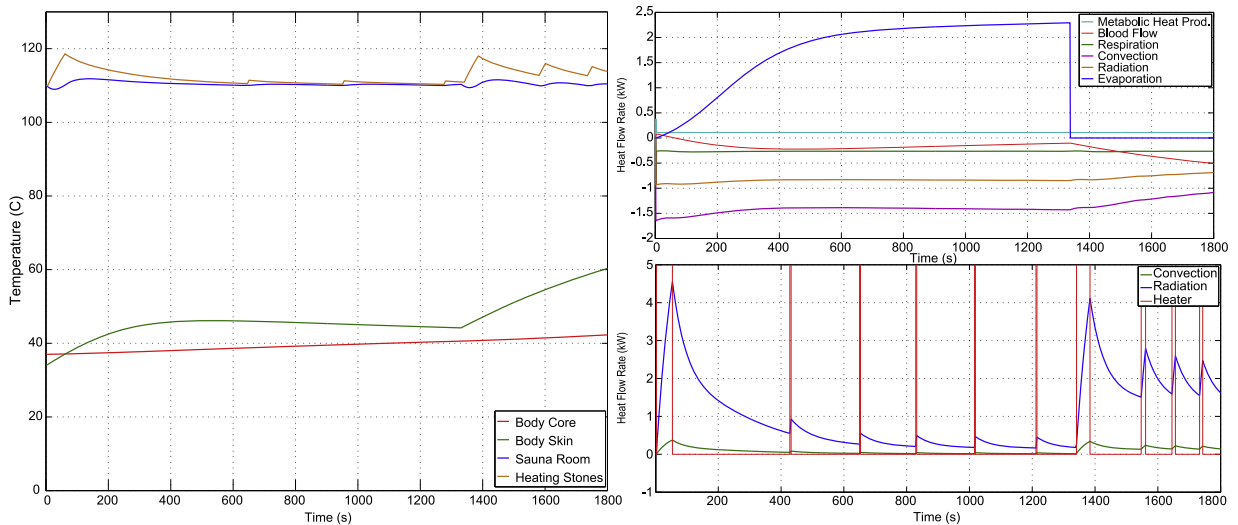


Fig. 13. Simulation results for the dry sauna: temperatures of the human body and the sauna environment thermal entities (left), and heat flow rates of the human body (top right) and of the sauna (bottom right) thermal interactions.

Physical parameters of the outside environment are shown in the following table:

Parameter	Unit	Initial value
relative humidity	–	10%
specific humidity	–	0.019826
air pressure	kPa	110.0
air relative velocity	m/s	1.5
convection coefficient (air, free)	kW/(m ² K)	0.025

7.2. Analyzing physiological thermoregulation

In this section we use HI-Maude to analyze how long the human body can survive in a sauna using only physiological thermoregulation. To analyze different aspects about the sauna, we model three kinds of saunas:

- a dry sauna, where no water is poured on the heating rocks;
- a moderately humid sauna, where half a liter of water is poured every 5 minutes;
- an extremely humid sauna, where half a liter of water is poured every 30 seconds.

To analyze what happens to our virtual experimental subject after 30 minutes in the sauna, we can use the HI-Maude simulation command

```
(hrew cs1 in time <= 1800 using euler stepsize 1.0 .)
```

where `cs1` is an initial state consisting of all appropriate physical entity and physical interaction objects with their respective initial attribute values (in this initial state, there is only one person in the sauna). Fig. 13 shows the simulation results for the dry sauna for the average person up to 30 minutes. In the beginning the skin can handle the heat from the sauna room well, while the core temperature increases slowly. However at some point between minute 20 and 23, the skin temperature increases drastically. As we see in the graph on the right, at that point the sweating stops, possibly due to severe hyperthermia or second degree skin burn. At the same point, the heat flow from the blood vessels transfers heat from the skin to the core at an increasing rate.

Fig. 14 shows the simulation results for the *moderately humid* sauna for the average person up to 30 minutes. We can see the effect of pouring water every 5 minutes on the rocks. The water causes heat from the rocks to be transferred to the room, which causes an instant increase of the room temperature, and an instant decrease in the temperature of the heating rocks. The drastic increase in the skin temperature occurs slightly before 20 minutes. The effect of pouring water can be seen clearly at the heat flows in the human body, in particular the heat flow directly connected to the room. An instant increase in the room temperature causes instant changes to evaporation, respiration, convection, and radiation.

Fig. 15 shows the simulation results for the *extremely humid* sauna. The intensive water pouring causes the drastic change to the skin temperature much earlier, and the core temperature increases more rapidly. From heat flows of the human body,

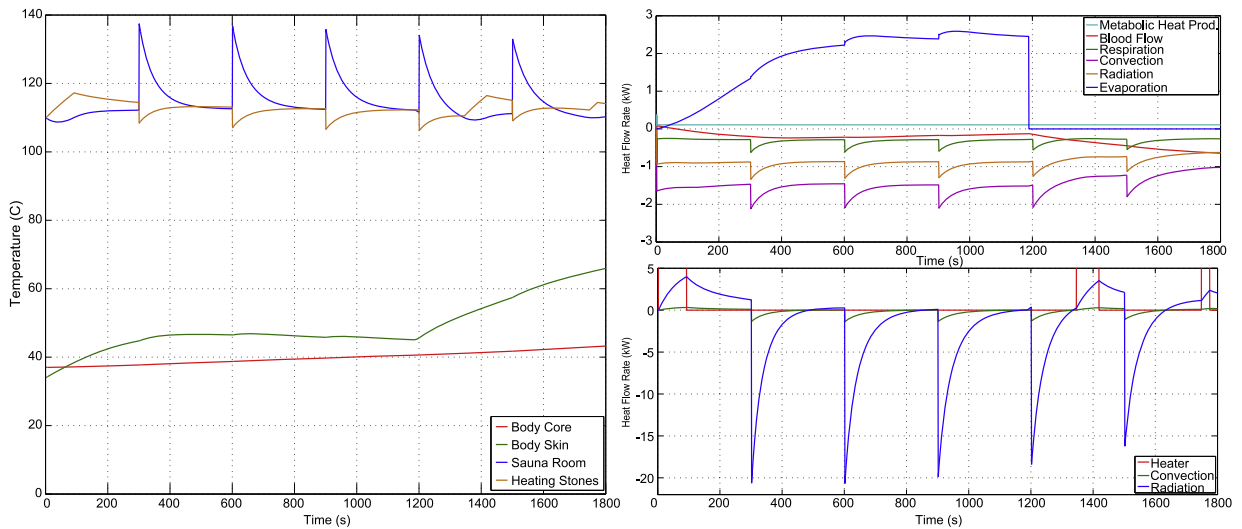


Fig. 14. Simulation results for the sauna with moderate water pouring.

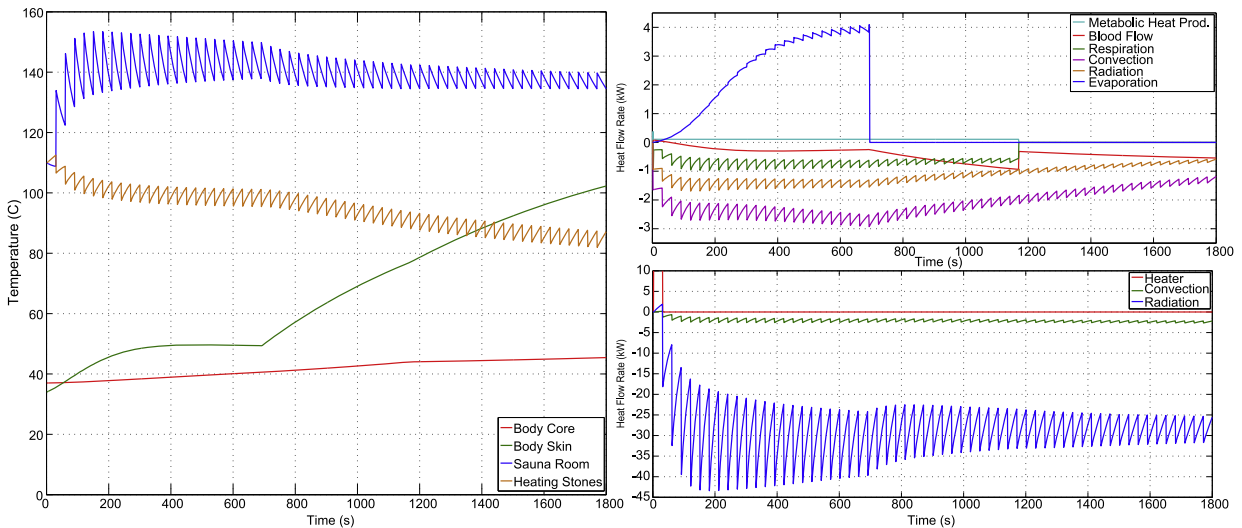


Fig. 15. Simulation results for the sauna with extreme water pouring.

we see that sweating stops much earlier. At some point between minute 16 and 20, metabolism and respiration stops: the person is in the death zone.

In our model of the human body, the *potentially-lethal* situation occurs when either the core experiences severe hyperthermia, the skin gets second degree burn, or the body is severely dehydrated. To find out how long a person can stay safe before encountering serious problems, we use HI-Maude's find earliest command:

```
(hfind earliest cs1 =>*
  {REST:Configuration
    < personCore : CoreHumanBody | coreState : CRST, bWaterState : BWST >
    < personSkin : SkinHumanBody | skinState : SKST >
    < coreSkinBlood : BloodFlow | entity1 : personCore, entity2 : personSkin >
    such that (CRST==sevHyperthermia or SKST==secondDBurn or BWST==sevDehydration)
    using euler stepsize 1.0 .)
```

We have also executed this analysis command using the Runge–Kutta 2nd order (*rk2*) and Runge–Kutta 4th order (*rk4*) numerical methods instead of the Euler (*euler*) method. The following table shows the results (and the CPU times in

parenthesis) of the analysis command above for different sauna types, persons, and numerical methods used in the analysis¹⁰:

Person	Num. method	Sauna		
		Dry	Moderate wet	Extreme wet
Normal	euler	1177 s (468 881 ms)	1104 s (448 652 ms)	719 s (313 138 ms)
	rk2	1176 s (874 476 ms)	1105 s (900 189 ms)	722 s (541 576 ms)
	rk4	1176 s (1 562 457 ms)	1105 s (1 331 092 ms)	722 s (788 942 ms)
Trained	euler	2436 s (1 230 723 ms)	2160 s (1 107 333 ms)	1266 s (626 899 ms)
	rk2	2436 s (3 211 172 ms)	2161 s (2 843 659 ms)	1271 s (1 510 340 ms)
	rk4	2436 s (6 347 407 ms)	2161 s (4 644 950 ms)	1271 s (2 264 514 ms)
Unhealthy	euler	780 s (282 293 ms)	597 s (217 920 ms)	340 s (131 585 ms)
	rk2	780 s (285 897 ms)	598 s (222 497 ms)	341 s (136 069 ms)
	rk4	780 s (590 649 ms)	598 s (400 154 ms)	341 s (203 391 ms)

If we are interested in only one body condition, e.g., to find out how long the person can stay in the sauna room before becoming severely hyperthermic, we can simplify the command above to:

```
(hfind earliest
  cs1 =>* {REST:Configuration <personCore:CoreHumanBody | coreState:sevHyperthermia>}
  using rk2 stepsize 1.0 .)
```

The following table shows the results of the analysis command above for different sauna environments and different persons (with the results obtained by using the `euler` method shown in parenthesis when it differs from that obtained by using the `rk2` and `rk4` methods)¹¹:

Person	Body condition	Environment (sauna)		
		Dry	Moderately humid	Extremely humid
Normal	sev. hyperthermia	1176 s (1177 s)	1105 s (1104 s)	722 s (719 s)
	sev. dehydration	no result	no result	no result
	skin 2nd burn	1365 s (1366 s)	1259 s (1258 s)	774 s (770 s)
Trained	sev. hyperthermia	2436 s	2161 s (2160 s)	1271 s (1266 s)
	sev. dehydration	no result	no result	no result
	skin 2nd burn	2792 s	2458 s (2457 s)	1397 s (1392 s)
Unhealthy	sev. hyperthermia	780 s	728 s	468 s (466 s)
	sev. dehydration	no result	no result	no result
	skin 2nd burn	784 s	598 s (597 s)	341 s (340 s)

Further analysis showed that persons typically die from severe hyperthermia before becoming severely dehydrated, explaining why states where the person is severely dehydrated are not reachable.

7.3. What caused the 2010 accident?

Our analyses above show that even the average person should endure 12 minutes in the wet sauna before the onset of major injuries. Therefore, something must have gone terribly wrong in 2010. Indeed, Timo Kaukonen told journalists before the final that day that the conditions seemed tougher than usual. We next propose and analyze some possible explanations for the still unsolved tragedy that could cause major injuries to a five-time world champion in around 6 minutes:

1. The initial sauna room temperature is, as expected, 110 °C. But the temperature of the heating rocks is way warmer. We understand from the manual of the heating system of the product used in Heinola that the temperature sensor only monitors the air temperature of the sauna room and not of the heating rocks. Our HI-Maude experiments with different temperatures indicate that the initial temperature of the rocks needed to be around 250 °C to explain the tragedy.
2. Maybe the most immediate suggestion is that the temperature sensor was malfunctioning and that the real temperature was higher. Our analysis shows that even at 150 °C, they should be fine for more than 10 minutes. A temperature of around a whopping 210 °C is needed to explain the tragic outcome.

¹⁰ As we can see in the table, the Runge–Kutta method of 2nd order seems to be a very good compromise between efficiency and accuracy, as it gives the same results as the more sophisticated Runge–Kutta method of 4th order, while having a computation time that is much closer to that of the simpler Euler method. This is consistent with our results in [22], where we compared the precision and execution times on a smaller case study for which the precise solution was available.

¹¹ The analyses yield the same result (in seconds) using the numerical methods `rk2` and `rk4`.

3. The humidity of the sauna is extremely high from the start. Our experiments show that we needed to start with 39 liters of water vapor instead of the expected 10 liters to explain the outcome.

The results of our HI-Maude analyses of these hypothesis are (computation CPU time in parenthesis):

Person	Body condition	Num. meth.	Possible causes		
			Heating rocks 250°C	Room air 210°C	Water vapor 39 liters
Trained	sev. hyperthermia	euler	349 s (137 300 ms)	328 s (124 485 ms)	785 s (344 444 ms)
		rk2	350 s (181 877 ms)	328 s (158 676 ms)	783 s (656 578 ms)
		rk4	350 s (272 818 ms)	329 s (243 532 ms)	783 s (986 408 ms)
	sev. dehydration	euler	no result	no result	no result
		rk2	no result	no result	no result
		rk4	no result	no result	no result
	skin 2nd burn	euler	414 s (228 247 ms)	393 s (209 384 ms)	311 s (302 428 ms)
		rk2	415 s (305 529 ms)	393 s (244 487 ms)	304 s (331 776 ms)
		rk4	415 s (382 219 ms)	394 s (337 212 ms)	304 s (416 362 ms)

7.4. Analyzing behavioral thermoregulation

Finally, in this section we deal with the behavioral/voluntary action part of the human thermoregulatory system. That is, we investigate whether a person will always reach his/her tolerance limit for cold/heat before the onset of major injuries.

As mentioned in Section 5, our model uses empirical expressions to predict thermal sensation. This model uses an 11-point index scale from -5 to $+5$, where -5 denotes intolerable cold, and $+5$ denotes intolerable heat. However, this model is used for modeling human comfort in a normal situation, whereas we want to model the mental state of a person who is willing to take a risk and endure the pain from an extreme surrounding environment for fame and glory. We therefore conjecture that we can apply the same thermal sensation prediction model by stretching the index value from 5.0 to 9.0. To analyze how long the person can stay in the sauna before it becomes unbearably hot w.r.t. the original and the stretched thermal sensation limit, we use the hybrid find earliest command to find out the earliest time the thermal sensation value exceeds the tolerance limit value:

```
(hfind earliest cs1 =>*
  {REST:Configuration <cortex:Cortex | tsensVal:TS, tsensTolLimit:TSL>} such that (TS >= TSL)
  using rk2 stepsize 1.0 .)
```

The results are presented in the following table¹²:

Person	Tolerance limit	Sauna		
		Dry	Moderately humid	Extremely humid
Normal	5	100 s	100 s	85 s
	9	1215 s	1130 s	300 s
Trained	5	110 s	110 s	90 s
	9	2645 s	2330 s	1330 s
Unhealthy	5	95 s	95 s	80 s
	9	320 s	315 s	225 s

If we combine these results with the results in Section 7.2, we can conclude with the good news that all kinds of persons will feel a discomfort (attribute `tsensVal`) that is greater than their pain threshold $+5$ (“intolerable heat”) before the onset of major injuries. For example, a normal person would feel the heat to be intolerable after 85 seconds in the extreme sauna, whereas we see in Section 7.2 that (s)he can safely stay in that sauna for around 12 minutes. However, mental and physical exercise that stretches the tolerance threshold from “intolerable heat” ($+5$) to a whopping $+9$ would be dangerous for both the normal person and the champion contestant, since the latter would only consider leaving the extreme sauna after 1330 seconds, whereas he might become severely hyperthermic already after 1266 seconds.

8. Related work

As seen in this paper, the human thermoregulatory system is very complex, and seems to be significantly beyond the scope of the (much less expressive) traditional formal models for hybrid systems, such as hybrid automata [35] and hybrid

¹² We only show the results for the numerical methods `rk2` and `rk4` (which coincide); the results for `euler` are very similar, with only two minor deviations.

Petri nets [36]. It should therefore not come as a surprise that we have not seen any other work that uses formal methods and tools to model and analyze the human thermoregulatory system.

Among informal/simulation models that model the human thermoregulatory system reasonably close to our level of abstraction, it is worth mentioning the Human Comfort Module of the commercial ThermoAnalytics system.¹³ This is an advanced simulator for analyzing human thermal comfort within complex environments indoors, outdoors, and in transportation systems. It takes into account radiant, convective, and conductive heat transfer, including localized thermoregulatory responses, such as perspiration, respiration, activity level, and blood flow changes. The Human Comfort Module provides the tools to design and optimize heating/cooling systems or improve clothing and uniform design. Likewise, the UC Berkeley Multinode Comfort Model models human thermoregulation, including explicitly considering multiple body layers (core, muscle, fat, and skin tissues) and a clothing layer, physiological mechanisms such as vasodilation, vasoconstriction, sweating, and metabolic heat production, as well as interactions such as convection, conduction and radiation between the body and the environment [37]. However, these tools do not support formal modeling, and have not been used to analyze the human in oppressive saunas.

Other work on computer analysis of aspects of the human thermoregulatory system includes: the work reported in [38] on analyzing the efficiency of space suites, although nothing is said about what language was used (“...[models] adapted for a personal computer”); a FORTRAN implementation of a model of the human thermoregulatory system used to analyze behavior during physical exercise [39]; and a simulation model using the Euler method, and focusing on heat-mass transfers in space flight conditions [40], but where only a number of differential equations are given, whereas not even the language in which the simulations are done is given. These models all use different models of the human thermoregulatory system: [38] considers the 225-node Wissler model and the 41-node Metabolic Man model, whereas [40] divides the body into 17 segments. The paper [41] describes some MATLAB simulations of the 41-node model, but only some parameters are presented, and not much of the model itself. We are not aware of any work using popular frameworks such as Simulink or KeYmeara to model the human thermoregulatory system.

Among our own previous work, our paper [7] introduces the formal effort/flow-based object-oriented modeling technique for continuously interacting hybrid systems; the papers [42,22] describe how the Euler and the Runge–Kutta numerical approximation methods have been adapted to the effort/flow setting and then formalized/implemented in Maude. The papers [4,5] introduce the HI-Maude tool, and, finally, the paper [18] provides a shorter exposition of the sauna case study. The precise relationship between [18] and the current paper is given in the introduction.

Finally, we refer to [4] for a discussion about formal models and tools related to HI-Maude.

9. Concluding remarks

We have applied the recently developed HI-Maude formal modeling methodology for interacting hybrid systems to formally model the human thermoregulatory system and the sauna used in the Sauna World Championships. This is an extremely demanding case study, where the continuous dynamics of the entire system is a combination of the continuous dynamics of many parts, each of which can be described by differential equations. This task was made feasible by the modeling methodology’s support for decomposing the definition of the complex continuous behaviors of the entire system into the much simpler tasks of defining the continuous behavior of *single* physical entities (such as the body core, the skin, the sauna room, the heating rocks, etc.) and the physical interactions between these entities. In particular, HI-Maude provides an object-oriented modeling methodology for complex hybrid systems, where the continuous dynamics can be specified at the class level. This contrasts with most, if not all, formal models for hybrid systems, where the user must explicitly specify the continuous dynamics of the *entire* system; this definition of the continuous dynamics must then be redefined for each new configuration of objects in the state. Even if such a complex model could be developed for the human thermoregulatory system, that effort would have to be significantly redone for different contexts, such as when the human is in the extreme sauna. In contrast, our model could be “connected” to the sauna model without any modifications.

We have presented a detailed and realistic formal model, with all assumptions based on scientific/medical knowledge, of the human thermoregulatory system and its interactions with different kinds of environments, including oppressive saunas. Using the HI-Maude simulation and model checking tool, we have simulated and further formally analyzed the reaction of different kinds of people to various saunas. The results of our analyses are fairly close to what we know about how long both professionals and amateurs (sports writers, rock stars, etc.) can endure in the sauna [34,33,43].

In particular, we have used our detailed models and the HI-Maude tool to propose some possible explanations for the still unresolved tragedy at the 2010 Sauna World Championships. We have also included a model of sensing thermal discomfort and our analyses show that a person will feel “intolerable discomfort” before major injuries in the sauna should appear.

It would also be interesting to use our model to analyze how long a person can endure extreme cold before significant problems occur. Our model of the human thermoregulatory system should be equally suitable for this task without further changes (apart from adding a cold environment).

The results of HI-Maude analyses must, however, be seen in light of the fact that simulation and model checking is done w.r.t. the numerical methods for approximating the continuous behaviors. Since our methods are not based on, say,

¹³ <http://www.thermoanalytics.com>.

over-approximating the reachable state space, but instead on explicit-state execution of transitions, our formal analyses cannot *guarantee* the corresponding properties. That is, temporal logic “model checking” and reachability analysis in HI-Maude could give the wrong answers: both false positives and false negatives could be reported by the tool. Therefore, to have any confidence in the correctness of the model checking analyses, we must in future work identify classes of models and properties for which we can guarantee the correctness of HI-Maude model checking. The point is that the present paper has indicated that HI-Maude indeed seems to support a useful modeling methodology for very complex hybrid systems, which means that developing sound model checking methods for HI-Maude is a worthwhile task. That said, since the enabledness of our rewrite rules never depends on the *precise* value of a continuous parameter (for example, the rule `normal-to-modhyperthermia` cannot only be applied when the body temperature is exactly 38.9°, but can be applied when the body temperature is greater than or equal to 38.9°), we *intuitively* cover “all possible behaviors” of the system, albeit with slight imprecisions due to the use of numerical approximations.

Acknowledgements

We would like to thank the reviewers for very helpful comments on earlier versions of this paper. We gratefully acknowledge financial support for this work by The Research Council of Norway through the Rhythm project and by The Research Council of Norway and the German Academic Exchange Service through the DAAD ppp project HySmart.

References

- [1] Vikings tackle Stringer dies from heatstroke, <http://static.espn.go.com/nfl/news/2001/0731/1233494.html>, 2001.
- [2] M. Winkeljohn, Heat injuries forcing new technologies, <http://sports.espn.go.com/ncaa/recruiting/football/news/story?id=4372652>, 2010, accessed February 2, 2012.
- [3] Medical experiments of the holocaust and Nazi medicine, <http://remember.org/educate/medexp.html>, accessed January 22, 2013.
- [4] M. Fadlisyah, P.C. Ölveczky, E. Ábrahám, Object-oriented formal modeling and analysis of interacting hybrid systems in HI-Maude, in: Proc. SEFM'11, in: Lecture Notes in Computer Science, vol. 7041, Springer, 2011.
- [5] M. Fadlisyah, P.C. Ölveczky, The HI-Maude tool, in: Proc. CALCO'13, in: Lecture Notes in Computer Science, vol. 8089, Springer, 2013, pp. 322–327.
- [6] P.C. Ölveczky, J. Meseguer, The Real-Time Maude tool, in: C.R. Ramakrishnan, J. Rehof (Eds.), Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08), in: Lecture Notes in Computer Science, vol. 4963, Springer, 2008, pp. 332–336.
- [7] M. Fadlisyah, E. Ábrahám, D. Lepri, P.C. Ölveczky, A rewriting-logic-based technique for modeling thermal systems, in: Proc. RTRTS'10, in: Electronic Proceedings in Theoretical Computer Science, vol. 36, 2010.
- [8] C.-L. Hwang, S.A. Konz, Engineering models of the human thermoregulatory system—a review, IEEE Trans. Biomed. Eng. BME-24 (4) (1977) 309–325, <http://dx.doi.org/10.1109/TBME.1977.326137>.
- [9] A. Gagge, J. Stolwijk, Y. Nishi, An effective temperature scale based on a simple model of human physiological regulatory response, ASHRAE Trans. 77 (1) (1971) 247–262.
- [10] K. Parsons, Human Thermal Environments: The Effects of Hot, Moderate, and Cold Environments on Human Health, Comfort and Performance, Taylor & Francis, London, New York, 2003.
- [11] ASHRAE Handbook: Fundamentals, SI edition, American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., 2005.
- [12] M. McPherson, Subsurface Ventilation and Environmental Engineering, Chapman & Hall, 1993.
- [13] H. Hensel, Thermoreception and Temperature Regulation, Monogr. Physiol. Soc., vol. 38, Academic Press, 1981.
- [14] C. Tipton, ACSM's Advanced Exercise Physiology, Lippincott Williams & Wilkins, 2006, <http://books.google.no/books?id=YAAT1-hebMgC>.
- [15] L.A. Bloomfield, How Things Work: The Physics of Everyday Life, Wiley, Hoboken, NJ., 2006.
- [16] C. Plantadosi, The Biology of Human Survival: Life and Death in Extreme Environments, Oxford University Press, 2003.
- [17] The Engineering ToolBox, <http://www.engineeringtoolbox.com/>.
- [18] M. Fadlisyah, P.C. Ölveczky, E. Ábrahám, Formal modeling and analysis of human body exposure to extreme heat in HI-Maude, in: Proc. WRLA'12, in: Lecture Notes in Computer Science, vol. 7571, Springer, 2012, pp. 139–161.
- [19] P.E. Wellstead, Introduction to Physical System Modelling, Academic Press, 1979.
- [20] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C. Talcott, All About Maude – A High-Performance Logical Framework, Lecture Notes in Computer Science, vol. 4350, Springer, 2007.
- [21] P.C. Ölveczky, J. Meseguer, Semantics and pragmatics of Real-Time Maude, High.-Order Symb. Comput. 20 (1–2) (2007) 161–196.
- [22] M. Fadlisyah, P.C. Ölveczky, E. Ábrahám, Formal modeling and analysis of hybrid systems in rewriting logic using higher order numerical methods and discrete-event detection, in: F. Arbab, H. Mirian (Eds.), Proc. CSSE'11, IEEE, 2011, pp. 1–8.
- [23] J. Meseguer, Membership algebra as a logical framework for equational specification, in: F. Parisi-Presicce (Ed.), Proc. WADT'97, in: Lecture Notes in Computer Science, vol. 1376, Springer, 1998, pp. 18–61.
- [24] J. Meseguer, Conditional rewriting logic as a unified model of concurrency, Theor. Comput. Sci. 96 (1992) 73–155.
- [25] R. Bruni, J. Meseguer, Semantic foundations for generalized rewrite theories, Theor. Comput. Sci. 360 (1–3) (2006) 386–414.
- [26] P. Viry, Equational rules for rewriting logic, Theor. Comput. Sci. 285 (2002) 487–517.
- [27] J. Meseguer, A logical theory of concurrent objects and its realization in the Maude language, in: G. Agha, P. Wegner, A. Yonezawa (Eds.), Research Directions in Concurrent Object-Oriented Programming, MIT Press, 1993, pp. 314–390.
- [28] T. Henzinger, P. Kopke, A. Puri, P. Varaiya, What's decidable about hybrid automata?, in: Journal of Computer and System Sciences, ACM Press, 1995, pp. 373–382.
- [29] M. Fadlisyah, E. Ábrahám, P.C. Ölveczky, Formal modeling and analysis of extreme heat exposure to the human body in HI-Maude – a case study inspired by the tragedy at 2010 Sauna World Championship, Technical report available at <http://folk.uio.no/mohamf/HI-Maude>, 2012.
- [30] I. Herman, Physics of the Human Body, Springer, 2007.
- [31] M. Owen, R. Parsons, 2005 Ashrae Handbook, SI edition, American Society of Heating, 2005.
- [32] Swedish Standard Institute, Ergonomics of the thermal environment – determination and interpretation of cold stress when using required clothing insulation (IREQ) and local cooling effects, 2007 (ISO 11079:2007).
- [33] R. Reilly, Sports from Hell: My Search for the World's Dumbest Competition, Doubleday/ESPN Books, 2010.
- [34] R. Reilly, The point of no return, ESPN.com, <http://sports.espn.go.com/espn/news/story?id=5451867>, 2010, accessed November 1, 2013.

- [35] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine, The algorithmic analysis of hybrid systems, *Theor. Comput. Sci.* 138 (1995) 3–34.
- [36] R. David, H. Alla, On hybrid Petri nets, *Discrete Event Dyn. Syst.* 11 (1–2) (2001) 9–40.
- [37] C. Huizenga, Z. Hui, E. Arens, A model of human physiology and comfort for assessing complex thermal environments, *Build. Environ.* 36 (6) (2001) 691–699, [http://dx.doi.org/10.1016/S0360-1323\(00\)00061-5](http://dx.doi.org/10.1016/S0360-1323(00)00061-5), <http://www.sciencedirect.com/science/article/pii/S0360132300000615>.
- [38] V.L. Pisacane, L.H. Kuznetz, J.S. Logan, J.B. Clark, E.H. Wissler, Use of thermoregulatory models to enhance space shuttle and space station operations and review of human thermoregulatory control, Tech. rep. 20070022493, NASA Johnson Space Center, 2007, <http://ntrs.nasa.gov/search.jsp?R=20070022493>.
- [39] P.A. Hancock, Predictive validity of a computer model of body temperature during exercise, *Med. Sci. Sports Exerc.* 13 (1) (1981) 31–33.
- [40] E.A. Kurmazenko, T.V. Matjushev, N.V. Soloshenko, I.V. Dokunin, A detailed simulation model of the human organism thermoregulation system, in: *Proc. Sixth European Symposium on Space Environmental Control Systems*, European Space Agency, 1997, pp. 815–821, SP-400.
- [41] T. Miller, D. Nelson, G. Bue, L. Kuznetz, Dynamic simulation of human thermoregulation and heat transfer for spaceflight applications, in: *Proc. 41st International Conference on Environmental Systems*, vol. 2, American Institute of Aeronautics and Astronautics, 2011, pp. 1076–1089.
- [42] M. Fadlisyah, P.C. Ölveczky, E. Ábrahám, Adaptive-step-size numerical methods in rewriting-logic-based formal analysis of interacting hybrid systems, *Electron. Notes Theor. Comput. Sci.* 274 (2011) 17–32.
- [43] Sauna world championships, http://en.wikipedia.org/wiki/World_Sauna_Championships, accessed November 1, 2013.