SCIENCE CHINA

Month Year Vol. Vol No. No: 1–16
doi:

An Evaluation Framework for Energy Aware Buildings using Statistical Model Checking

DAVID Alexandre[1*], DU DeHui[2†], LARSEN Kim G.[1‡], MIKUČIONIS Marius[1§] & SKOU Arne[1¶]

[1]*CISS, Department of Computer Science, Aalborg University, Aalborg, Denmark,*
[2]*Shanghai Key Laboratory of Trustworthy Computing East China Normal University, Shanghai* 20062*, China,*

Received June 1, 2012; accepted ????

Abstract Cyber-physical systems are to be found in numerous applications throughout society. The principal barrier to develop trustworthy cyber-physical systems is the lack of expressive modelling and specification formalisms supported by efficient tools and methodologies. To overcome this barrier, we extend in this paper the modelling formalism of the tool UPPAAL-SMC to *stochastic hybrid automata*, thus providing the expressive power required for modeling complex cyber-physical systems. The application of Statistical Model Checking provides a highly scalable technique for analyzing performance properties of this formalisms.
A particular kind of cyber-physical systems are Smart Grids which together with Intelligent, Energy Aware Buildings will play a major role in achieving an energy efficient society of the future. In this paper we present a framework in UPPAAL-SMC for energy aware buildings allowing to evaluate the performance of proposed control strategies in terms of their induced comfort and energy profiles under varying environmental settings (e.g. weather, user behaviour, ...). To demonstrate the intended use and usefulness of our framework, we present an application to the Hybrid Systems Verification Benchmark of [10].

Keywords Timed Automata, Stochastic Hybrid Systems, Model Checking, Statistical Model Checking, Performance Evaluation, Energy Consumption, Cyber Physical Systems, Smart Grid, Energy Aware Buildings, Hybrid System Verification Benchmark

Citation DDLM12 An Evaluation Framework for Energy Aware Buildings using Statistical Model Checking.

*adavid@cs.aau.dk
†dehuidu@cs.aau.dk
‡kgl@cs.aau.dk
§marius@cs.aau.dk
¶ask@cs.aau.dk

1 Introduction

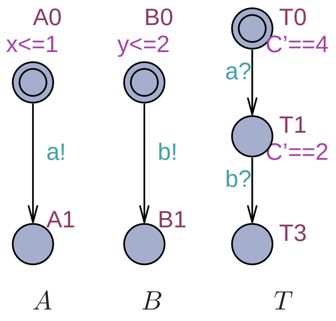
Trustworthy Cyber-Physical Systems The term cyber-physical systems [13] refers to the tight conjoining of and coordination between computational and physical entities. Cyber-physical systems are large-scale distributed systems, often viewed as networked embedded systems, where a large number of computational components are deployed in a physical environment. Each component collects information about and offers services to its environment (e.g., environmental monitoring and control, health-care monitoring and traffic control). This information is processed either at the component, in the network or at a remote location (e.g., the base station), or in any combination of these.

Cyber-physical systems are to be found in numerous applications throughout the society, ranging from automated highway systems (AHS) [9], air traffic control systems [16], personal and medical devices [11], Smart Grids and energy aware buildings to mention a few. Being as such omnipresent, it is utmost important that these systems are trustworthy, in terms of adaptability, autonomy, efficiency, functionality, reliability, safety, security and usability. However, there are a number of challenges that have to be addressed before such a vision can be realized. The principal barrier to develop trustworthy cyber-physical systems is the lack of modelling and specification formalisms with supporting tools and methodologies that comprehend cyber and physical resources in a single unified framework.

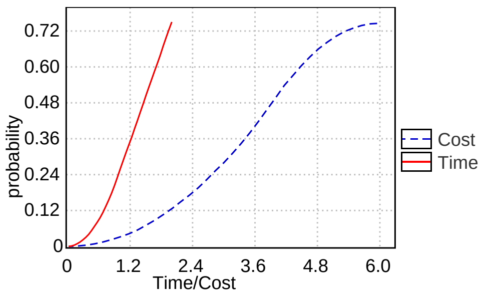
A characteristic of cyber-physical systems is that they have to meet a multitude of quantitative constraints, e.g., timing constraints, power consumption, memory usage, communication bandwidth, QoS, and often under uncertainty of the behaviour of the environment. Existing model-driven methodologies for embedded systems are rather sophisticated in handling functional requirements, and some methods are good at handling special kinds of quantitative constraints; however, there is a lack of a mathematical foundation and supporting tools allowing to handle the combinations of quantitative aspects concerning, such as time, stochastic behaviour, energy consumption.

Stochastic Hybrid Systems and Statistical Model Checking In our previous work [7, 8] we have proposed the formalism of Priced Timed Automata (PTA), that are extensions of timed automata [2], where clocks may have different rates (even potentially negative) in different locations. PTAs are as expressive as linear hybrid automata [1] providing high expressive power useful for modelling complex cyber-physical systems, but also rendering most problems either undecidable or too complex to be solved with classical model checking approaches. To overcome these limitations, we proposed in [7] to apply Statistical Model Checking (SMC) techniques [14, 18, 15, 12], which is a highly scalable simulation-based approach. Of course, in contrast to an exhaustive model-checking approach, a simulation-based solution does not guarantee a correct result with 100% confidence. However, it is possible to bound the probability of making an error by increasing the simulation effort. Simulation-based methods are known to be far less memory and time intensive than exhaustive ones, and are sometimes the only option [3]. Also, several interesting properties – including performance properties – that cannot be expressed in classical temporal logic may be analyzed. SMC consists in randomly generating and monitoring simulations runs of the system and verify whether they satisfy a given property. The results are then used by statistical algorithms in order to compute among others an estimate of the probability for the system to satisfy the property. Such an estimate is correct up to some confidence that can be parameterized by the user. Several SMC algorithms that exploit a stochastic semantics for PTAs have recently been implemented in Uppaal-smc [7, 8, 5], including a distributed implementation demonstrating linear speed-up, thus providing additional scalability.

In the present paper, we significantly extend the modelling formalisms of Uppaal-smc to *stochastic hybrid automata* (SHA), which are (stochastic) timed automata whose clock rates can be changed into be constants or expressions depending on other clocks, effectively defining ODEs. As we shall see this extension enables the elegant modelling and efficient analysis of cyber-physical systems with complex stochastic and continuous behaviour of the environment.



(a) An NSHA, $(A|B|T)$.



(b) Cumulative probabilities for `time` and `Cost`-bounded reachability of T_3.

Figure 1: A simple NSHA illustrating races.

Energy Aware Buildings A particular kind of cyber physical systems are Smart Grids which together with intelligent, energy aware buildings will play a major role in the society in the future by balancing the growing amount of distributed renewable energy plants towards the varying amount of consumer demands. Major world governments led by EU, China and U.S. have defined plans for the further developments of Smart Grids until 2020 [4, 6, 17]. In these plans, energy aware buildings are supposed to plan their energy consumption/production on a 24 hour basis based on forecasts for weather, prices and desired comfort level. The challenge of developing such plans is addressed in a number of international and national projects, and the authors are currently collaborating with the Danish national defense building administration in a pilot study on providing an estimate of the potential energy savings through intelligent forecast and control of energy.

Contribution and Outline As a contribution within the general area of energy aware buildings and the specific pilot study, we offer in this paper a framework in UPPAAL-SMC for modelling and analyzing energy aware buildings. In Section 2 we first present the tool UPPAAL-SMC and its modelling and validation features. In Section 3 we present the energy aware building framework, which consists of several parameterized components (rooms, building, heaters, weather, user, . . .) as well as a collection of properties for evaluating comfort and energy profiles of various control strategies. To demonstrate the intended use and usefulness of our framework, we present in Section 4 an application to the Hybrid Systems Verification Benchmark of [10] addressing the control of the temperature of rooms in a given building. In addition to the original challenge we offer evaluation of performance properties related to comfort and energy consumption in more complex environmental settings. We also demonstrate the scalability of the performance analysis offered by UPPAAL-SMC. Finally we offer a conclusion including future work in Section 5.

2 Statistical Model Checking

UPPAAL-SMC supports the analysis of stochastic hybrid automata (SHA) that are timed automata whose clock rates can be changed to be constants or expressions depending on other clocks, effectively defining ODEs. This generalizes the model used in our previous work [8, 7] where only linear priced automata were handled. Our new release UPPAAL-SMC 4.1.10[1] supports fully hybrid automata with ODEs and a few built-in complex functions (such as $sin, cos, log, exp, sqrt$)!

We assume that SHA are input-enabled, deterministic (with a probability measure defined on the sets of successors), and non-zeno. SHA communicate via broadcast channels and shared variables to generate Networks of SHA (NSHA).

Fig. 1(a) provides an NSHA with three components A, B, and T as specified using the UPPAAL GUI. One can easily see that the composite system $(A|B|T)$ has the transition sequence:

[1] `www.uppaal.org`.

$\big(A_0B_0T_0,[x=0,y=0,C=0]\big) \xrightarrow{1}\xrightarrow{a!} \big(A_1B_0T_1,[x=1,y=1,C=4]\big) \xrightarrow{1}\xrightarrow{b!} \big(A_1B_1T_3,[x=2,y=2,C=6]\big)$, demonstrating that the final location T_3 of T is reachable. As shown in Fig. 1(b), location T_3 is reachable within cost 0 to 6 and within total time 0 and 2 in $(A|B|T)$ depending on when (and in which order) A and B choose to perform the output actions $a!$ and $b!$. Assuming that the choice of these time-delays is governed by probability distributions, a measure on sets of runs of NSHA is induced, according to which quantitative properties such as *"the probability of T_3 being reached within a total cost-bound of 4.3"* become well-defined.

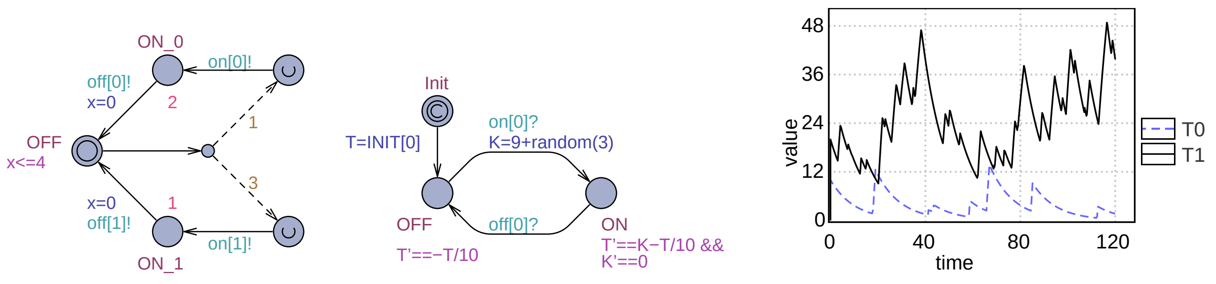
In our early work [7], we provide a natural stochastic semantics, where SHA components associate probability distributions to both the time-delays spent in a given state as well as to the transition between states. In Uppaal-smc uniform distributions are applied for bounded delays and exponential distributions for the case where a component can remain indefinitely in a state. In an NSHA the components repeatedly race against each other, i.e. they independently and stochastically decide on their own how much to delay before outputting, with the "winner" being the component that chooses the minimum delay. For instance, in the NSHA of Fig. 1(a), A wins the initial race over B with probability 0.75.

In addition, ODEs are solved by an internal (invisible) automaton that naturally races against all the other automata with a small discrete time step. Every time this internal automata wins, it forces the re-computation of all clock rates, achieving a linear piece-wise approximation of the actual function given by the ODEs.

As observed in [7], though the stochastic semantic of each individual SHA in Uppaal-smc is rather simple (but quite realistic), arbitrarily complex stochastic behavior can be obtained by their composition when mixing individual distributions through message passing. The beauty of our model is that these distributions are naturally and automatically defined by the NSHA.

A Simple 2-Room Example To illustrate the various aspects of the (extended) modeling formalism supported by Uppaal-smc, we consider the case of two independent rooms that can be heated by a single heater shared by the two rooms, i.e., at most one room can be heated at a time. Figure 2(a) shows the automaton for the heater. It turns itself on with a uniform distribution over time in-between $[0, 4]$ time units. With probability 1/4 room 0 is chosen and with probability 3/4 room 1. The heater stays on for some time given by an exponential distribution (rate 2 for room 0, rate 1 for the room 1). In summary, one may say that the controller is more eager to initiate the heating of room 1 than room 0, as well as less eager to stop heating room 1. The rooms are similar and are modelled by the same template instantiated twice. Figure 2(b) shows room 0. The room is initialized to its initial temperature and then depending on whether the heater is turned on or not, the evolution of the temperature is given by $T' = -T/10$ or $T' = K - T/10$. Furthermore, when the heater is turned on, its heating is not exact and is picked with a uniform distribution of $K \in [9, 12]$, realized by the update `K=9+random(3)`.

This example illustrates the support for NSHA in Uppaal-smc with extended arithmetic on clocks and generalized clock rates.



(a) Stochastic heater. (b) Room 0. (c) Room temperatures.

Figure 2: A simple two room example.

Extended Input Language Uppaal-smc takes as input NSHA as described above. Additionally, there is support for other features of the Uppaal model checker's input language such as integer variables, data structures and user-defined functions, which greatly ease modelling. Uppaal-smc allows the user to specify arbitrary rates for the clocks, which includes a mix of integer and clock expressions on any location. In addition, the automata support branching edges where weights can be added to give a distribution on discrete transitions. It is important to note that rates and weights may be general expressions that depend on the states but not just simple constants.

Checking Queries The fundamental principle in Uppaal-smc is to generate runs and evaluate some expression on the states along the obtained run. Runs are always *bounded*, either by time, by a number of steps, or more generally by cost (when using a clock explicitly). The engine has a built-in detection of Zeno behaviours to stop the generation of such runs. Examples of the syntax for the different types of bounds are `[<=100]` for 100 time units since the beginning of the run, `[#<=50]` for 50 discrete transitions taken from the initial state, and `[x<=200]` until the clock `x` reaches 200[2].

Uppaal-smc supports simulations with monitoring custom expressions, probability evaluation, hypothesis testing, and probability comparison. We can simulate and plot the temperatures with the query

```
simulate 1 [<=120]{Room(0).T,Room(1).T}
```

The query asks the checker to simulate one run over 120 time units and plot the temperatures of `Room(0)` and `Room(1)`. The heater in this example is purely stochastic and does not meet any particular requirement. Still, the simulation obtained from this query in Figure 2(c) shows that the heater is able to maintain the temperatures within (mostly) distinct intervals.

We can evaluate on a shorter time scale the probability for the temperature of `Room(0)` to stay below 15 and the temperature of `Room(1)` to stay above 10 with the queries

```
Pr[<=100]([] Room(0).Init || Room(0).T <= 15)
Pr[<=100]([] Room(1).Init || Room(1).T >= 10)
```

The results are respectively in $[0.32, 0.43]$ and $[0.38, 0.49]$. The precision and confidence of these so-called confidence intervals are user-defined and influence the number of runs needed to compute the probability. In this example, to set the precision to ± 0.05 with a confidence of 95%, we need 738 runs. In fact if we are only interested in knowing if the second probability is above a threshold it may be more efficient to test the hypothesis

```
Pr[<=100]([] Room(1).Init || Room(1).T >= 10) >= 0.42
```

which is accepted in our case with 1109 runs for a level of significance of 95%. To obtain an answer at comparable level of precision with probability evaluation, we would need to use a precision of ± 0.005, which would require 73778 runs instead.

The tool can also compare probabilities without needing to compute them individually. We can test the hypothesis that the heater is better at keeping the temperature of `Room(1)` above 10 than keeping the temperature of `Room(0)` below 15:

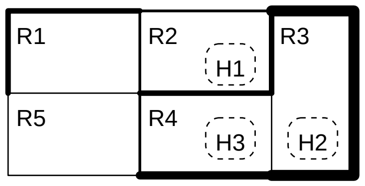
```
Pr[<=100]([] Room(1).Init || Room(1).T >= 10) >=
Pr[<=100]([] Room(0).Init || Room(0).T <= 15)
```

which is accepted in this case with 95% level of significance with 1432 runs.

3 Framework for Energy Aware Buildings

This section describes our framework for modeling and evaluating control systems for energy aware buildings. The framework consists of a number of templates and queries which can be instantiated

[2]It is up to the modeler to ensure that the clock eventually reaches the bound.



(a) Rooms R_i with heaters H_k.

$$\begin{pmatrix} 0.00 & 0.40 & 0.00 & 0.00 & 0.50 \\ 0.40 & 0.00 & 0.20 & 0.20 & 0.00 \\ 0.00 & 0.20 & 0.00 & 0.50 & 0.00 \\ 0.00 & 0.20 & 0.50 & 0.00 & 0.40 \\ 0.50 & 0.00 & 0.00 & 0.40 & 0.00 \end{pmatrix}$$

(b) Adjacency matrix a.

$$\begin{pmatrix} 0.25 & 0.35 & 0.10 & 0.15 & 0.45 \end{pmatrix}$$

(c) Environment vector b.

$$\begin{pmatrix} 10.0 & 7.0 & 10.0 & 11.0 & 9.0 \end{pmatrix}$$

(d) Heating vector c.

Figure 3: Layout of a building and its representation.

and composed to reflect a particular building configuration with customizable control strategies. The templates are parameterized hybrid stochastic automata accepted by UPPAAL and the queries address probabilistic properties over time, comfort and energy counters, providing insights about the comfort and energy consumption. The framework is our set of templates and analysis methodology describing how to model and analyze the building models using UPPAAL-SMC. In this paper we explore a number of configurations using different templates and parameter settings. There are three parameter-dimensions:

- Environment temperatures: We use different patterns to model fixed values, abstract oscilation covering ranges of values, and a simple daily weather.
- Strategies for secondary controllers: We use three different strategies of switching over the heating between the rooms.
- User temperature profiles: We use different thresholds of temperatures to define user preferences that are either static or presence-based.

We examine $3 \times 3 \times 2 = 18$ combinations in total and draw some conclusions based on data provided by the tool. First, we describe our assumptions about the building setup and then proceed how this setup is modeled and analyzed using integrated UPPAAL toolkit.

3.1 Heated Building Setup

As a starting point, consider a setup proposed by [10] as a benchmark challenge for hybrid systems model-checkers. The benchmark consists of a building layout with temperature dynamics, autonomous heaters and a central controller deciding which room gets a heater. The room temperature dynamics is described by a differential equation:

$$T_i' = \sum_{j \neq i} a_{i,j}(T_j - T_i) + b_i(u - T_i) + c_i h_i$$

where T_i and T_j are the temperatures in room i and j respectively, u is the environment temperature and h_i is equal to 1 when the heater is turned on in the room i and 0 otherwise. The building layout is encoded by an adjacency matrix a where $a_{i,j}$ is a heat exchange coefficient between rooms i and j. The heat exchange with an environment is encoded in a separate vector b, where b_i is a energy loss coefficient for room i. An energy supply from a heater is encoded in a vector c, where c_i is a power coefficient for room i. Figure 3 shows a benchmark building configuration instance (`HEAT15` in [10]) with rooms and heaters, where the wall thickness corresponds to an isolation defined by a and b. Each heater is equipped with a bang-bang controller configured to turn on the heating ($h_i := 1$) when the temperature T_i is below threshold on_i

room	1	2	3	4	5
off	21	21	21	21	21
on	19	19	19	19	19
get	16	17	18	17	16
low	15	16	16	16	15
dif	1.0	1.0	1.0	1.0	1.0
imp	1	30	2	3	4
pow	5	5	5	5	5

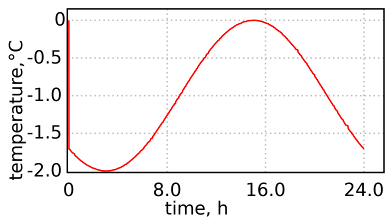
Table 1: Temperature thresholds.

and turn off ($h_i := 0$) when the temperature T_i is greater than off_i. Whenever the heating is turned on, the heaters consume the amount of power denoted by vector *pow*. The central controller can switch-over the heating from one room to another. The room is said to be needing a heater if the temperature drops below its *get* threshold and it is said to be outside comfort zone if the temperature drops below *low*. Table 1 shows a list of temperature thresholds for each room. Table 2 describes three strategy variations of when the heating can be switched over. Strategy 1 is the original one from [10], which is based on heuristics that the temperature difference between rooms should not be too high. Strategy 2 is based on assumption that one should not take heating away from a room which also needs heating. Strategy 3 tells that the heating can be taken away if the heater may potentially be turned off without disturbing the local control objective. To reduce the non-determinism further, we consider probabilistic choices between the possible room destinations denoted by probabilistic weights *imp*.

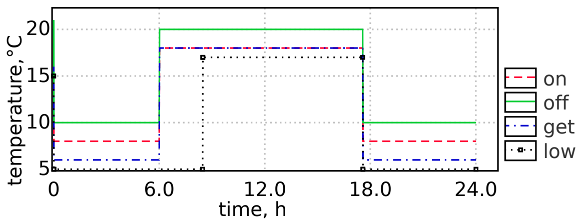
Table 2: Strategies for switching-over the heating from room j to room i.

Strategy 1	Strategy 2	Strategy 3
room i has no heater	room i has no heater	room i has no heater
room j has a heater	room j has a heater	room j has a heater
temperature $T_i \leqslant get_i$	temperature $T_i \leqslant get_i$	temperature $T_i \leqslant get_i$
difference $T_j - T_i \geqslant dif_i$	threshold $T_j \geqslant get_j$	threshold $T_j \geqslant on_j$

Further we propose to augment this setup with specific weather conditions and a user profile to make a more realistic case and consider more options for optimizing the energy consumption. The benchmark [10] assumes that the environment temperature is within a range between $0°C$ and $-2°C$ without any specific dynamics. We consider three cases of temperature curves: constantly at $-2°C$ (worst case scenario), rapidly changing between $0°C$ and $-2°C$ and slowly changing between $0°C$ and $-2°C$ (as in a night and day cycle). Figure 4(a) shows one daily cycle.



(a) Outside temperature.



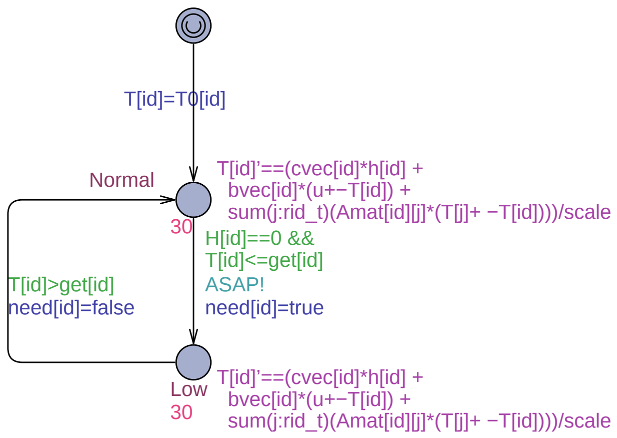
(b) User profile: preferred temperature thresholds.

Figure 4: Temperature dynamics over one day and night cycle.

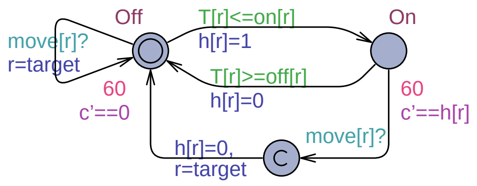
3.2 Modeling in Uppaal-smc

The goal is to provide a flexible and scalable modeling framework, which is achieved through modular composition of separate SHA processes. The complete model consists of the following SHA processes composed in parallel: weather environment, rooms, heaters, central controller and a user profile for each room. Each room has its own identifier from room-identifier type domain `rid_t` which is an integer ranging from 1 to the number of total rooms in the building. Similarly the heaters are identified by `hid_t` type. Then the building layout (room adjacency matrix) is encoded by two dimensional array over `rid_t`×`rid_t`. The dynamical coefficients, threshold vectors and temperatures are represented by arrays of integers and clocks over `rid_t` domain. Our model assumes that the model time units are in hours and the coefficients are specified as scaled integers with $scale == 100$.

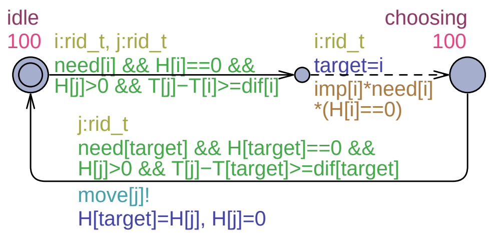
SHA model of a room with an identifier *id* is shown in Fig. 5(a). The first (urgent) transition sets the temperature to the initial value ($T0[id]$) and moves to `Normal` location where the temperature changes according to the dynamics specified as invariant ($cvec[id]$ amount of contribution from the heater when $h[id] == 1$, $bvec[id]$ is a contribution from an environment with temperature u and the rest is from adjacent rooms). The room may move to location `Low` and set the variable $need[id]$ to true showing that it needs a heater if the temperature drops below $get[id]$ threshold. Similarly the room may return from `Low` to `Normal` if the temperature becomes higher than the threshold. The exponential rate of 30 specifies that the guards on the edges are evaluated according to exponential delay distribution, on average of 30 times per time unit (hour). Figure 5(b) shows the heater controller model which moves between locations


T[id]=T0[id]
Normal
30
T[id]'==(cvec[id]*h[id] + bvec[id]*(u+−T[id]) + sum(j:rid_t)(Amat[id][j]*(T[j]+ −T[id])))/scale
H[id]==0 && T[id]<=get[id]
ASAP!
need[id]=true
T[id]>get[id]
need[id]=false
Low
30
T[id]'==(cvec[id]*h[id] + bvec[id]*(u+−T[id]) + sum(j:rid_t)(Amat[id][j]*(T[j]+ −T[id])))/scale

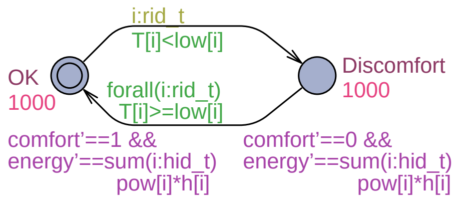
(a) Template for `Room` temperature.



(b) Template for `Heater` control.



(c) Template for strategy 1 (as in [10]).



(d) Template of `Monitor` for comfort and energy.

Figure 5: The main stochastic hybrid automata templates use in our case-study.

`Off` and `On` based on the temperature thresholds $on[r]$ and $off[r]$ where r is the room number that the heater is in. The guards are evaluated 60 times per hour on average and the energy cost of individual heater is computed by local clock variable c increasing with a rate $h[r]$ in location `On`. The heating can be switched over to another ($target$) room by a message $move[r]$, which is trivial in location `Off` but it requires switching the heater off in location `On`.

The central controller decides how to move the heater from one room to another room. If more than one room need the heater at the same time, the central controller will choose one candidate room according to the importance of the rooms. Figure 5(c) shows the template for the first strategy described in Table 2. The second and third strategies follow the same template except for the last condition given in the table. The idea of this strategy template is to monitor all possible heating switch-overs from room i to j, choose a particular destination based on stochastic choice (dashed edge with a stochastic weight $imp[i]$) and then issue a request to $move$ the heater from room j to $target$. The variable $H[i]$ denotes the heater identifier if it has a heater and 0 otherwise.

We consider two templates for the user profile requirements, a static profile as in [10]) and a dynamic profile. Fig. 4(b) shows our dynamic profile that models a more realistic office-user like behavior:

- the cycle starts with night settings (lower temperature thresholds),
- at 6 the settings are switched to a day mode for early preheating,
- between 8 and 9 a room user arrives and sets the low threshold,

- between 12 and 13 the user goes for a lunch and thus opens a ventilation (*bvec* is increased) for one hour,
- between 17 and 18 the user leaves and sets the night mode back and the cycle starts over again.

3.3 Properties

Uppaal-smc supports to visualize the values of expressions along runs, which helps the user monitor the behavior of the system. The query `simulate 1 [<=2*day] { T[1], T[2], T[3], T[4], T[5] }` computes one simulation trace and monitors the temperature of the environment and the five rooms. The result is shown in Fig. 6. The movements of heaters can be traced similarly.

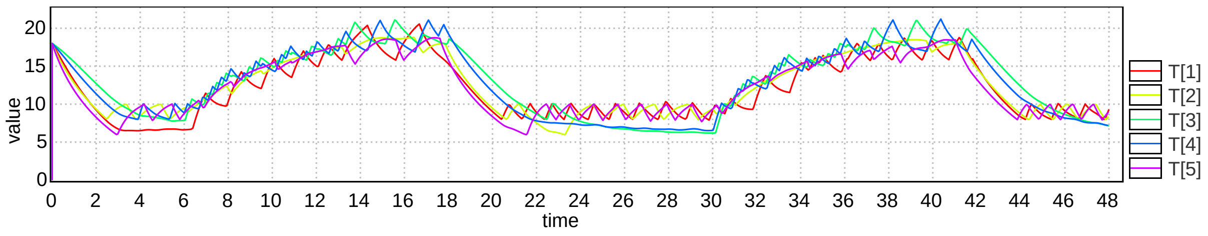


Figure 6: Sample simulation of room temperature dynamics.

The comfort requirement says that all temperatures in the building should be higher than the *low* threshold, thus the probability of a discomfort over two days can be estimated by the following query:

```
Pr[<=2*day](<> time>0 && exists(i:rid_t) T[i]<low[i])
```

This property can only estimate the chance of observing the temperature below a threshold, which may potentially last for very short time and thus a better metric is to estimate the duration of a comfort time, which requires a monitoring variable encoding this time.

For monitoring purpose we add an additional process `Monitor` to encode the notion of comfort as shown in Fig. 5(d). The idea of the monitor that it stays in the location `OK` while all the room temperatures are above their *low* threshold and it moves to discomfort whenever there is a room with temperature below *low*, then it would move back to `OK` when all temperatures are above their *low* threshold. Now the probability of a discomfort over two days can be estimated by the following query:

```
Pr[<=2*day](<> time>0 && Monitor.Discomfort)
```

Furthermore, location `OK` monitors the comfort time by specifying that the clock *comfort* is increased by rate 1, while it is being stopped in location `Discomfort`. Thus the *comfort* time can be estimated by the following queries:

```
E[time<=2*day; 1000] (<> max: comfort)
Pr[comfort<=2*day] (<> time>=2*day)
```

They both compute the same thing: the first computes the estimated value of comfort time over 1000 traces within a span of two days (the number of traces is explicit), and the second computes the probability of reaching time beyond two days within a large energy bound – which is trivially true, but in addition the tool provides a probability distribution of the *energy* values (the number of traces is implicit, determined by the tool parameters).

Similarly to comfort time, the consumed energy is estimated by a clock *energy*, which increases with a rate of current total power consumption. The currently drawn total power is a sum of $pow[i] * h[i]$ over all heaters (see constraints for $energy'$ in Fig. 5(d)), where $pow[i]$ is the power of a heater i and $h[i]$ is 1 when the heater i is on. The following two queries estimate the value of the *energy* variable:

```
E[time<=2*day; 1000] (<> max: energy)
Pr[energy<=1000000] (<> time>=2*day)
```

4 Application of the Framework

This section describes the application of our framework to the Hybrid System Verification Benchmark, which addresses the control of the temperature of rooms in a given building. Some experiments have been carried out with Uppaal-smc, which demonstrates how to estimate and compare the performance of various control strategies proposed in Section 3. According to the factorial design space, there are six various scenarios given in Table 3. Each scenario has different weather conditions and user profiles. Our

Table 3: Different scenarios for the energy aware buildings.

Case1A	Case1B	Case2A	Case2B	Case3A	Case3B
Daily Weather Static User	Daily Weather Dynamic User	Rapid Weather static User	Rapid Weather Dynamic User	Flat Weather Static User	Flat Weather Dynamic User

aim is to analyze and compare three control strategies for each scenario. The capabilities of our tool allows us, for each scenario, to

- estimate and compare the probability of experiencing discomfort within a certain time bound,
- estimate and compare the distribution of the accumulated time of comfort,
- and estimate and compare the distribution of energy consumption.

4.1 Analyzing the Probability of Experiencing Discomfort

We expect that users will experience discomfort when the room temperature drops below a low threshold. Given a specific scenario and control strategy, we want to estimate the probability of experiencing discomfort within a certain time bound. With the query

```
Pr[<=2*day](<> time>0 && Monitor.Discomfort)
```

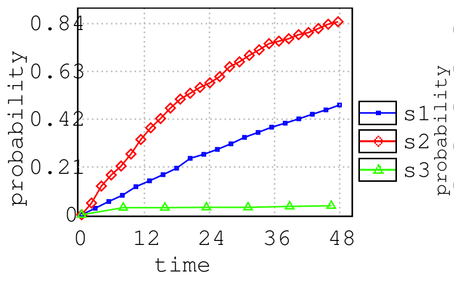
the probability of discomfort for each room with various strategies is evaluated. The superposed plots of the cumulative probability distribution for each scenario are shown in Fig. 7. We show the results of the different strategies side-by-side to compare them and to identify the best strategy. For example, Fig. 7(b) shows that Strategy3 is superior with lowest probability of discomfort. Furthermore, we want to know what the effect of the user profile is on the probability of discomfort. For example, with daily weather, the difference between static user and dynamic user should be inspected according to Fig. 7(c) and Fig. 7(f). From an overall point of view, if users are allowed to change the setting of room temperature (i.e. dynamic user), the probability distribution is concentrated in the time interval [8.0,8.8], but, for the static users the discomfort may be experienced at any time during two days. It is obvious that a dynamic user will experience much more comfort than a static user. According to the other sub-figures in Fig. 7, the same conclusion should be reached for the other scenarios.

As main conclusion we state the following:

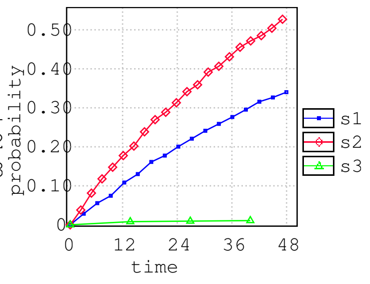
- *For dynamic users, the three strategies are indistinguishable in terms of the probability of experiencing discomfort.*
- *For static users, Strategy 3 gives the lowest probability of experiencing discomfort with daily weather model.*

4.2 Analyzing Accumulated Comfort Time

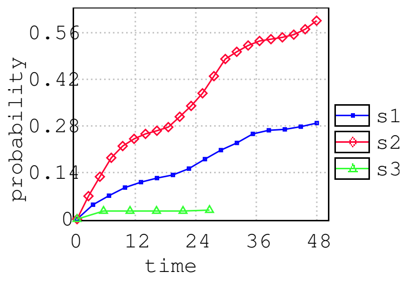
Accumulated comfort time is an important performance property to be estimated, which tells the total comfort time that a user may have been exposed to. The probability distributions of accumulated comfort time within an overall time-bound of 2 days are estimated with the query:



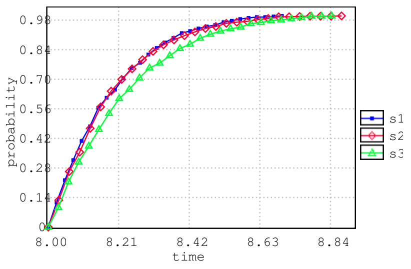
(a) Flat weather, Static user.



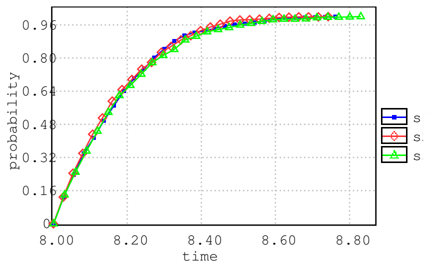
(b) Rapid weather, Static user.



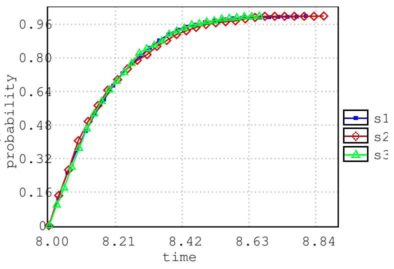
(c) Daily weather, Static user.



(d) Flat weather, Dynamic user.



(e) Rapid weather, Dynamic user.

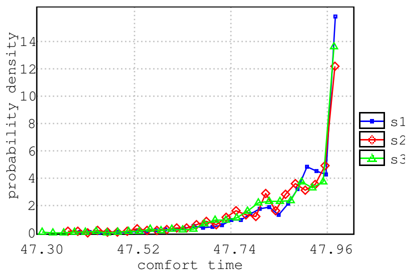


(f) Daily weather, Dynamic user.

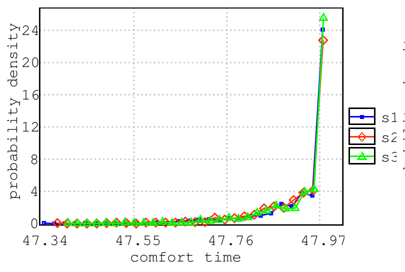
Figure 7: Cumulative probability distribution of discomfort for different cases.

```
Pr[comfort<=2*day] (<> time>=2*day)
```

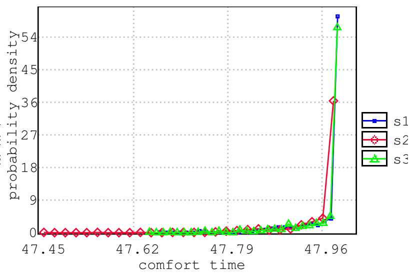
Each sub-figure in Fig. 8 shows the superposed plots of the probability density distribution of comfort time for different strategies. For example, Fig. 8(f) shows that Strategy 1 gives the maximum accumulated comfort time when the setting is daily weather with dynamic user. For Strategy 2 and Strategy 3, there is no distinguishable difference in terms of the distribution of accumulated comfort time over 2 days.



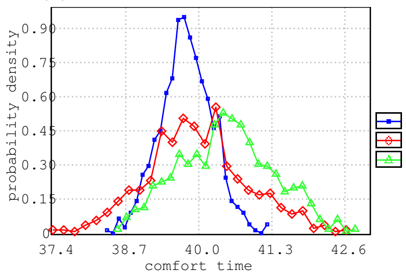
(a) Flat weather, Static user.



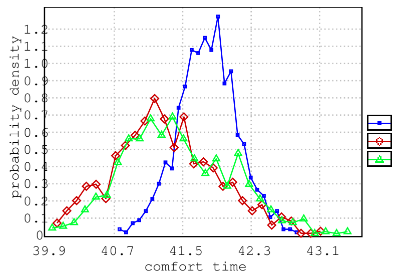
(b) Rapid weather, Static user.



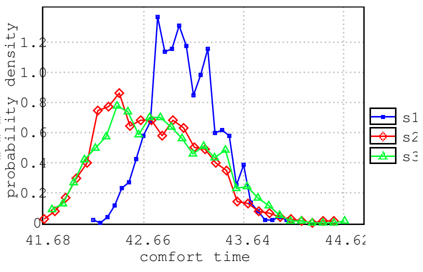
(c) Daily weather, Static user.



(d) Flat weather, Dynamic user.



(e) Rapid weather, Dynamic user.



(f) Daily weather, Dynamic user.

Figure 8: Probability density distribution of comfort time for different cases.

As main conclusion we state the following:

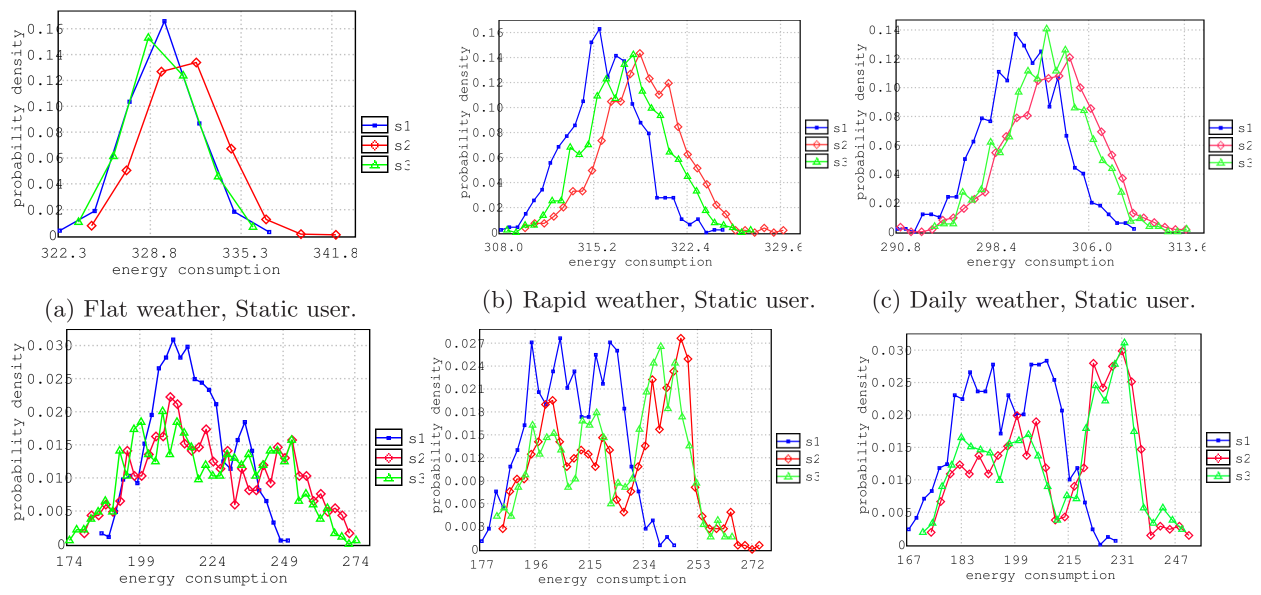
- *For static users, the three control strategies are indistinguishable in terms of the distribution of accumulated comfort time over 2 days;*
- *For dynamic users, Strategy 1 gives the maximum accumulated comfort time regardless of the weather model.*

4.3 Analyzing Energy Consumption of Strategies

For the energy aware buildings, it is very important to analyze and minimize the energy consumption under different scenarios. In this subsection, the performance analysis of the various strategies in terms of their energy consumption will be provided. By verifying the query:

```
Pr[Monitor.energy<=1000000](<>  time>=2*day)
```

the probability distribution of the total energy consumption over 2 days is estimated. The superposition of the plots for the three strategies are shown in Fig. 9. Concerning energy consumption, the situations with the dynamic user are much better than the one with static user. The difference between three strategies with the same environment and user profile are shown in each sub-figure. For example, the



(a) Flat weather, Static user. (b) Rapid weather, Static user. (c) Daily weather, Static user.

(d) Flat weather, Dynamic user. (e) Rapid weather, Dynamic user. (f) Daily weather, Dynamic user.

Figure 9: Energy consumption for different cases over two days.

superposed plots in Fig. 9(c) show that Strategy 1 consumes least energy. The detailed evaluation data are shown in Table 4.

Table 4: Energy consumption with daily weather and static user.

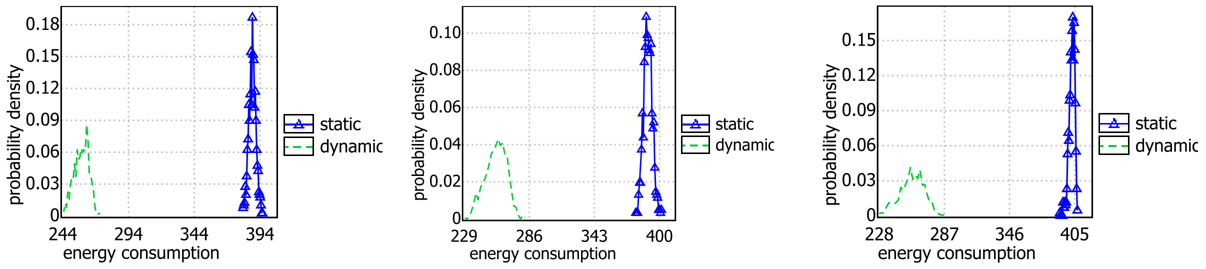
Strategy	Minimum	Maximum	Mean
Strategy 1	290.419	312.14	301.157
Strategy 2	294.462	314.08	303.82
Strategy 3	292.75	311.805	303.048

We make the following main conclusions:

- *For static users, there is no significant difference between the average energy consumption of the three strategies, independent of the weather condition;*
- *For dynamic users, Strategy 2 consumes least energy, independent of the weather condition.*

4.4 Analyzing Performance Effect of User Profile

We attempt to evaluate the effect of users profile on the overall energy consumption. For each strategy, we compare the difference of energy consumption between static users and dynamic users. The results are shown in Fig. 10, where the legend *dynamic* represents dynamic user and *static* represents static user. Here, the weather is daily changing with day and night.



(a) Strategy 1. (b) Strategy 2. (c) Strategy 3.

Figure 10: Probability density distribution of energy consumption for static and dynamic user.

We state the following main conclusion:

- *All three control strategies demonstrate significant reduction in energy consumption (approximately 40%) by taking the dynamic behavior of users into account.*

5 Conclusion

In this paper we have presented a modelling framework for energy aware buildings, which allows to evaluate the performance of proposed control strategies in terms of their induced comfort and energy profiles under varying environmental settings. The framework is realized in the tool Uppaal-smc, which has been extended with stochastic hybrid systems as a new and expressive modeling formalism offering a range of statistic model checking algorithms for their efficient analysis. Also the framework has been applied to the Hybrid Verification Tool Benchmark of [10]. In particular we have identified an energy optimal strategy (Strategy 2), which is even superior from a (accumulated) comfort point of view. Also, we have demonstrated that taking the dynamic requirements of a user into account may significantly lower the energy consumption. The framework is easily extendable with other environmental features such as changeable price of energy, as well as forecast of available renewable energy (wind, solar, . . .). Within the ongoing pilot project with the Danish national defense building administration we also plan to extend the framework to incorporate information from alarm-systems, ventilation systems, light systems in order to identity even more energy saving strategies.

Acknowledgements

This work was supported by Youth Foundation of the National Natural Science Foundation of China (Grant No. 60903021) and Creativity Project for East China Normal University. Also the work was supported by the Danish VKR Center of Excellence MT-LAB (`www.mt-lab.dk`), and the Sino-Danish Basic Research Center IDEA4CPS (`www.idea4cps.dk`).

References

1 Alur R, Courcoubetis C, Halbwachs N, Henzinger TA, Ho PH, Nicollin X, Olivero A, Sifakis J, and Yovine S. The algorithmic analysis of hybrid systems. Theor. Comput. Sci. 138, 1 (1995), 3–34.

2 Alur R, and Dill DL. A theory of timed automata. Theor. Comput. Sci. 126, 2 (1994), 183–235.

3 Basu A, Bensalem S, Bozga M, Caillaud B, Delahaye B, and Legay A. Statistical abstraction and model-checking of large heterogeneous systems. In Formal Techniques for Distributed Systems, J Hatcliff and E Zucca, Eds., vol. 6117 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, pp. 32–46.

4 Brunner H, de Nigris M, Gallo AD, Herold I, Karg WHL, Koivuranta K, Papic I, Lopes JP, and Verboven P. Mapping & gap analysis of current european smart grids projects. Tech. rep., Smart Grids ERA-Net, 2012.

5 Bulychev PE, David A, Larsen KG, Legay A, Mikučionis M, and Poulsen DB. Checking and distributing statistical model checking. In NASA Formal Methods (2012), vol. 7226 of Lecture Notes in Computer Science, Springer, pp. 449–463.

6 Sgcc white paper on green development (the first among chinese corporations). `http://www.sgcc.com.cn/ywlm/socialresponsiility/whitepaper`, 2010.

7 David A, Larsen KG, Legay A, Mikučionis M, Poulsen DB, Vliet JV, and Wang Z. Statistical model checking for networks of priced timed automata. In FORMATS (2011), LNCS, Springer, pp. 80–96.

8 David A, Larsen KG, Legay A, Mikučionis M, and Wang Z. Time for statistical model checking of real-time systems. In Proceedings of the 23rd international conference on Computer aided verification (Berlin, Heidelberg, 2011), LNCS, Springer-Verlag, pp. 349–355.

9 Deshpande A, Godbole DN, Göllü A, and Varaiya P. Design and evaluation tools for automated highway systems. In Hybrid Systems (1995), R Alur, TA Henzinger, and ED Sontag, Eds., vol. 1066 of Lecture Notes in Computer Science, Springer, pp. 138–148.

10 Fehnker A, and Ivancic F. Benchmarks for hybrid systems verification. In HSCC (2004), R Alur and GJ Pappas, Eds., vol. 2993 of Lecture Notes in Computer Science, Springer, pp. 326–341.

11 Jiang Z, Pajic M, Moarref S, Alur R, and Mangharam R. Modeling and verification of a dual chamber implantable pacemaker. In TACAS (2012), C Flanagan and B König, Eds., vol. 7214 of Lecture Notes in Computer Science, Springer, pp. 188–203.

12 Katoen JP, Zapreev IS, Hahn EM, Hermanns H, and Jansen DN. The ins and outs of the probabilistic model checker MRMC. In Proc. of 6th Int. Conference on the Quantitative Evaluation of Systems (QEST) (2009), IEEE Computer Society, pp. 167–176.

13 Lee EA. Cyber physical systems: Design challenges. In ISORC (2008), IEEE Computer Society, pp. 363–369.

14 Legay A, Delahaye B, and Bensalem S. Statistical model checking: An overview. In RV (2010), vol. 6418 of Lecture Notes in Computer Science, Springer, pp. 122–135.

15 Sen K, Viswanathan M, and Agha G. Statistical model checking of black-box probabilistic systems. In CAV (2004), LNCS 3114, Springer, pp. 202–215.

16 Tomlin C, Pappas GJ, Lygeros J, Godbole DN, and Sastry S. Hybrid control models of next generarion air traffic management. In Hybrid Systems (1996), PJ Antsaklis, W Kohn, A Nerode, and S Sastry, Eds., vol. 1273 of Lecture Notes in Computer Science, Springer, pp. 378–404.

17 Us white house press release on smart grids, october 27, 2009. `http://www.whitehouse.gov/the-press-office/president-obama-announces-34-billion-investment-spur-transition-smart-energy-grid`.

18 Younes HLS, and Simmons RG. Statistical probabilistic model checking with a focus on time-bounded properties. Inf. Comput. 204, 9 (2006), 1368–1409.