Analysis of Energy-Efficient Buildings through Simulation and Formal Methods

Maryam Ehsanpour[1], Luciano Baresi [2], Matteo Rossi [2]
Ernesto Damiani[3]

[1] Dipartimento di Informatica, Universita degli Studi di Milano, Via Comelico 39/41,
20135 Milano,Italy
maryam.ehsanpour@unimi.it

[2] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{luciano.baresi,matteo.rossi}@polimi.it

[3] Department of Electrical and Computer Engineering, Khalifa University
Al Zafranah, Abu Dhabi, United Arab Emirates
ernesto.damiani@kustar.ac.ae

Abstract.

Abstract. In the last few years, the increasing interest in energy saving has led to invest resources in many research fields, in order to develop new technologies to reduce energy consumption. One such field is energy management systems for buildings. The interactions between the system controlling the building and its environment are complicated by the fact that they involve important timing properties, which can be expressed through a variety of formalisms. This paper introduces a formal model, expressed in a temporal logic language, through which designers can evaluate the temporal behavior of buildings' energy management systems. To this end, the formal model is analyzed through simulation, that is, through the generation of traces satisfying the temporal logic model.

Keywords: Data analysis, Energy efficient, Formal methods.

1   Introduction

Managing the technology that is used to control the various functions of modern buildings is a complex task. The U.S. Department of Energy estimates that buildings in industrialized countries account for around 40% of total energy consumption. Researchers have shown that savings in the order of 20-30% can be gained through careful energy management and control in buildings, even without changing the structure of the hardware configuration of their energy supply systems[1]. The problem of modeling and designing software control systems has been approached through different techniques, such as temporal logic languages[2], which are supported by a wide range of

techniques for model analysis and verification. This work aims to introduce a solution for describing the management and control of energy consumption in buildings through rules expressed in temporal logic. The architecture of the approach is described through a simple example, namely the control of Rooms provided with various appliances.

The rest of the paper is organized as follows. Section 2 introduces the simulation tools on which the approach is founded; Section 3 introduces the temporal logic language used for describing control rules; and Section 4 illustrates the case study which exemplifies the main characteristics of the proposed analysis method based on temporal logic rules. Section 5 concludes. The Appendix shows an example of execution trace produced through the Zot tool, which is used in this work to simulate temporal logic models.

2 Tool support

2.1 OpenModelica

A number of well-established techniques have been developed over the years to model physical phenomena, which constitute the environment of software systems. Typically, these phenomena are described through a continuous notion of time, using approaches that can be classified as either causal or acausal. In causal approaches, the model is decomposed into blocks which interact through input and output variables. The object-oriented language Modelica [3], instead, follows a acausalapproach where each (physical) component is specified through a system of differential algebraic equations; connections between modules, which correspond to physical principles, are set by equating effort variables and by balancing flow variables. OpenModelica[1] is an open-source simulation engine based on the Modelica language; it can be interfaced with the Zot tool described below to create closed-loop simulations of software systems and their environments [4].

2.2 Zot

Zot[2]can perform formal verification of models described through temporal logic formulae. More precisely, let us call S the model of the designed system, which corresponds to a set of temporal logic formulae. If S is input to the Zot tool, the latter looks for an execution trace of the modelled system; that is, for each time instant and for each predicate appearing in the formulae of S, it looks for an assignment of values which make the formulae of S true. If no such trace is found (i.e., if the model is *unsatisfiable*), then the model is contradictory, meaning that it contains some flaw that makes it unrealizable in practice.

---

[1]http://openmodelica.org
[2]github.org/fm-polimi/zot

3 The TRIO temporal logic language

In this section we provide a brief overview of the temporal logic, called TRIO [1], which has been used in this work to capture the rules governing the energy management system. TRIO allows users to express timing properties of systems, including real-time ones. TRIO adopts a linear notion of time, and can be used to express properties over both discrete and continuous temporal domains. Table 1 presents some typical TRIO temporal operators, which are defined in terms of the basic *Dist* operator, where Dist(φ, d) means that formula φ holds at the instant exactly *d* time units from the current one.

Table 1. TRIO temporal operators, including thoseused in the formulae of this paper

Operator	Definition	Meaning
Futr(φ, d)	$d > 0 \wedge \text{Dist}(\varphi, d)$	Φ occurs d instants in the future
Past(φ, d)	$d > 0 \wedge \text{Dist}(\varphi, -d)$	Φ occurs d instants in the past
Alw(φ)	$\forall t\, (\text{Dist}(\varphi, d))$	Φ always holds
Lasts(φ, d)	$\forall (0 < t < d \rightarrow \text{Futr}((\varphi, t))$	Φ holds for the next d time units
Lasted(φ, d)	$\forall (0 < t < d \rightarrow \text{Past}((\varphi, t))$	Φ helds for the last d time units
Until(φ, ψ)	$\exists t\, (\text{Futr}(\psi, t) \wedge \text{Lasts}((\varphi, t))$	Ψ will occur and φ will hold until then
Since(φ, ψ)	$\exists t\, (\text{Past}(\psi, t) \wedge \text{Lasted}((\varphi, t))$	Ψ occured and φ held since then

4 Case Study: Rooms with Appliances

In this section we briefly describe how the energy management system is captured through the TRIO temporal logic language. In particular, we focus on an example of building whose rooms are provided with appliances whose energy consumption needs to be controlled. The model of the building is fed to the Zot tool to generate traces, which can then be analyzed to unearth anomalous behaviors. Fig. 1. shows the corresponding Modelica diagram describing the structure and the components of the analyzed system. The appliances considered are: 1. the Electric Boiler, whose desired temperature is 70°C; 2. the *Washer* whose desired temperature is 60°C and whose washing cycle lasts an hour;3. The *Bath*, which can be used both as shower or as a bath; 4. The *Oven*, whose nominal power consumption is 1000*W*; 5. The *DishWasher*, whose washing cycle is an hour and a half; 6. The *Refrigerator*, whose nominal power consumption is 90*W*;7. The *HoodCookTop* and the *Stove*, which are used when a user cooks with the stove and keeps the hood turned on.

These appliances are connected through three different networks: the hydraulic, the thermal, and the electric network.

In the rest of this section, we present the formulae capturing the behavior of the appliances.
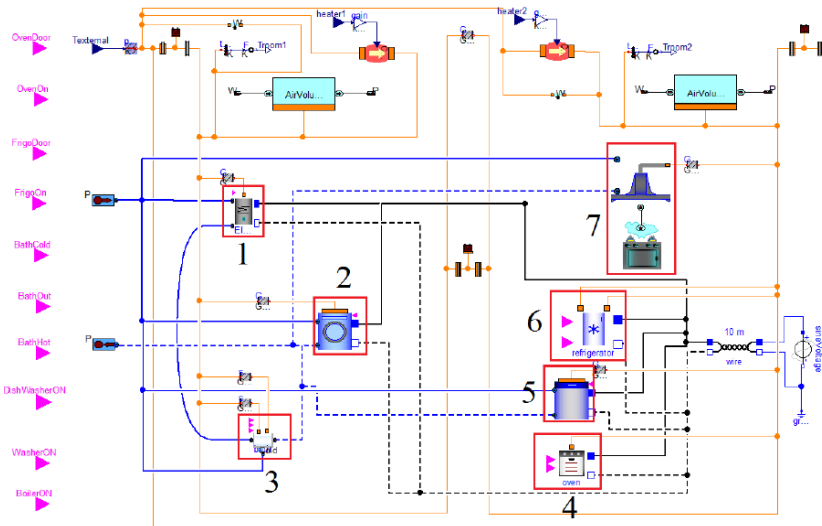
Fig. 1. Diagram describing rooms with appliances

- Electric boiler

For the *Electric Boiler*, if the water temperature is less than 70°C for 2 minutes, then the boiler is turned on (represented by predicate boilerOn being true):

$$Alw(Lasted(boiler\_water\_temp< boiler\_desired\_temp, 2)\Rightarrow boilerOn).$$

If the water temperature is higher than 70°C for 2 minutes, then the controller is turned off (i.e., "not boilerOn" holds):

$$Alw(Lasted(boiler\_water\_temp> boiler\_desired\_temp, 2)\Rightarrow \neg boilerOn).$$

The following formula constrains the difference between the current temperature and the one at the next minute when the boilers turned on; in particular, the temperature must increase (i.e., the difference is greater than 0), but by no more than 2 degrees (i.e., the difference is less than 3).

$$Alw(boilerOn \Rightarrow next(boiler\_water\_temp) - boiler\_water\_temp> 0 \wedge$$
$$next(boiler\_water\_temp) - boiler\_water\_temp< 3)$$

Also after the boiler is turned off, the temperature decreases by with a rate of at most 2 degrees per minute: $Alw(\neg boilerOn \Rightarrow next(boiler\_water\_temp) - boiler\_water\_temp<1 \wedge next(boiler\_water\_temp) - boiler\_water\_temp>-3)$

The following formula constrains the range of the temperature to be between -20°C and 100°C: $Alw(boiler\_water\_temp< 101 \wedge boiler\_water\_temp> -20)$

- Washer:

The time washing cycle lasts60 min:

$$Alw(washer\_start\Leftrightarrow Futr(washer\_end, 60))$$

The washer is on between a "start" and an "end" command.

$$Alw(washer\_on \Leftrightarrow Since_{ei}(\neg washer\_end, washer\_start)$$

The washer cannot start again until the current cycle has finished.

$$Alw(washer\_start \Rightarrow Until_{ei}(\neg washer\_start, washer\_end)$$

- Dryer :

The model of the dryer is very similar to the one of the washer.

The time spin/dryer cycle is 30 min. $Alw(dryer\_start \Leftrightarrow Futr(dryer\_end, 30))$

The dryer is on between a "start" and an "end" command.

$$Alw(dryer\_on \Leftrightarrow Since_{ei}(\neg dryer\_end, dryer\_start)$$

The dryer cannot not start again until the current cycle has finished.

$$Alw(dryer\_start \Rightarrow Until_{ei}(\neg dryer\_start, dryer\_end)$$

- Oven :

If the door of the oven is opened for 2 minutes, the controller is on (i.e., it flashes):

$$Alw(Lasted(oven\_door\_open, 2) \Rightarrow oven\_door\_flash)$$

- Dishwasher :

The model of the dishwasher is also very similar to the one of the washer and of the dryer.

- Refrigerator :

The model of the refrigerator is similar to the one for the oven; that is, if the refrigerator door is opened for 2 minutes, the controller is on (i.e., it flashes).

5   Conclusion

This paper introduces imulation-based mechanisms based on rules formalized in temporal logic, which can help improve the user's understanding of modeled systems. The proposed approach can be used to model (complex) systems and describe their properties formally; it allows the user to simulate the behavior of the system and verify the satisfiability of the properties of interest. We exemplified our approach on a case study concerning the modeling of an energy management system for a building.

References

1. Guian.X , Xu.Z , Jia.Q : Energy efficient building facilitated by micro grid . IEEE Power and Energy Society General Meetings, 2010.
2. Furia, C., Mandrioli, D., Morzenti, A., Rossi , M : Modelling Time in computing.Springer-Verlag, 2012.
3. Fritzson.P.: Principles of object-oriented modeling and simulation with Modelica 2.1. John Wiley and Sons, 2004.
4. Baresi, L., Ferretti, G., Leva, A., Rossi, M : Flexible logic-basedco-simulation of Modelica models. In: IEEE InternationalConference on Industrial Informatics (INDIN),pp. 635-640 (2012)

Appendix: Example of execution trace of the model

```
------ time 0 ------
BOILER_WATER_TEMP = 68.0
REF_DOOR_FLASH BOILERON
REF_DOOR_OPEN
------ time 1 ------
BOILER_WATER_TEMP = 70.0
BOILERON
------ time 2 ------
BOILER_WATER_TEMP = 71.0
------ time 3 ------
BOILER_WATER_TEMP = 69.0
REF_DOOR_FLASH
BOILERON
------ time 4 ------
**LOOP**
BOILER_WATER_TEMP = 70.0
------ time 5 ------
BOILER_WATER_TEMP = 68.0
REF_DOOR_OPEN
------ time 6 ------
BOILER_WATER_TEMP = 66.0
------ time 7 ------
BOILER_WATER_TEMP = 64.0
BOILERON
------ time 8 ------
BOILER_WATER_TEMP = 65.0
BOILERON
------ time 9 ------
BOILER_WATER_TEMP = 66.0
BOILERON
------ time 10 ------
BOILER_WATER_TEMP = 67.0
BOILERON
------ time 11 ------
BOILER_WATER_TEMP = 68.0
BOILERON
------ time 12 ------
BOILER_WATER_TEMP = 69.0
OVEN_DOOR_OPEN
 BOILERON
------ time 13 ------
BOILER_WATER_TEMP = 70.0
```

```
BOILERON
------ time 14 ------
BOILER_WATER_TEMP = 71.0
 BOILERON
------ time 15 ------
BOILER_WATER_TEMP = 72.0
OVEN_DOOR_OPEN
BOILERON
------ time 16 ------
BOILER_WATER_TEMP = 73.0
OVEN_DOOR_OPEN
------ time 17 ------
OVEN_DOOR_FLASH
BOILER_WATER_TEMP = 73.0
REF_DOOR_FLASH
------ time 18 ------
BOILER_WATER_TEMP = 71.0
------ time 19 ------
BOILER_WATER_TEMP = 70.0
OVEN_DOOR_OPEN
 REF_DOOR_OPEN
------ time 20 ------
BOILER_WATER_TEMP = 69.0
 REF_DOOR_FLASH
BOILERON
------ end ------
```