© Copyright IFAC
ENERGY SAVING CONTROL IN PLANTS AND BUILDINGS
Bansko, Bulgaria, 2006

AUTOMATION IN BUILDINGS AND FORMAL METHODS

Pavel Burget [*] **Ondřej Šprdlík** [**] **Richard Šusta** [***]

[*] *DCE-Prague: Department of Control Engineering, Faculty of Electrical Engineering, Prague, Czech Republic*
burgetpa@control.felk.cvut.cz
[**] *DCE-Prague,* sprdlo1@control.felk.cvut.cz
[***] *DCE-Prague,* susta@control.felk.cvut.cz

Abstract: The paper presents a case study based on our practical model of energy saving in a small building equipped by an industrial distributed control system. First, we overview our model built over LonWorks networking platform and then we discuss increasing the reliability of control programs by formal methods. Their possible application is demonstrated by the example of verifying the control program for a solar space heating. We show that some safety verifications are performable without constructing a whole hybrid model. Finally, we discuss the results achieved up to date.
The principles presented in this case study are applicable on a general level for other similar systems. *Copyright © IFAC 2006*

Keywords: automatic control, distributed control, control program, programmable logical controllers, model, formal methods, case study

1. INTRODUCTION

This paper discusses how to increase the reliability of a control program. It follows from our experiments with the practical model of energy saving in a small building controlled by a distributed industrial system. There are two main reasons for constructing model: research and education.

At first, distributed solutions offers many advantages, for example in terms of reducing wiring, flexibility, and system reliability. They naturally improve the transparency of programs by their subdividing into several smaller units instead of concentrating all elaborations into a single central point. Such splitting has many advantages concerning well-arranged implementation.

However, such approach has also potential drawbacks for further studies, as network problems and software shortage caused by missing ready-made software modules for distributed control. Therefore, new programs are written.

The reliability of new programs is often seen as an unknown link. Fortunately, our and other's practical experiences show that overall maintenance of new software usually requires higher expenses only in the initial phases of a system's operation, because software repair effort often follows nonhomogeneous Poisson

[1] A part of this work is supported by 2C06010 project.

process with intensity function $\mu\lambda\,\mathrm{e}^{-\lambda t}, t > 0$, where $\mu \geq 0$ and $\lambda \geq 0$ are parameters of a program, e.g. Mockus *et al.* (2003). But the initial phases can cost more than admissible under unfavorable conditions, especially in smallish projects with limited quotations for maintenance.

Fortunately, smaller distributed programs can be checked in reasonable time even by methods that are generally susceptible to the well-known state-explosion problem. Therefore, we have examined possible applications of formal methods in this area. Apart from other purposes, our model of energy saving in a small building is also intended for an education, possibly for a remote learning. In this case, the situation is much worse due to inexperienced programmers and preprocessing their codes by formal methods can protect the model against shock stresses in extreme states.

Organization of the paper: In section 2, we overview our experimental model of energy saving automation in buildings. The results and gained experiences with formal methods are given in section 3. In section 4, we conclude with a summary and remarks about future plans. In section 5, we discuss formal methods utilized for analysis and related works.

2. THE EXPERIMENTAL MODEL

The model has been designed as a two-floor administrative building, see Figure 1. Its dimensions are 2600x750x1600mm. Two offices, one on each floor, constitute the right-hand part of the building, while the left-hand part is composed just of a big lecture room. Between both parts there is a lift shaft and a corridor. The left-hand part can be departed in order to be exhibited at a fair or conference.

Several physical quantities, such as temperature and lighting intensity can be controlled. Furthermore, the behavior of the building can depend on its state, i.e. on the presence of people in certain areas of the building, on the time of day, on the environment of the building, etc. At this stage of the project, we have involved occupancy detection and time-of-day behavior. Other schemes of behavior are under consideration now.

In the right-hand part, there is an electric warm-water furnace and a solar collector which are used to heat the two offices using radiators. The commissioning of its control program will be main topic of Section 3. Photograph 2 shows the uncovered model.

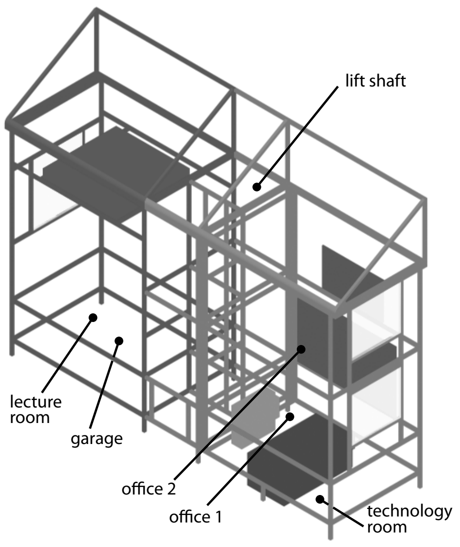

Fig. 1. The structure of the model

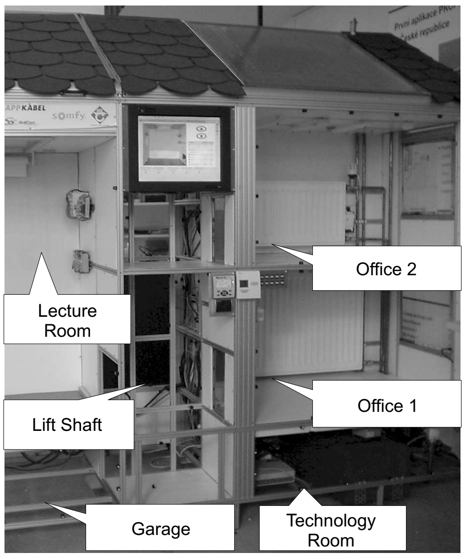

Fig. 2. The uncovered model

2.1 Control devices

The control system of the building is structured according to the hierarchical model of distributed control systems as can be seen in Fig. 3.

Level 1 corresponds to the connection of field devices directly, i.e. locally, or via a sensor bus such as DALI or EnOcean. LON can be used as sensor bus too. Relating to our model, one control system with corresponding connection of sensors/actuators is used for each room. In other words, the sensor bus or local connection of input/outputs is used within each room whereas the control systems in individual rooms are at level 2 and are connected via LON. Individual LON segments are inter-

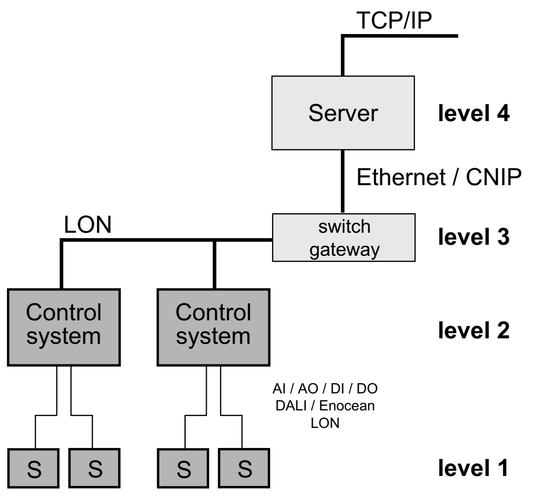

Fig. 3. Hierarchical structure of the building

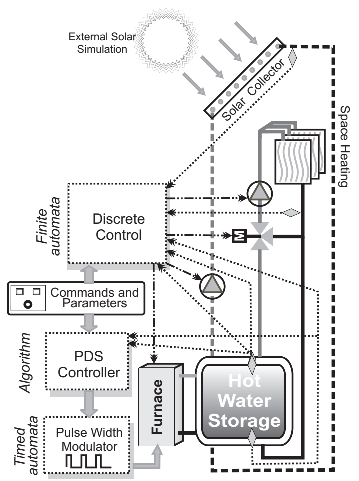

Fig. 4. Simplified Structure of Heating Control

connected at level 3 via L-Switch device and the gateway functions are performed by L-IP device, which enables connection of the entire building to the superior server and internet.

The model contains several rooms based on different technologies and approach to the connection of devices at level 1 (see Fig. 3). In the following subsection we will describe only one basement room which program was verified.

2.1.1. Technology room In the lower right-hand part – basement there is a technology room with a heating furnace, LonWorks switch and IP router and power supplies for the whole part. There is a dedicated PLC which is used to control the furnace, which program will be main topic of the next section.

There are two warm-water pipelines for the radiator and solar collector circulations. For simplicity sake they lead to the tank in the furnace and the pump in each pipeline determines whether the water circulates or not. In a real installation there would be a heat exchanger in the furnace and a valve for each pipeline. Thus, the the solar collector on the roof is involved in the heating system and its operation is controlled by controlling the pump in the respective pipeline. The temperature is measured at the input and output of the furnace and at the output of the solar collector.

It is important the furnace control system is completely autonomous because it is crucial for safe operation that the temperature and the pressure in the furnace do not exceed their maximum values. Thus we decided to use TAC Xenta PLC with LON connection so that we have enough information about the furnace for the visualization and the furnace can be safely operated even if the communication goes down and some disturbances appear.

In order to be as realistic as possible we have also installed a three-way valve which can be used to bypass the furnace and perform circulation of the water just in the radiators. In this way we can optimize the energy consumption for the heating system.

Modular programmable devices, which are extensible in their number and potentials, offer other interesting advantage for an automation in buildings — a complete freedom to design every possible functionality. To open possibility for modeling different strategies, the solar space heating, which includes the furnace, radiators, and the solar collector, is controlled by WAGO modular PLC.

3. TESTING OF SOLAR HEATING CONTROL

We have tested the controller for the solar space heating. Its program was designed, written and debugged by another group, which give us only the plan of building and their 'finished' source code. Our analysis consists of several phases: the decomposition of the program, a checking safety properties and liveness properties, and tests of the whole model. We discuss these steps in the following subsections.

3.1 Decomposition of Control Program

Figure 4 depicts the simplified structure of the controller of heating system described in Subsection 2.1.1. Its discrete part switches valves and pumps according to comparisons of measured temperatures to specified thresholds.

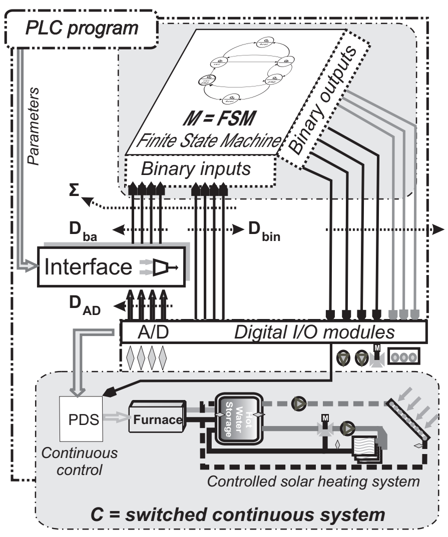

Fig. 5. Extraction of FSM

The furnace temperature is stabilized by PDS (proportional derivative and strain) controller algorithm that is based on input and output temperatures inside the hot water storage.

All parts of this program were extracted from CoDeSys programming environment for WAGO PLC and prepared for the further processing by decomposing into initializations and algorithms of continuous and discrete control. We also preprocessed PLC program by the transfer set conversion tool that gives results suitable for NuSMV checkers. This step is not necessary and can be omitted, but it increases the speed of the checking, see related works in Section 5.

The PDS algorithm does not contains loop and always terminates. We successfully checked its validity by its comparing its behavior with a known reference controller. The action variable of the PDS controller is converted by a PWM (pulse width modulator) subroutine and send to the furnace. The PWM subroutine behaves as an autonomous timed automaton with 2 states and it clearly contains no deadlock.

Finally, we joined verified PDS controller together with the solar heating into $\mathcal{C}$ general switched continuous system, defined for example in Labinaz *et al.* (1997).

We concentrate here on isolation of $\mathcal{M}$ discrete part that controls $\mathcal{C}$, as depicted in Figure 5. Let us denote a set of all binary variables as $\mathcal{B}$ and a set of all integer numbers as $\mathbb{I}$.

We also introduce $\alpha(A)$ defined by Cartesian product $\{0,1\}^{|A|}$ where A is any finite nonempty ordered set of binary variables, $A \subset \mathcal{B}$. Using $\alpha(.)$ we define $\mathcal{M}$, a finite state machine (FSM) accepting only binary inputs $\Sigma \subset \mathcal{B}$. It is defined as a tuple

$$\mathcal{M} \overset{df}{=} \langle \Sigma, \Omega, V, A_\Sigma, \delta_P, Q_0 \rangle \tag{1}$$

where Σ, Ω, and V are a nonempty set of binary inputs, outputs, and internal variables. A nonempty subset $A_\Sigma \subset \alpha(\Sigma)$ stands for all recognized inputs, $Q_0 \in \alpha(V) \times \alpha(\Omega)$ represents an initial state of δ_P program described as partial mapping:

$$\delta_P(q,x) : \alpha(V) \times \alpha(\Omega) \times A_\Sigma \rightarrow \alpha(V) \times \alpha(\Omega) \tag{2}$$

where $q \in \alpha(V) \times \alpha(\Omega)$ and $x \in A_\Sigma$.

$\mathcal{M}$ belongs to Moore's automaton family of FSM with $X = \alpha(\Sigma)$ input alphabet, $Y = \alpha(\Omega)$ output alphabet, and $Q = \alpha(V) \times Y$ set of states. Its output function ω depends only on Y and δ transition function and it corresponds to δ_P in all defined points. Q_0 initial state is given by an initialization block of the program.

Our FSM accepts only binary signals. All its inputs are directly mappable into Ω outputs due to joining the PDS controller with the solar heating, see in Figure 5. On the contrary, $\mathcal{C}$ outputs split into two disjoint input sets $\Sigma = D_{\text{bin}} \cup D_{\text{ba}}$, where $D_{\text{bin}} \subset \mathcal{B}$ includes all direct binary inputs of $\mathcal{M}$ and $D_{\text{ba}} \subset \mathcal{B}$ represents binary results of mapped analog values from A/D convectors.

Without loss of generality, we may assume that the block of A/D convectors gives D_{AD} set of variables that consist of temperature values and their required linear combinations. Their data are integer numbers with finite resolutions r_{AD}, usually 12 or 16 bits.

The interface converts D_{AD} into D_{ba} set of binary variables compare their values with constant parameters from a given set P. The mappings have the forms:

$$\phi_i(D_{\text{AD}}, P) : \mathbb{I} \times \mathbb{I} \rightarrow \{0,1\} \tag{3}$$

3.2 Safety properties

We show in this subsection that some safety properties can be verified fast without modeling continuous part of the system, i.e., without identifying whole controlled general discrete dynamic system (GGHDS).

Safety properties are usually described in computational tree logic (CTL) by AG x statements, i.e., x proposition should hold globally on all execution paths. To verify proper reactions to events, we replace x by $x_1 \Rightarrow \text{AX}\ x_2$, which asserts that x_2 becomes true on all execution paths at the latest in the next step after detecting x_1 event. For example:

$$\text{AG}(t_{\text{solar}} \geq t_{\text{maxs}} \Rightarrow \text{AX}\ \mathit{SolarValve} = 1)$$
$$\text{AG}(t_{\text{tanker}} \geq t_{\text{maxt}} \Rightarrow \text{AX}\ \mathit{Heating} = 0)$$

The first statement specifies that $\mathit{SolarValve}$ valve should always open when t_{solar} solar panel temperature will reach t_{maxs} maximum level. Similarly, the second specification tests switching the heating off if overheating the hot water tanker is detected. The both specifications convert temperature to binary values by comparisons $\mathcal{M}$ inputs with constants.

We define $\alpha(\Sigma, \mathcal{C}) \subset \alpha(\Sigma)$ as a set containing n-tuples, $n = |\Sigma|$, with inputs measured on all possible trajectories of $\mathcal{C}$. Thus, $\alpha(\Sigma, \mathcal{C})$ represents an input alphabet of $\mathcal{C}$ in feedback loop $\mathcal{M}$ and $\mathcal{C}$. We define $L(\alpha(Sigma, \mathcal{C}), \mathcal{C})$ language as a set of all possible input sequences of $\mathcal{C}$ in the feedback loop. If we denote Kleene closure by $()^*$ then

$$L(\alpha(Sigma, \mathcal{C}), \mathcal{C}) \subseteq (\alpha(\Sigma(\mathcal{C})))^* \subseteq (\alpha(\Sigma))^* \quad (4)$$

It is possible to prove the following. Let be given a CTL specification $S = AG(x_1 \Rightarrow AXx_2)$, where x_1, x_2 are logical expressions with variables of $\Sigma \cup \Omega$. If S is satisfied for all strings w of some given L input language, which is accepted by M, then S will also hold for all strings from any non-empty subset $L_i \subseteq L$. The same conclusion is also valid for a simplified CTL specification AGx_1.

In reality, the result is not surprising because the both safety specifications deal with proper reactions of FSM to input events. Therefore, the reverse conclusion does not hold. The validity of S specification for strings from $L_i \subset L$ is only a necessary condition, but not a sufficient one, because it does not guarantee the same for whole L language.

There are generally two ways how to modify the interface for checking CTL specifications without C — replacement of ϕ mapping in Equation 3 and reducing resolutions r_{AD} of A/D convectors. The first approach utilizes D_{ba} as direct inputs, so a model checker will test specifications utilizing strings from $(\alpha(\Sigma))^*$ input language. Practical experiments with NuSMV model checker lead to approximately 2^{27} reachable states and 229764 binary decision diagrams (BDD) nodes. The both safety specifications were validated in 3 seconds on 2.4 GHZ Pentium with 1GB RAM.

The second method replaces D_{AD} variables by enumerated types that decrease r_{AD} resolution of A/D convectors. We replaced measured temperatures by 16-state variables, definable in NuSMV model checker. In this case, an input language is less than $(\alpha(\Sigma))^*$ and false announcements are reduced. The checking is only slightly prolonged even if the number of reachable states was nearly 2^{39}. The states added by enumerated types increase BBD nodes approximately only twice, to 448623 nodes. The both safety propositions required 9 seconds.

3.3 *Liveness Properties*

We will test liveness properties of output bits to prevent "a hanged output", i.e., an output remaining forever in one state. The liveness should globally hold on all executions paths. In any state, it should always exists at least one future path (EF) that changes $b \in \Omega$ state.

We can test liveness of outputs in CTL by the following pair of specifications:

$$\text{AG}(b = 1 \Rightarrow \text{EF } b = 0)$$

$$\text{AG}(b = 0 \Rightarrow \text{EF } b = 1)$$

It can be easily proved that a validity of EF predicate statement performed with the aid of strings from a given L language does not generally imply its validity for subsets or L. On the contrary, if liveness properties are not satisfied for all strings of L then they cannot be also valid for its subsets.

These test were performed by NuSMV by similar way as the checks of safety properties. They reveals that two bits remains in state "1".

3.4 *Testing the Model*

The whole control program was transformed into HyTech model Henzinger *et al.* (1997). This step required a lot of effort but it did not reveal new facts because the solar heating is relatively a slow and stable process.

Besides, HyTech utilizes ACTL logic specifications that do not allow utilizing predicates needed for liveness properties. We could test only safety properties, which were already validate by easier way.

After all, we utilized whole model only for checking threshold levels to prevent frequently switching valves on and off.

4. CONCLUSION AND FUTURE WORK

In this paper, we have reported a case study in applying formal methods for automation in buildings. We introduced our practical model of distributed automation in building with solar space heating.

We outlined the decomposition of a control program to continuous and discrete part and we showed that some verification can be easier performed without continuous parts. This approach also allow partial checking liveness properties, which cannot be tested by HyTech model checker.

Our work may be considered as a suggestion for similar studies. The project is still in progress and more complex control programs are now debugging, so our future work naturally concerns their validations. Our long term intentions will include automatic checking programs for remote learning to let students test them on real models.

5. APPLIED METHODS AND RELATED WORKS

During our research efforts, we surveyed hundreds of related works in the area of analyzing control programs, therefore, we narrow their list only to methods closely related to this paper and our contributions.

- We convert an analyzed program into transfer sets that was introduced in Šusta (2003). In short, the transfer sets expresses control programs as operations with concurrent evaluations of several expressions all at once and they naturally remove all inefficient coding, especially in case of control programs written in ladder diagrams.
- The transfer sets can be easily converted into the model checker, e.g. NuSMV utilized in this paper Cimatti *et al.* (2002). The fast conversion tool was created by Šprdlík in Java language Šprdlík (2005). Its outputs can be usually verified faster than programs directly converted into model checker languages, as described e.g. in Rausch and Krogh (1998) and Canet *et al.* (2000).
- The transfer sets can be also utilized for cross-compiling into other computer language, e.g. C or C#, see Šusta (2004).

To show advantages our approach, we also tested one more complex program, which controls home gates, by two different ways. First, we utilized the direct conversion described in Canet *et al.* (2000) and measured time that was needed for testing different CTL propositions by available implementations of SMV model checker: CMU SMV, NuSMV, and Cadence SMV.

Finally, we performed these tests the same program but now preprocessed by the transfer set conversion.

Table 1 shows that the checkers run faster for the preprocessed program. The first row

Table 1. Average ratios of speeding up

SMV Implementation	**Our method faster by**
CMU	$>$ 1000 times
CMU +*OBDD reorder*	66 times
NuSMV	189 times
NuSMV +*OBDD reorder*	5.2 times
Cadence SMV	10.9 times

presents only a rough guess because 2 tests of directly converted program were aborted due to too long runs.

The improvements are enormous because the analyzed program was written in ladder diagram language and the transfer set conversion gives good results in this case. In other situations, a speeding up effect strongly depends on programming styles and no generally valid conclusion can be drawn from Table 1.

REFERENCES

Canet, G., S. Couffin, J.-J. Lesage, A. Petit and Ph. Schnoebelen (2000). Towards the automatic verification of PLC programs written in instruction list.

Cimatti, A., E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani and A. Tacchella (2002). Nusmv 2: An opensource tool for symbolic model checking. In: *Proceeding of the 14th International Conference on Computer-Aided Verification (CAV'2002)*.

Henzinger, Thomas A., Pei-Hsin Ho and Howard Wong-Toi (1997). HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer* **1**(1–2), 110–122.

Labinaz, G., M. Bayoumi and K. Rudie (1997). A survey of modeling and control of hybrid systems. *Annual Reviews of Control* **21**, 79–92.

Mockus, A., D. Weiss and P. Zhang (2003). Understanding and predicting effort in software projects. In: *In Proceedings of the 25th International Conference on Software Engineering (ICSE 2003)*.

Rausch, M. and B. H. Krogh (1998). Formal verification of PLC programs. In: *American Control Conference, Philadelphia, PA, USA*.

Šprdlík, Otakar (2005). Formal verification of PLC programs by SMV and UPPAAL. In: *15th International Conference on Process Control 05 [CD-ROM]*. Slovak University of Technology, Bratislava.. pp. 134–1–134–7.

Šusta, Richard (2003). Verification of PLC Programs. PhD thesis. CTU-FEE Prague. avail. at http://dce.felk.cvut.cz/susta/.

Šusta, Richard (2004). Low cost simulation of PLC programs. In: *7th IFAC Symposium* *on Cost Oriented Automation COA 2004, Gatineau (Québec) Canada*. Université du Québec en Outaouais. pp. 219–224.