MODELLING AND ANALYSIS OF THE LIFT SYSTEM AS A HYBRID SYSTEM

Dominik VOŠČEK, Anna JADLOVSKÁ, Dominik GRIGĽÁK
Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic, Tel.: +421 55 602 42 14, E-mail: dominik.voscek@tuke.sk, anna.jadlovska@tuke.sk, dominik.griglak@student.tuke.sk

ABSTRACT

This paper deals with one of the challenges of cyber-physical systems, namely modelling them as hybrid systems. Specifically the paper aims to utilize hybrid systems framework onto the lift system which comes from the real laboratory lift. The mathematical model was derived using hybrid automata framework in synergy with linear temporal logic. Hybrid automata framework was used to describe continuous dynamics as well as discrete dynamics of the lift system mathematical model. Linear temporal logic was used to formally define rules according to which the lift system is constrained. The whole logic, system dynamics and constrains are then implemented within MATLAB/Simulink environment using Stateflow toolbox. Finally, the verification of the lift mathematical model is performed within chosen scenarios.

Keywords: *cyber-physical system, hybrid automata, hybrid system, linear temporal logic, lift*

1. INTRODUCTION

Today we are witnesses of cyber-physical systems (CPS) rise. CPS are defined as an integration of physical processes with computation platforms. One of the many challenges that CPS posses, described in [1] and [2] at length, is to model such a CPS within hybrid systems framework.

Hybrid automata (HA) represent natural modelling framework for the hybrid systems [3]. This is caused by modelling the systems based on the physical laws (continuous part) and logic (discrete part). One of such systems located at Department of Cybernetics and Artificial Intelligence, FEEI, TUKE is a laboratory model of the lift.

The main purpose of this paper is to utilize hybrid system framework in cooperation with the linear temporal logic (LTL). Based on this synergy a mathematical hybrid model of the lift inspired by this laboratory model is designed. The simulation model was implemented utilizing Stateflow modelling framework within MATLAB/Simulink simulation environment.

The procedure in this paper follows a methodology for modelling, analysing and controlling hybrid systems which was introduced in [4] and [5]. The methodology consists of several steps, namely determination of possible modes and transitions between them followed by assigning continuous dynamics to these modes. At this point it is possible to simulate and analyse the hybrid system.

This paper is structured as follows, at the beginning a general notion of hybrid automata is described. In the following chapter a detailed mathematical model of the lift is described and then verification of the hybrid system as a whole is performed.

2. HYBRID AUTOMATA

One of the basic mathematical representations used for the description of hybrid systems is the representation of hybrid automata, which is an extension of the finite state machines formalism for continuous dynamics in individual discrete states [6].

Hybrid automaton can be considered as a tuple $H = (Q, X, f, Init, Dom, E, G, R)$, where:

- Q is the finite set of discrete modes $\{q_1, q_2, \ldots, q_{max}\}$,
- $X \subseteq \mathrm{Re}^n$ represents state space, in which continuous dynamics of H evolves,
- $\boldsymbol{f}$: $Q \times \mathrm{Re}^n \to \mathrm{Re}^n$ is the collection of vector fields describing the continuous dynamics of the state space vector $\mathbf{x}$(t) in mode q,
- $Init \subseteq Q \times X$ is a set of possible initial hybrid states,
- $Dom : Q \to 2^X$ locates $\forall q \in Q$ a domain in the state space $\mathrm{Dom}(q) \subseteq X$,
- $E \subseteq Q \times Q$ is a set determining pairs (q_i, q_j) between which a transition is possible,
- $G(q_i, q_j)$ is a guard set, which assigns to each edge $(q_i, q_j) \in E$ a set of required continuous states for the transition,
- R: $E \times \mathrm{Re}^n \to \mathrm{Re}^n$ is a reset map which defines for each edge $(q_i, q_j) \in E$ and continuous state $\mathbf{x}(t) \in \mathrm{Re}^n$ a change of continuous state between transition from mode q_i to mode q_j [6].

The state of hybrid automaton H is then defined as a pair $(q, \mathbf{x}(t)) \in Q \times \mathrm{Re}^n$, where $\mathbf{x}(t) \in X$ and $q \in Q$.

2.1. Execution of a hybrid automaton

Time behaviour of a hybrid automaton can be described in the following steps [7]:

1. Initialization of the hybrid automaton H: $(q_0, \mathbf{x}_0) \in$ Init,
2. continuous state evaluation $\mathbf{x}(t)$ in consideration of the continuous dynamics $\dot{\mathbf{x}}(t) = \mathbf{f}(q_0, \mathbf{x}(t)), \mathbf{x}(0) = \mathbf{x}_0$ while the discrete mode remains constant $q(t) = q_0$,
3. hybrid automaton evolves according to the continuous dynamics $\mathbf{f}(q_0, \mathbf{x}(t))$ while $\mathbf{x}(t) \in \mathrm{Dom}(q_0)$,
4. if at any moment, the state space vector $\mathbf{x}(t)$ reaches guard set $G(q_0, q_1)$, then:
   - a transition from discrete mode q_0 to discrete mode q_1 takes place,
   - continuous state $\mathbf{x}(t)$ updates from current value $\mathbf{x}^-(t)$ (before transition) to value $\mathbf{x}^+(t)$ (after transition), where $(\mathbf{x}^-(t), \mathbf{x}^+(t)) \in R(q_0, q_1)$,

5. hybrid automaton evolves its continuous dynamics according to the continuous dynamics $\mathbf{f}(q_1, \mathbf{x}(t))$ and the whole process is repeated.

2.2. Hybrid automaton graphical representation

Often, it is appropriate to represent the hybrid automaton as an oriented graph. Graph nodes represent individual discrete modes and edges represent individual possible transitions between these discrete modes. An initial set, continuous dynamics and domain is assigned to each node. There is also assigned an element from a guard set and a reset map to each edge [6]. An example of such a graphical representation is shown in Fig. 1.

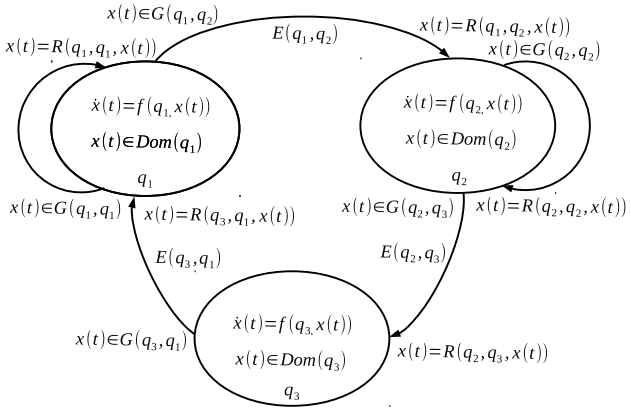

Fig. 1 An example of a hybrid automaton oriented graph

3. DESCRIPTION OF A LIFT AS A HYBRID SYSTEM

The lift system (LS), whose functionality was inspired by the actual laboratory model of the lift is described from the construction point of view in [8]. It is important to state that the LS does not exactly matches all the physical parameters of the laboratory model, e.g. height of the each floor. Moreover, some lift mechanisms were simplified, e.g. uniform rectilinear motion of the LS is considered instead of real nonlinear form with acceleration and decceleration.

The LS can be considered as a hybrid automaton as it contains dynamics that is continuous over time in some discrete modes. The functionality of the system varies according to the mode in which the lift dynamics evolves [9].

LS's main functionality consists of getting the lift to the requested floor. To operate the LS, the required sequence of the floors must be followed and then processed by the system's logic to ensure that the LS reaches the desired floor. The requests are represented by pushing the appropriate buttons. The LS can move onto 4 floors in total, therefore it contains 4 push-buttons to send a request. The active requirements are shown in the laboratory model by LED lights. In the LS, requests are sent to the system as logic inputs and active requests are considered as logic outputs.

The lift door is opened and closed automatically upon arrival to the desired floor. At the same time, the request is cleared when the desired floor is reached. If no request is active, the lift door closes and the lift is idle and waits for a request. The logic of the lift is considered as follows, when multiple requests are triggered, the lift moves to the top floor and then proceeds to the lower floors whose requirements are active. Upward movement is only possible if no lower floor requirements are active.

3.1. Logic and parameters of the lift

According to LTL nomenclature and its operators $\mathscr{G}$ for *globally*, $\mathscr{F}$ for *finally*, $\mathscr{X}$ for *next*, $\mathscr{U}$ for *until* and based on the LS description it is possible to introduce the following atomic propositions within LTL framework [10]:

- h_i the lift stays on the i-th floor,
- d_i the door on the i-th floor is open,
- r_i there is a request on the i-th floor,
- d_{so} door space occupied.

Then for the LS inspired by the real lift behaviour following LTL formulas Φ_i can be defined:

1. The doors are "safe", i.e., a floor door is never open if the lift is not staying there

$$\Phi_1 = \mathscr{G}(\bigwedge\nolimits_{i=1,2,3,4}(\neg h_i \rightarrow \neg d_i)). \tag{1}$$

2. Any requested floor will eventually be served

$$\Phi_2 = \mathscr{G}(\bigwedge\nolimits_{i=1,2,3,4}(r_i \rightarrow \mathscr{F}(h_i \wedge d_i))). \tag{2}$$

3. Upward movement is only possible if no lower floor requirements are active

$$\begin{aligned}\Phi_3 =& \mathscr{G}(\bigwedge\nolimits_{i=1,2,3,4} h_i \wedge r_{j<i} \rightarrow \\ & \rightarrow \mathscr{X}((\bigwedge\nolimits_{k=i+1,\ldots,i_{max}} \neg h_i)\mathscr{U}(h_j \wedge d_j))).\end{aligned} \tag{3}$$

4. If the door space is occupied, do not close the door, neither change the height of the lift until the door space is not occupied

$$\begin{aligned}\Phi_4 =& \mathscr{G}(\bigwedge\nolimits_{i=1,2,3,4} h_i \wedge d_{so} \rightarrow \\ & \rightarrow \mathscr{X}((\bigwedge\nolimits_{i=1,2,3,4} h_i \wedge d_i)\mathscr{U}(\neg d_{so}))).\end{aligned} \tag{4}$$

5. Eventually there will be a last request, i.e. there is a time point after which no floor is requested any more

$$\Phi_5 = \mathscr{F}\mathscr{G}(\bigwedge\nolimits_{i=1,2,3,4}(\neg r_i)). \tag{5}$$

Subsequently, the interconnection between continuous and discrete variables is stated in the following sections of this paper. At first, brief description and overview of physical quantities is stated in Tab. 1, the parameters of the lift are shown in Tab. 2.

LS is modelled as a hybrid system to illustrate transitions between individual states. While designing the LS it was necessary to create an oriented graph of transitions between system modes, with nodes representing system

modes and edges displaying transitions between modes as stated in section 2.2 based on conditions introduced in this section. The oriented graph of the LS shown in Fig. 2 shows top level modes, with each mode having own oriented graph one level deeper.

Table 1 Lift System physical quantities

Nomen.	Range definition	Description
$h(t)$	$h(t) \in < h_1; h_4 >$	lift height
$d(t)$	$d(t) \in < 0; d_{max} >$	door position
$h_{req}(t)$	$h_{req}(t) \in$	req. lift
	$\in \{h_1; h_2; h_3; h_4\}$	height
$\tau(t)$	$\tau(t) \in < 0; \tau_{max} >$	timer
$v_d(t)$	$v_d(t) \in$	door opening
	$\in \{-v_{dlin}; 0; +v_{dlin}\}$	velocity
$v_h(t)$	$v_h(t) \in$	lift movement
	$\in \{-v_{hlin}; 0; +v_{hlin}\}$	velocity

Table 2 Lift System parameters

Nomen.	Value	Unit	Description
h_{dif}	1.5	$[m]$	floor difference
h_1	0	$[m]$	1st floor height
h_2	$h_1 + h_{dif}$	$[m]$	2nd floor height
h_3	$h_1 + 2h_{dif}$	$[m]$	3rd floor height
h_4	$h_1 + 3h_{dif}$	$[m]$	4th floor height
d_{max}	1.2	$[m]$	width of one side of the door
δ_d	0.01	$[m]$	tolerance for the door position
δ_h	0.03	$[m]$	tolerance for the lift position
τ_{max}	3	$[s]$	timer reset value
v_{dlin}	1	$[ms^{-1}]$	door opening linear velocity
v_{hlin}	0.1	$[ms^{-1}]$	lift movement linear velocity

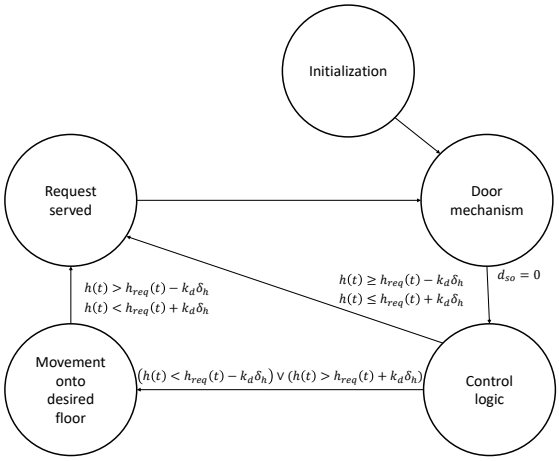

Fig. 2 Top level graphical representation of the Lift System

The transition graph shows the transitions between each major LS modes. The modes and transitions between them are described in detail in the logical sequence within this section.

3.2. Initialization mode

LS operation process starts in the *Initialization* mode. This mode can not be transitioned to from any other mode, which is emphasized by the fact that no possible transition is drawn to this mode. This mode is responsible for initializing LS, setting initial variables, resetting request values, evaluating internal variables and parameters in the LS simulation model. The initialization action is performed only once, and this mode can only be accessed upon the LS restart.

3.3. Door mechanism mode

The *Door mechanism* mode is responsible for opening and closing the door. The position of the door is denoted by the physical quantity $d(t)$, as mentioned in Tab. 1. The fully closed door has the position of $d(t) = d_{max}$, the open door has a position of $d(t) = 0$. The opening of the door is described by the relationship $\dot{d}(t) = v_d(t)$.

The door opening velocity $v_d(t)$ is considered to be positive if the door is being closed, negative while opening the door. The value of $v_d(t)$ was chosen to be constant in the simulation (with respect to the sign), thus the movement of the door is represented by linear velocity as shown in Fig. 3.

Door mechanism mode is triggered in three cases:

- After the lift reaches the desired floor h_{req},
- when the lift is on the floor where the activation button is pressed at the moment,
- if there is an obstacle within the door space at the moment of closing the door, i. e. $d_{so} = 1$.

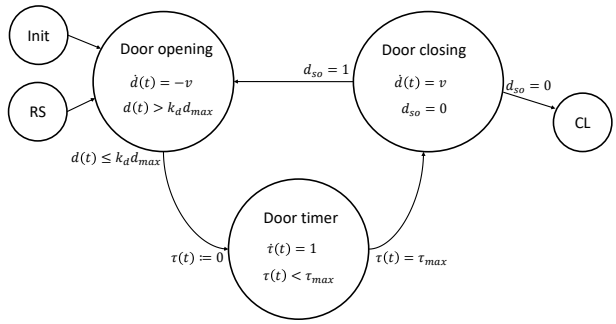

Fig. 3 *Door mechanism* mode

When the door is open, a timer $\tau(t)$ is triggered to ensure that the door is open for the time τ_{max}. The timer $\tau(t)$ is reset at the moment when the lift door is fully open. After passing the time τ_{max}, it is then assessed whether there is an obstacle in the door. If there is not, i. e. $d_{so} = 0$, the door is being closed. If there is an obstacle in the door space, i. e. $d_{so} = 1$, the door remains open and the timer $\tau(t)$ is reset, i. e. $\tau(t) = 0$. If there is an obstacle at the door space when closing the door, or if there is an obstacle in the door space after the maximum time τ_{max} has elapsed, the door remains open and the timer $\tau(t)$ starts counting again.

After closing the door the transition from the *Door mechanism* mode to the *Control logic* mode occurs.

3.4. Control logic

The *Control logic* mode is responsible for assessing the floor on which the lift should appear. Thus, it determines the value of $h_{req}(t)$. LS works on the top-down collection principle as stated in (3). If all requirements are active, the lift comes to the top floor. Subsequently, it gradually gets to the floor below, so it passes through all the floors until it reaches the lowest desired floor, in this case the first floor. The lift is only allowed to move to the higher floors, if only there is no active request on the lower floors or if the lift is already located on the lowest floor. When moving the lift down and simultaneously activating the higher floor requirement, the lift continues to move downwards and ignores all the above requirements until the LS clears all the requirements on the lower floors.

3.5. Movement onto desired floor

This mode is active only if there exist a request to drive the lift to a desired floor h_{req}. It is also necessary to mention that the lift is not present on the floor where the request exists. In such a case the lift is not expected to move. Also, the *Request served* (RS) mode is activated and then the system takes a transition to *Door mechanism* (DM) mode as shown in Fig. 4.

Let us therefore consider the case that there is just one active request of different height $h_{req}(t)$ than the current height of the lift $h(t)$. If the lift is above the required height $h_{req}(t)$, i. e. $h(t) > h_{req}(t)$, it follows that the downward movement of the lift occurs, i. e. $v_h(t) = -v_{hlin}$. Lift movement is defined by the linear function. The movement of the lift upwards and downwards is a uniform rectilinear motion, the speed of the lifts movement v_{hlin} is in the direction of the desired floor $h_{req}(t)$. The lift then begins to descend to the desired floor at the speed of the v_{hlin}, and after reaching the height $h(t) < h_{req}(t) + \delta_h$, the lift stops, i.e. $v_h(t) = 0$, and the LS takes a transition to *Request served* mode.

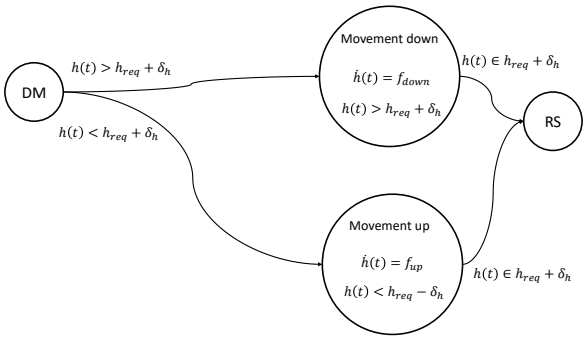

Fig. 4 *Movement onto the desired floor* mode

Then consider the case that the lift is below the required height $h_{req}(t)$, i. e. $h(t) < h_{req}(t)$. The upward movement of the LS, which is again a uniform rectilinear motion, causes the lifts to move with the speed $v_h(t) = v_{hlin}$. Movement is carried out until the height of the lift reaches the desired floor level, i. e. $h(t) > h_{req}(t) - \delta_h$ and LS takes a transition to *Request served* mode.

3.6. Request served

This mode is responsible for clearing active LS requests and can occur in two cases, namely:

- after activating the floor requirement on which the lift is currently located,
- after the lift has reached the desired floor.

If the floor requirement is cleared the LS enters the *Door Mechanism* mode. Each time only one request is served. The oriented graph is depicted in Fig. 5.

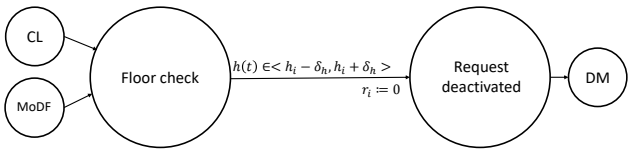

Fig. 5 *Request served* mode

4. VERIFICATION OF THE LIFT SYSTEM

The design of the entire LS logic and its functionality was implemented in the MATLAB/Simulink simulation environment using the Stateflow toolbox. It offers a number of options in working with hybrid systems and finite state automata [11].

The Stateflow toolbox allows creation of blocks that represent the modes of the system. Subsequently, these

blocks are linked by one-way arrows that determine the transition of the system from one mode to the another. In the individual system modes, actions are being performed either upon entering the mode, while staying in the mode or during exiting the mode. Furthermore, it is possible to determine the transition conditions, i. e. in which case there is a possible transition from one mode to another. Upon the system transition, it is possible to determine the execution of the action, which is executed during this transition.

From the aforementioned functionality, it is possible to state that the possibilities offered by the Stateflow toolbox allow the modelling of hybrid systems because it is easy to implement modes and transitions between these modes. Modes and transitions within hybrid system can be directly rewritten into the Stateflow toolbox by state blocks and arrows for state transitions, with state and mode concepts being considered as equivalent ones.

The simulation scheme of LS in the Simulink environment using a block named *Lift System* is shown in Fig. 6. This scheme shows the individual logic inputs to the lift block, which represent the activation of the lift buttons on the each floor. *DoorSpaceOccupied* is an logic input which represents the obstacles within the door space, i. e. the value d_{so} at a given time. At the output of the scheme are all needed variables to monitor the whole functionality of the LS.

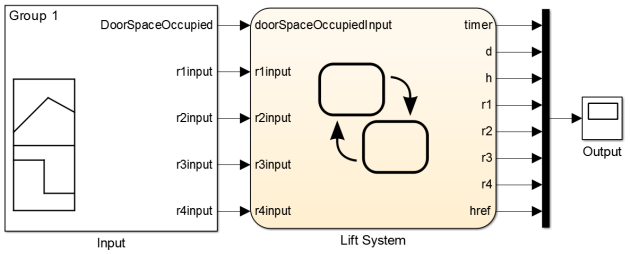

Fig. 6 Stateflow scheme of the Lift System

4.1. Verification of the Request Processing

As mentioned in the section 3, the LS works on the principle of top-down collection. The purpose of this scenario is therefore to verify the performance of the lift in the presence of several active requirements at the same time for calling the LS on a given floor. In parallel, the LS behaviour is monitored when activating the requirement on the upper floor while there are still active requirements on the lower floors.

Inputs for the LS's request activation are considered to last a short time, about *0.5s*, which represents the real push of the LS button. These inputs enter the LS block and are temporarily stored as active until they are deactivated in the *Request served* mode. Fig. 7 shows the individual active requirements, their deactivation and the desired lift height h_{req}.

At time $t = 0s$, all the requirements are brought to the input, so all the push buttons are pressed together at once. In the LS, it is evaluated that the lift should appear on the first floor with the height h_1 if it is not on this floor. In the first seconds, the lift door is opened and closed, then the height of the top floor h_4 is chosen as the height required, i.e. $h_{req}(t) = h_4$.

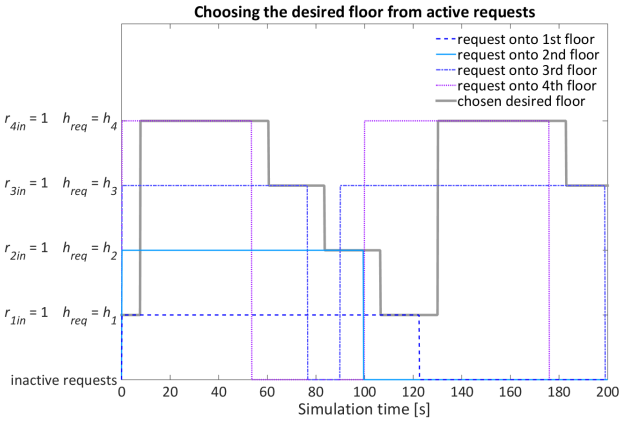

Fig. 7 Processing of the active requerements

The lift then moves towards this floor, which can be seen in Fig. 7. When the height $h(t) = h_4$ is reached, the request r_4 is deleted, the lift door opens and closes again and the new required height $h_{req}(t) = h_3$ is selected. This continues with the lower floors h_2 and h_1 until all lower floor requirements are deactivated. It is therefore verified that the lift serves requirements in the top-down manner.

The system does not select the required height $h_{req}(t)$ as the height of the highest floor with the active requirement even if the lift entry meets this requirement and the requirements on lower floors are still active. Activation of the *3*rd floor requirement occurred at time $t = 90s$ and activation of the *4*th floor arrival occurred at time $t = 100s$. These requirements did not affect the movement of the LS on the *1*st floor. Subsequently, at the time $t = 130s$, a new required lift height $h_{req}(t)$ was chosen as the height of the highest floor with the active requirement, in this case $h_{req}(t) = h_4$. The fact that the $r_{3_{in}}$ floor requirement 3 occurred earlier than activating the $r_{4_{in}}$ floor requirement does not affect the choice of the *4*th floor as the desired floor.

4.2. Lift System Movement Verification

This subsection contains a verification of the lift movement based on the changes of the desired floor h_{req}. The change of height $h(t)$, depending on the change of the actual desired height $h_{req}(t)$, is shown in Fig. 8. For the first $200s$ of the simulation, the required height $h_{req}(t)$ has the same course as that shown in Fig. 7. The LS always reaches the desired floor. The time during which the lift stays on the floor is the time it takes to open and close the door. At the time approximately $t = 200s$, the LS remains on the same floor due to the fact that no requirements are active. Approximately at the time $t = 250s$ the LS stays on the given floor due to the presence of the obstacle. The lift leaves this floor only after removing the obstacle from the door, which is simulated by setting the logic input $d_{so} = 0$.

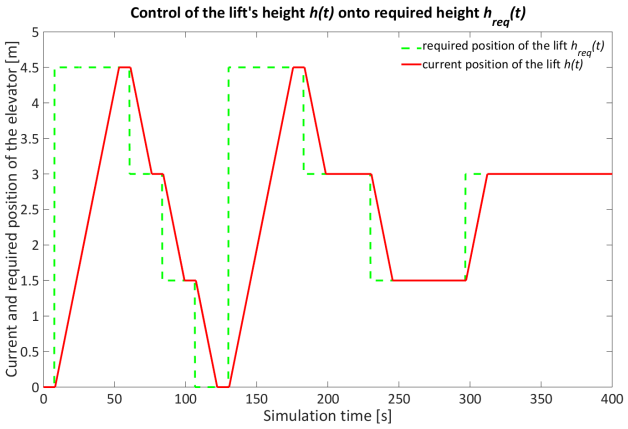

Fig. 8 Movement of the lift according to required height

4.3. Door Mechanism Verification

Three signals were monitored when verifying the functionality of the door mechanism, namely the position of the door $d(t)$, the obstacle in the door space d_{so} and the timer $\tau(t)$. Door mechanism scenario is displayed from the time $t = 220s$ within the course of the scenario for the LS movement verification shown in Fig. 8. The time behaviour of each of the three signals is shown in Fig. 9.

The obstacle was found in the door at the time $t = 245s$ immediately after the door was opened and was removed at the time $t = 290s$. The existence of the obstacle in the door space can only be introduced in the case of the open door, which is also valid within the LS simulation model.

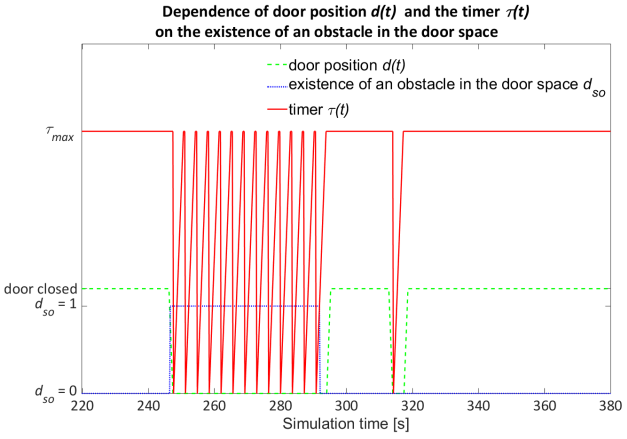

Fig. 9 Door functionality with the obstacle within the door space

At the time approximately $t = 245s$, the door opens until it is fully opened. At the moment of their opening, the timer $\tau(t)$ is set to zero, $\tau(t) = 0$. Subsequently, the timer value increases with time until it reaches the value τ_{max}. At the timer value $\tau(t) = \tau_{max}$, it is verified that there is an obstacle in the door. If so, the timer $\tau(t)$ will reset to zero again and a new countdown will begin. This process is repeated until the obstacle is removed from the door. At the moment when the timer reaches $\tau(t) = \tau_{max}$ and the obstacle in removed from the door space $d_{so} = 0$, the door is closed. At approximately $t = 310s$ to $t = 320s$, it is possible to see opening of the door, timer reset, and door closure in the usual way. During this process the LS remains on the requested floor h_{req}.

5. CONCLUSION

In this paper we examined one of the CPS challenges, which is hybrid modelling of such systems. This challenge was applied within the lift system whose representation came from the laboratory model. The system captures both the continuous and the discrete dynamics as well as the constrains under which the lift system is defined. These constrains were formulated within linear temporal logic framework.

After derivation of a mathematical model, a simulation model was implemented in MATLAB/Simulink environment using Stateflow toolbox. Stateflow toolbox met all the criteria to model such a hybrid system and various case-studies were conducted in order to verify the derived mathematical model of the lift system. It could be observed that the system met all the criteria given by linear temporal logic which encompasses safety requirements and continuous operation of the LS.

Since the LS served to verify the proposed methodology combining hybrid automata framework with linear temporal logic, one of the possible enhancements of the LS is to make it more user oriented, e.g. online changing the simulation parameters such as change of the desired floor, open door time etc.

Another challenges within modelling of cyber-physical systems can be further studied and analysed based on this lift system such as more complex hybrid systems with tighter constraint as well as control design of such systems.

ACKNOWLEDGEMENT

This publication is the result of the Project implementation: University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology - II. phase, ITMS: 313011D232 (60%) and "Development of the Center of Information and Communication Technologies for Knowledge Systems", ITMS: 26220120030 (10%), both supported by the Operational Programme Research & Development funded by the ERDF, grant TUKE FEI-2015-33: Research Laboratory of Nonlinear Underactuated Systems (30%).

REFERENCES

[1] WAN, J. – YAN, H. – SUO, H. – LI, F.: *Advances in cyber-physical systems research*, KSII Transactions on Internet and Information Systems **5**, No. 11 (2011) pp. 1891–1908, ISSN 1976-7277.

[2] VOŠČEK, D.: Cyber-Physical Systems, Modelling Framework and Application into Distributed Control Systems, SCYR 2016, Košice TU, pp. 88–91, ISBN 978-80-553-2566-8.

[3] YANG, Y. – ZHOU, X.: Cyber-physical systems modeling based on extended hybrid automata, In: Fifth International Conference on Computational and Information Sciences (ICCIS), IEEE, 2013, pp. 1871–1874, ISBN-13: 978-0-7695-5004-6.

[4] VOŠČEK, D. – JADLOVSKÁ, A.: Modelling and control of a cyber-physical system represented by hydraulic coupled tanks, In: 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI), IEEE, 2017, pp. 439–444, ISBN 978-1-5090-5654-5.

[5] VOŠČEK, D.: Contribution to hybrid models of cyber-physical systems and their implementation into distributed control system, Dissertation prospectus, 2017, pp. 69.

[6] LYGEROS, J. – JOHANSSON, K. H. – SIMIC, S. N. – ZHANG, J. – SASTRY, S. S.: *Dynamical properties of hybrid automata*, IEEE Transactions on automatic control **48**, No. 1 (2003) pp. 2–17, ISSN 0018-9286.

[7] SCHUTTER, B. D. – HEEMELS, M.: Modeling and control of hybrid systems, Lecture notes, TU Delft, 2015, http://www.dcsc.tudelft.nl/ sc4160/

[8] FERENČÍK, N. – JADLOVSKÝ, J. – KOPČÍK, M. – ZOLOTOVÁ, I.: PI control of laboratory model lift via ladder logic in PLC, In: 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI), IEEE, 2017, pp. 383–386, ISBN 978-1-5090-5654-5.

[9] GRIGĽÁK, D.: Modelling, Analysis, Control and Simulation of Hybrid Systems, Bachelor thesis (supervisor Anna Jadlovská), Košice, 2017.

[10] VARDI, M. Y.: An automata-theoretic approach to linear temporal logic, In: Logics for concurrency, Springer, Berlin, Heidelberg, 1996, pp. 238–266, ISBN 978-3-540-49675-5.

[11] https://www.mathworks.com/help/stateflow/

Received September 18, 2017, accepted November 9, 2017

BIOGRAPHIES

Dominik Vošček was born in August 29th, 1991. In 2015 he graduated (MSc.) with honors and dean's prize at the Department of Cybernetics and Artificial Intelligence of the Faculty of Electrical Engineering and Informatics at Technical University of Košice. Since September 2015 he has been internal PhD. student at the Department of Cybernetics and Artificial Intelligence. The topic of his dissertation thesis is focused on cyber-physical systems, specifically at their modelling as hybrid systems. In addition, the range of his interest is physical dynamic system modelling, optimal control algorithms design and their verification on simulation and laboratory models within MATLAB/Simulink simulation environment.

Anna Jadlovská was born in October 29th, 1960. She received her MSc. degree in the field of Technical Cybernetics at the Faculty of Electrical Engineering of the Technical University of Košice in 1984. She defended her PhD thesis in the domain of Automatization and Control in 2001 at the same University; her thesis title was Modelling and Control of Non-linear Processes Using Neural Networks. Since 1993 she worked in the Department of Cybernetics and Artificial Intelligence Faculty of Electrical Engineering and Informatics Technical University of Košice as an Associate Assistant and since 2004 she has been working as an Associate Professor. Her main research activities include the problems of adaptive and optimal control in particular predictive control with constraints for non-linear processes using neural networks and methods of artificial intelligence (Intelligent Control Design) which are also tested on the laboratory models within MATLAB/Simulink simulation environment. She is the author of scientific articles and contributions to various journals and international conference proceedings, as well as being the co-author of some monographs.

Dominik Grigľák was born in March 29th, 1995. He studies at the Department of Cybernetics and Artificial Intelligence of the Faculty of Electrical Engineering and Informatics at Technical University of Košice. In 2017, he received his bachelor degree. His bachelor thesis was focused on modelling, analysis and controlling of hybrid systems.