Statistical Model Checking of Human-Robot Interaction Scenarios

Livia Lestingi[†], Mehrnoosh Askarpour[*], Marcello M.Bersani[†], Matteo Rossi[†]

[†] Politecnico di Milano, `{firstname}.{lastname}@polimi.it`

[*]McMaster University, `askarpom@mcmaster.ca`

Robots are soon going to be deployed in non-industrial environments. Before society can take such a step, it is necessary to endow complex robotic systems with mechanisms that make them reliable enough to operate in situations where the human factor is predominant. This calls for the development of robotic frameworks that can soundly guarantee that a collection of properties are verified at all times during operation. While developing a mission plan, robots should take into account factors such as human physiology. In this paper, we present an example of how a robotic application that involves human interaction can be modeled through hybrid automata, and analyzed by using statistical model-checking. We exploit statistical techniques to determine the probability with which some properties are verified, thus easing the state-space explosion problem. The analysis is performed using the Uppaal tool. In addition, we used Uppaal to run simulations that allowed us to show non-trivial time dynamics that describe the behavior of the real system, including human-related variables. Overall, this process allows developers to gain useful insights into their application and to make decisions about how to improve it to balance efficiency and user satisfaction.

1 Introduction

Robots have been mostly used in industrial environments. Indeed, manufacturing lines that feature robotic manipulators, as well as mobile robots in automated warehouses, are very common nowadays. Nevertheless, these applications usually involve strong workspace separation between humans and robots (e.g., through cages or dedicated lanes). Over the last few years, thanks to the phenomenon called Industry 4.0, a lot of effort has been put into trying to inject human-robot collaboration into industrial contexts. On the other hand, robots employed in non-industrial settings still represent extraordinary cases despite their massive potential. Their deployment could, for example, relieve health professionals from the most repetitive tasks, such as transporting objects or delivering paperwork, so that they would have more time for duties which are exclusive to humans, i.e., taking care of patients. Similarly, robots could provide support in disaster relief situations, for instance when buildings at risk of collapse need to be inspected.
Before users can welcome personal assistant robots into their daily routines, they require strong guarantees about the fact that these machines will not cause any harm nor any damage. The issue of safety in service robotics has already been tackled in previous works, firstly by standards, as assessed by Virk et al. [19] (most notably ISO:13482 [12]). The challenges of minimizing risk for service robots, as argued by Jacobs et al. [14], are due to the diversity of users that this technology reaches, and to the fact that most applications imply intended contact.
However, safety is not the only priority. It is also paramount that users perceive the addition of robots as an improvement rather than a step back. This involves building trust between the human and the machine (e.g., by making the robot able to transparently share its status), and assuring that the robot plan does not prevail over human needs. This is especially important when users in delicate conditions are involved,

such as patients who may be too tired or in discomfort to keep up with the robot's actions.
We have selected a relevant scenario where human-robot interaction is the most relevant aspect to be considered. A human and a battery-powered mobile robot are the main elements of the scenario. The novelty lies in the fact that non-traditional variables pertaining to human mental and physical conditions are embedded in the model. Given the complex dynamics of the system, the most appropriate choice is to develop the formal model as a network of Hybrid Automata. The modeling tool is Uppaal [4] and its extension Uppaal SMC [6]. Statistical Model Checking (SMC) assesses the probability of occurrence of certain critical events based on time-bounded runs of the system. In particular, in this paper we explore the likelihood of the human reaching full exhaustion while the robot is still moving: this highlights the need of robot decision-making policies that are aware of human needs. In the future, we envision the development of features that make the scenario easily customizable and verifiable by users who possess a technical background though not on formal methods. This could prove substantial, for example, to healthcare specialists willing to estimate the outcome of a real application case.
The rest of the paper is organized as follows: Section 2 surveys related works existing in literature; Section 3 outlines the background of the work; Section 4 introduces in detail the selected case study and Section 5 presents the main contribution of the paper, i.e. the hybrid automata; Section 6 reports the experimental results; finally, Section 7 draws some conclusion.

2 Related Work

The literature features a number of works on the use of timed automata to model complex systems. Molnar et al. [17] present a layered modeling architecture where agents correspond to hybrid systems interacting with each other. Wang et al. [20] [16] explore the possibility of modeling a simple robotic application using the Uppaal tool. They model every robot component as a set of automata that synchronize with each other to perform the mission. Once the model formally satisfies a set of properties, they automatically generate executable C++ code. Similarly, Halder et al. [11] model a mobile Kobuki robot ROS-based application in Uppaal, and subsequently perform model-checking to verify relevant properties about communication between nodes, e.g., that loss of information never occurs. The approach presented by Aniculaesei et al. [2] shows an approximation of the environment by means of static and dynamic obstacles, that might obstruct the movement of a mobile robot. In this case, Uppaal is used to verify collision-avoidance conditions. The work by Zhou et al. [22] is centered on motion planning, where the mission is specified with an MITL formula from which an automaton is generated to find feasible runs. Model-checking techniques are also applied to formal models of robotic systems. Webster et al. [21] analyze the case study of a personal robotic assistant operating in a house-simulating laboratory. High-level rules are translated into a Brahms workframe, which is subsequently translated into the language used by the SPIN model-checker. The work by Arai and Schlingloff [3] implements statistical model checking to evaluate the performances of autonomous transport robots in an industrial plant. In their case, the verification process aims at finding the optimal value of certain system parameters, e.g., how many robots are needed to complete the mission. Despite the intrinsic limitations of simulation, they manage to obtain results exhaustive enough so that stakeholders can make their decisions. Statistical model checking is also pivotal in the works by Foughali et al. [7] [8]. The authors compare different formalisms to model and formally verify concurrency [7]. In particular, they select Timed Transition Systems to formalize such components and perform statistical verification of the model of a quadricopter [8]. In this case, relevant properties involve the machine's capability to respond to requests in an orderly and consistent fashion to avoid accidents.

Despite model-checking receiving some attention over the last few years, most of the works applying formal verification approaches to robotics tend to focus on the robot itself and its internal structure. In particular, formal verification mostly deals with internal safety and integrity properties. Verifying this type of property holds major importance, for example, when the product is undergoing the certification process to be released into the market. On the other hand, applications that heavily feature human-interaction also require higher-level models and the verification of properties of a different nature. In this paper, we present an example of this type of model that mostly focuses on human-related aspects.

3 Background

In this Section, we introduce the bases on which the model of the scenario has been built. The selected formalism, as mentioned in Section 1, is that of *hybrid automata*. Timed automata are built on the simplest time dynamics [9]: each model features specific variables called *clocks*, whose value increases linearly with time unless they are reset. Hybrid automata [1] can be considered an extension of the aforementioned formalism since they support generic time dynamics. As per the semantics defined by [1], each automaton is composed of a set of *locations*. Each location can be endowed with constraints, called *flow conditions*, that determine how variables evolve with time. The switch from one location to another is realized through *transitions*. If a transition is endowed with a *guard* condition, it may be fired only when the condition is true. In some cases, it is useful to model transitions that only fire once an event e occurs: these are labeled by $e!$ and $e?$ if the transition respectively triggers the event e or awaits its occurrence. These labels are also called *channels* [4] since they allow the synchronization of concurrent systems in a way similar to the publish/subscribe pattern.
Hybrid automata are useful to model systems whose dynamics can be described by a set of non-elementary ODEs (Ordinary Differential Equations). In our specific case, ordinary timed automata would not have proven sufficiently expressive to capture either robot or human behaviors that possess complex dynamics. For example, hybrid automata are well-suited to capturing human-related variables in the system, such as their fatigue. The adopted model of human fatigue $F(t)$ is the one proposed by Konz [15] [13], which can also be found in eq.(1).

$$F(t) = \begin{cases} 1 - e^{-\lambda t} & \text{(a)} \\ e^{-\mu t} & \text{(b)} \end{cases} \tag{1}$$

Eq.(1)(a) represents how the level of fatigue increases (to a maximum value of $F = 1$) with time t while walking, and eq.(1)(b) how it decreases with time while resting ($F = 0$ means full recovery). Parameters λ and μ determine the duration of full fatigue accumulation/recovery cycles: if they have a low value, fatigue accumulation / recovery will be slow [13]. Finally, if $F(t_{max}) = 1$, t_{max} is called Maximum Endurance Time (MET) [13]. As for the robot, we assume it is powered by a lithium battery with a typical charge/discharge profile [18]. The battery charge is set to 100% at the beginning of the simulation, then it decays rapidly until 80% (with rate r_1) and between 20% and 0% (rate r_3), whereas the discharge rate decreases significantly in the nominal zone (rate r_2), i.e., between 80% and 20%.

4 Human-Robot Interaction Scenario

The case study involves a robot in a healthcare setting leading a newly registered patient to their room or a doctor's office. The actors, also depicted in Figure 1, are a battery-powered mobile robot and a human, and the synchronization is realized by having the robot lead the human along a hallway.

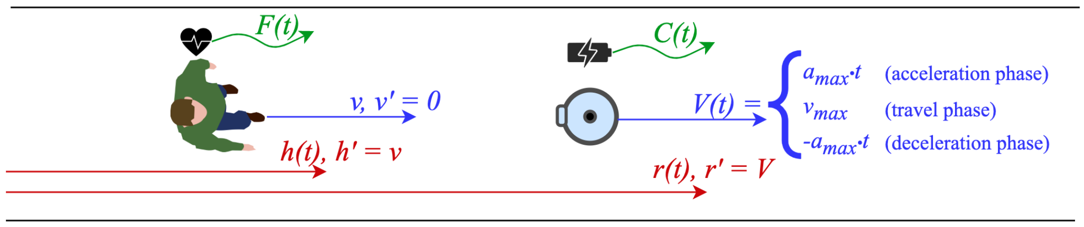

Figure 1: Diagram representing the scenario: green arrows $F(t)$ and $C(t)$ represent human fatigue and battery charge; red arrows $h(t)$ and $r(t)$ are the distances covered respectively by human and robot; blue arrows represent the constant speed of the human v and the robot's speed $V(t)$ which follows a trapezoidal profile.

In this scenario, the robot can start moving, stop, and start recharging. We also assume that the robot can recharge regardless of its position, and stop recharging when the battery charge is back to its full. In real life, the decision to start and stop could be caused by several factors, such as a sudden obstacle, which are not captured by the model. The simplest way to simulate this behavior is to assume that the probability to leave the *idle* state has an exponential distribution $1-e^{-\lambda t}$. As a consequence, the robot is more likely to take action as time passes and quicker as λ increases.
Since we are focusing on the interaction between the human and the robot rather than between the agents and the environment, assuming that the hallway has infinite length does not undermine the efficacy of the model. For the human, we assume they walk with constant speed v. The robot, instead, displays a trapezoidal velocity profile, typical in robotic applications [5], which identifies three phases: the acceleration phase, where acceleration is constant and velocity (V in Figure 1) increases linearly with time; the travel phase, where acceleration is 0 and velocity is constantly equal to its maximum; the deceleration phase, equivalent to the first phase but with negative acceleration. Human fatigue and battery charge (F and C in Figure 1) are modeled as explained in Section 3.
In this setting, with the specified boundaries, some critical events can occur that require verification. The aspect that we are most interested in analyzing is the likelihood of the human passing out due to over-exhaustion while the robot is moving, thus causing the failure of the mission.

5 Hybrid Automata Model

The complete model of the agents described in Section 4 is composed of the three hybrid automata depicted in Figure 2, which will now be presented in detail.
The automaton modeling the robot's behavior can be found in Figure 2(a). Variables r and V model the distance covered by the robot and its velocity described in Section 4, while constants $\mathsf{a}_{\mathsf{max}}$ and $\mathsf{v}_{\mathsf{max}}$ (also in Figure 1) correspond to maximum acceleration and velocity. In the initial state *idle*, the robot is not moving, hence its flow condition is $\dot{V}=0, \dot{r}=0$. From the initial state, the robot can fire two transitions, triggering two events: `start_recharging` and `start_moving`. In this preliminary version of the model, the choice is non-deterministic. In the first case, the automaton enters location *recharging* and the flow conditions remain unvaried. This location can be left only when the event `full_battery` is triggered. In the second case, the robot starts moving, hence the velocity profile described in Section 4 is initiated. The automaton first enters the location *starting*, i.e., the acceleration phase, where the following holds: $\dot{V}=\mathsf{a}_{\mathsf{max}}, \dot{r}=V$. When guard $V \geq \mathsf{v}_{\mathsf{max}}$ is true, the robot switches to location *moving*, corresponding to the travel phase, where $\dot{V}=0, \dot{r}=\mathsf{v}_{\mathsf{max}}$. When the robot fires the event `stop_moving`, it switches to location *stopping*, corresponding to the deceleration phase, hence $\dot{V}=-\mathsf{a}_{\mathsf{max}}, \dot{r}=V$. This phase is completed when guard $V \leq 0$ is satisfied, thus the robot returns to its *idle* state. Locations

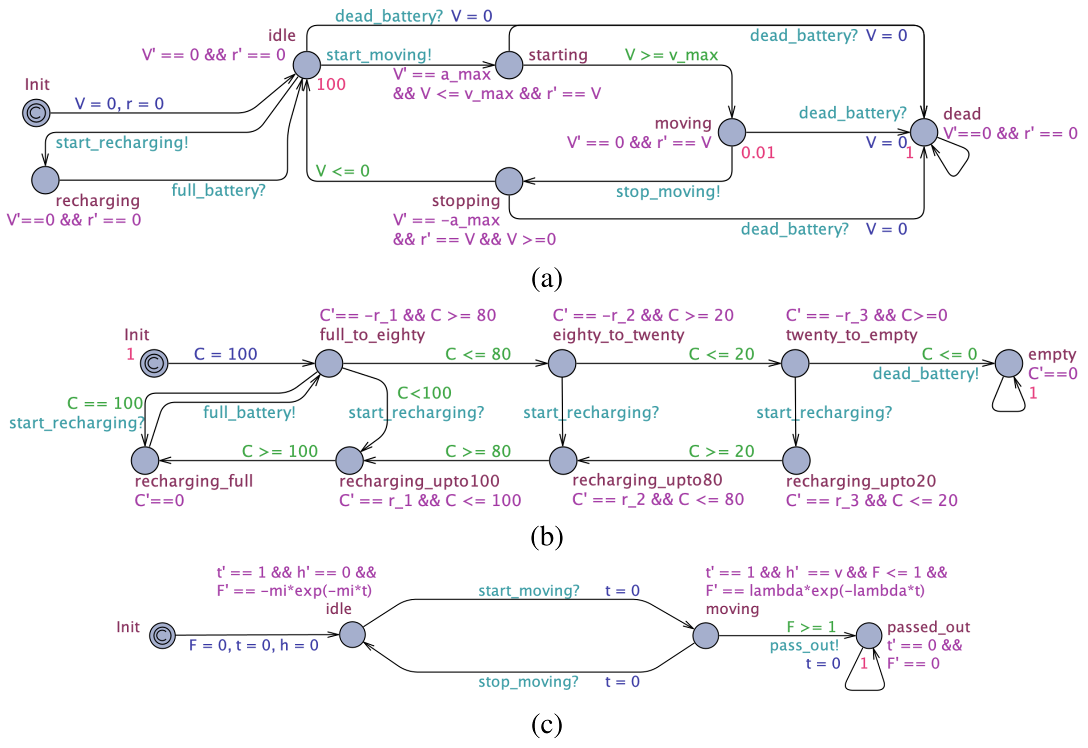

Figure 2: Hybrid Automata modeling: (a) robot, (b) robot battery, (c) the human.

starting and *stopping* are endowed with invariants $V \leq \mathsf{v_{max}}$ and $V \geq 0$, so that the outgoing transitions are respectively fired when V is exactly equal to $\mathsf{v_{max}}$ and 0. All locations, except for *recharging*, have an additional outgoing transition towards the deadlock state *dead*, which fires when event `dead_battery` occurs. In this state, the robot is forced to stop, hence V is reset to 0 and $\dot{V} = 0, \dot{r} = 0$.
The automaton corresponding to the battery, in Figure 2(b), captures the charge/discharge cycles described in Section 3. The time-dependent variable C models the level of charge, which is set to 100 in the initial state *full_to_80*. The three discharge cycle phases are identified by as many locations (*full_to_80*, *80_to_20*, and *20_to_empty*). There is a deadlock state *empty* that captures the case in which the battery is fully discharged and human intervention is required to make the robot operational again. The switch from one location to the next takes place when $C = \mathsf{C_{th}}$, with $\mathsf{C_{th}}$ respectively equal to $80, 20$, and 0. This is achieved through the invariant $C \geq \mathsf{C_{th}}$ on the starting location and the guard $C \leq \mathsf{C_{th}}$ on the outgoing transition. The last transition from *20_to_empty* triggers the event `dead_battery`, causing also the robot automaton to enter its deadlock state. The time-dynamics of C are constrained by flow conditions on the above-mentioned locations, respectively $\dot{C} = -\mathsf{r}_1, \dot{C} = -\mathsf{r}_2, \dot{C} = -\mathsf{r}_3$ and $\dot{C} = 0$ in the *empty* state. The locations corresponding to the recharge cycle (*recharging_upto20*, *recharging_upto80*, *recharging_upto100* and *recharging_full*) have a dual behavior with flow conditions $\dot{C} = \mathsf{r}_3, \dot{C} = \mathsf{r}_2, \dot{C} = \mathsf{r}_1$ and $\dot{C} = 0$. While recharging, the passage from one location to the next takes place when $C = \mathsf{C_{th}}$ with: $\mathsf{C_{th}}$ equal to $20, 80, 100$, invariants in the form $C \leq \mathsf{C_{th}}$ and guard conditions $C \geq \mathsf{C_{th}}$. When location *recharging_full* is reached, the *urgent* channel [4] `full_battery` is immediately triggered, so that the robot automaton also leaves its *recharging* location. The transition from the location of a discharging phase to its corresponding recharging one fires when the robot triggers the event `start_recharging`.
Finally, Figure 2(c) shows the automaton that models the human. In this case, the variables with non-

trivial time-dynamics are the covered distance h and the fatigue F, whose model has already been described in Section 3. A clock t, whose value grows linearly with time unless it is reset, is also required to keep track of active and resting time frames duration. When the automaton is in its initial state *idle*, we assume the human is in its resting phase, hence the flow conditions are $\dot{F} = -\mu e^{-\mu t}, \dot{h} = 0$. When the robot triggers the event `start_moving`, the human switches to its *moving* location and the clock t is also reset. In this case, since fatigue increases with time, the flow condition imposes $\dot{F} = \lambda e^{-\lambda t}$ and $\dot{h} = v$, as the distance increases as the human follows the robot. This location is also endowed with the invariant $F \leq 1$: once this condition becomes false, it means the fatigue has reached the maximum tolerable value and the human enters its deadlock state *passed_out*, where $\dot{F} = 0$. The switch from location *moving* back to *idle* occurs when the robot triggers the event `stop_moving` and the human goes back to rest.

6 Experimental Results

6.1 Verification and Simulation

The hybrid automata presented in Section 5 have been implemented in the Uppaal tool [4] and subjected to SMC experiments [6]. Given the parametric model, we assign the values justified in the following. Having established that 1 time unit corresponds to $1s$ and choosing the TurtleBot2[1] specifications as reference, we set $\mathsf{v}_{\mathsf{max}} = 65cm/s$ and $\mathsf{a}_{\mathsf{max}} = 50cm/s^2$. We also assume that the human moves at the same speed, although lower than their average, since they need to follow the robot ($\mathsf{v} = 65cm/s$). To comply with a real mobile robot charge life, the chosen battery parameters are $\mathsf{r}_1 = 0.035, \mathsf{r}_2 = 0.008, \mathsf{r}_3 = 0.055$, so that the resulting battery life is approximately $2.5h$. Finally we set $\lambda = \mu = 0.005$ (eq.1), which would mean full recovery/exhaustion after about $15min$ of resting/walking.

As mentioned in Section 4, the property to verify, formally expressed in eq.2, is that the automaton representing the human will *eventually* ($\diamond$) enter location *passed_out* while the robot is moving, hence while the automaton is either in location *starting* or *moving*. Classical model-checking properties such as *deadlock* are subsumed by 2.

$$P_{\leq t_s}[\diamond \ \ passed\_out \wedge (starting \vee moving)] \tag{2}$$

The tool allows us to evaluate the probability of the property being verified in the bounded interval $[0, t_s]$, which determines the timespan of simulations. By running the verification [2] for monotonically increasing values of t_s, we analyse how the probability evolves with time.
Figure 3(a) shows that the probability of the patient passing out is negligible for the first $5min$, it increases with time and finally crosses the 90% threshold after $2h$. This outcome is only apparently inconsistent with the $15min$ Maximum Endurance Time (MET). Indeed, this value of MET refers to intervals of non-stop walking, whereas in our model the robot can stop during operation and this gives the patient time to rest. An example of simulation is given in Figure 3(b) and (c), where the patient is fully exhausted at around time 5000 after walking for about $16min$ without stopping. The simulation also gives us some insights on the robot behavior and its battery, whose charge (the green line in Figure 3(b)) coherently decreases during movement and increases when the robot stops moving and starts recharging.

[1] https://www.turtlebot.com/turtlebot2/

[2] On a MacOs 10.14 machine with 8GB of RAM and 2 cores, running the whole experiment takes about 10min.

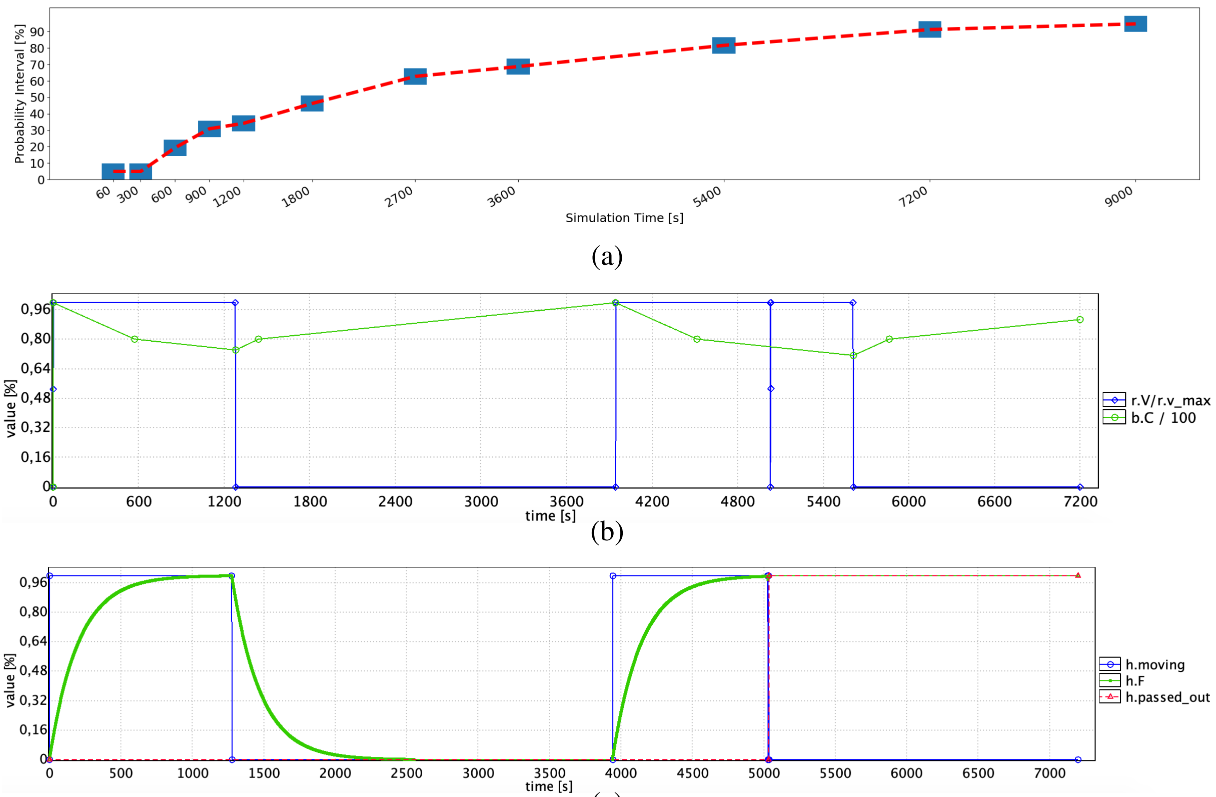

Figure 3: (a) shows the probability ranges (blue bars) calculated by Uppaal over increasing values of time, while the red line connects the average points of such intervals; (b) and (c) provide an example of system simulation in which the mission fails (the human passes out after about $5000s$): in (b) we show robot-related variables, and in (c) human-related ones.

6.2 Beyond Verification

Were stakeholders to synthesize a robot controller for this case study, the results of this experiment should be taken into account by adding a fatigue monitoring loop as a decision-making layer for the robot. The core idea is to trigger the events `start_moving` and `stop_moving` only when the value of fatigue belongs to an acceptable range: if the pair is moving and the patient becomes excessively tired, the robot should stop and restart when the human has had time to rest. In reality, this could be achieved by monitoring the distance between the robot and human: if this value crosses a certain threshold than the human is likely not being able to keep up. Alternative solutions could involve the use of a wearable device that can monitor the heartbeat or machine learning techniques to estimate human fatigue in real-time [10].

7 Conclusions

We have presented a high-level modeling approach of a human-robot interaction case study. By building upon these bases, we intend to add to the model a robot controller that takes into account non-traditional human-related factors. Further future refinements include additional interaction patterns and a realistic floor plan layout. Once the framework has reached a suitable stage of development, it will be tested in a real environment to verify the effectiveness of the synthesized controllers.

References

[1] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis & Sergio Yovine (1995): *The algorithmic analysis of hybrid systems*. *Theoretical computer science* 138(1), pp. 3–34, doi:10.1016/0304-3975(94)00202-T.

[2] Adina Aniculaesei, Daniel Arnsberger, Falk Howar & Andreas Rausch (2016): *Towards the verification of safety-critical autonomous systems in dynamic environments*. *arXiv preprint arXiv:1612.04977*, doi:10.4204/EPTCS.232.10.

[3] Ryota Arai & H Schlingloff (2017): *Model-based performance prediction by statistical model checking an industrial case study of autonomous transport robots*. In: *Proc. 25th CS&P 2017-Concurrency, Specification and Programming*.

[4] Gerd Behrmann, Alexandre David & Kim G Larsen (2004): *A tutorial on Uppaal*. In: *Formal methods for the design of real-time systems*, Springer, pp. 200–236, doi:10.1007/s100090050010.

[5] Taha Chettibi, Moussa Haddad, HE Lehtihet & Wisama Khalil (2006): *Suboptimal trajectory generation for industrial robots using trapezoidal velocity profiles*. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp. 729–735, doi:10.1109/IROS.2006.282621.

[6] Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis & Danny Bøgsted Poulsen (2015): *Uppaal SMC tutorial*. *International Journal on Software Tools for Technology Transfer* 17(4), pp. 397–415, doi:10.1007/s10009-014-0361-y.

[7] Mohammed Foughali (2017): *Toward a correct-and-scalable verification of concurrent robotic systems: insights on formalisms and tools*. In: *2017 17th International Conference on Application of Concurrency to System Design (ACSD)*, IEEE, pp. 29–38, doi:10.1109/ACSD.2017.10.

[8] Mohammed Foughali, Félix Ingrand & Cristina Seceleanu (2019): *Statistical model checking of complex robotic systems*. In: *International Symposium on Model Checking Software*, Springer, pp. 114–134, doi:10.1016/S1571-0661(05)80435-9.

[9] Carlo A Furia, Dino Mandrioli, Angelo Morzenti & Matteo Rossi (2012): *Modeling time in computing*. Springer Science & Business Media, doi:10.1007/978-3-642-32332-4.

[10] Yuri Gordienko, Sergii Stirenko, Yuriy Kochura, Oleg Alienin, Michail Novotarskiy & Nikita Gordienko (2017): *Deep learning for fatigue estimation on the basis of multimodal human-machine interactions*. *arXiv preprint arXiv:1801.06048*.

[11] Raju Halder, José Proença, Nuno Macedo & André Santos (2017): *Formal verification of ROS-based robotic applications using timed-automata*. In: *2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE)*, IEEE, pp. 44–50, doi:10.1109/FormaliSE.2017.9.

[12] ISO 13482 (2014): *Robots and robotic devices - Safety requirements for personal care robots*. International Organization for Standardization.

[13] Mohamad Y Jaber, ZS Givi & W Patrick Neumann (2013): *Incorporating human fatigue and recovery into the learning–forgetting process*. *Applied Mathematical Modelling* 37(12-13), pp. 7287–7299, doi:10.1016/j.apm.2013.02.028.

[14] Theo Jacobs & Gurvinder Singh Virk (2014): *ISO 13482 - The new safety standard for personal care robots*. In: *ISR/Robotik 2014; 41st International Symposium on Robotics*, VDE, pp. 1–6.

[15] Stephan Konz (2000): *Work/rest: Part ii-the scientific basis (knowledge base) for the guide 1*. *Ergonomics Guidelines and Problem Solving* 1(401), p. 38.

[16] Xinxin Li, Rui Wang, Yu Jiang, Yong Guan, Xiaojuan Li & Xiaoyu Song (2017): *Formal modeling and automatic code synthesis for robot system*. In: *2017 22nd International Conference on Engineering of Complex Computer Systems (ICECCS)*, IEEE, pp. 146–149, doi:10.1109/ICECCS.2017.17.

[17] Levente Molnar & Sandor M Veres (2011): *Hybrid automata dicretising agents for formal modelling of robots*. *IFAC Proceedings Volumes* 44(1), pp. 49–54, doi:10.3182/20110828-6-IT-1002.03022.

[18] Olivier Tremblay, Louis-A Dessaint & Abdel-Illah Dekkiche (2007): *A generic battery model for the dynamic simulation of hybrid electric vehicles*. In: *2007 IEEE Vehicle Power and Propulsion Conference*, Ieee, pp. 284–289, doi:10.1109/VPPC.2007.4544139.

[19] GS Virk, S Moon & R Gelin (2008): *ISO standards for service robots*. In: *Advances In Mobile Robotics*, World Scientific, pp. 133–138, doi:10.1142/9789812835772_0016.

[20] Rui Wang, Yong Guan, Houbing Song, Xinxin Li, Xiaojuan Li, Zhiping Shi & Xiaoyu Song (2018): *A formal model-based design method for robotic systems*. *IEEE Systems Journal* 13(1), pp. 1096–1107, doi:10.1109/JSYST.2018.2867285.

[21] Matt Webster, Clare Dixon, Michael Fisher, Maha Salem, Joe Saunders, Kheng Lee Koay & Kerstin Dautenhahn (2014): *Formal verification of an autonomous personal robotic assistant*. In: *2014 AAAI Spring Symposium Series*.

[22] Yuchen Zhou, Dipankar Maity & John S Baras (2016): *Timed automata approach for motion planning using metric interval temporal logic*. In: *2016 European Control Conference (ECC)*, IEEE, pp. 690–695, doi:10.1109/ECC.2016.7810369.