Formal Verification of Human-Robot Interaction in Healthcare Scenarios

Livia Lestingi[1], Mehrnoosh Askarpour[2], Marcello M. Bersani[1], Matteo Rossi[1]

[1] Politecnico di Milano, {firstname}.{lastname}@polimi.it
[2] McMaster University, askarpom@mcmaster.ca

Abstract. We present a model-driven approach for the creation of formally verified scenarios involving human-robot interaction in healthcare settings. The work offers an innovative take on the application of formal methods to human modeling, as it incorporates physiology-related aspects. The model, based on the formalism of Hybrid Automata, includes a stochastic component to capture the variability of human behavior, which makes it suitable for Statistical Model Checking. The toolchain is meant to be accessible to a wide range of professional figures. Therefore, we have laid out a user-friendly representation format for the scenario, from which the full formal model is automatically generated and verified through the Uppaal tool. The outcome is an estimation of the probability of success of the mission, based on which the user can refine the model if the result is not satisfactory.

Keywords: Human-Robot Interaction · Service Robots · Healthcare Robotics · Model-Driven Approach · Statistical Model Checking

1 Introduction

Robots are becoming increasingly widespread in non-industrial contexts. Future applications range from autonoumous means of transportation [28] to personal assistants [14]. If on one hand factory workers are specifically trained to interact with this type of machinery and grow familiar with safety practices over time, on the other hand a much wider range of users awaits robots in the near future [22] and this poses a new set of challenges. In the area of safety, strong guarantees of people's health preservation will be required, beginning from compliance with the standards. ISO 12100 [18] and ISO 13849-1 [20] provide general guidelines, whereas ISO 13482 [19] is specifically targeted to service robots. It is common knowledge for researchers in the field that testing techniques are not sufficient given the complexity of these systems [25], and there already exist works in literature that resort to formal verification for safety-critical scenarios [16, 30].
In this paper, we focus on human-related aspects in scenarios involving interactions between humans and robots. Service robots will support people in everyday life situations, and it is, therefore, crucial that human factors are accounted for in the robot decision-making process. To this end, we have developed a model-driven approach for the creation and formal verification of scenarios related to

human-centric robotic applications. Target scenarios feature mobile robots and their controllers interacting with humans in a defined environment. Separate scenarios tackle different interaction strategies, depending on what specific service the human is requesting. To ease the modeling process, we have identified a set of usual coordination patterns involving a human and a robot, which the user of the approach can customize and re-use with minimum effort. The *mission* of the robot, as more thoroughly discussed in Section 4, will be to provide all the services requested in a specific scenario, and the purpose of the approach is to allow the designer of the application to verify the feasibility of the mission.
The approach considers some physiological features of humans that make it particularly suited for the healthcare domain, but it is still general enough to befit a broader range of applications. The policies of the robot controller that drive the mission to success are designed to prioritise human needs over efficiency. This can make a difference when robots interact with people in discomfort or in pain, or when human lives are at stake and the controller has to make a decision—something that can happen in healthcare settings. Another key issue for practitioners is that modeling the volatility of human behavior is often considered beyond the capabilities of formal methods [25]. As a first step towards solving this issue, in our approach humans are not treated as rational agents that unmistakably execute their mission. Instead, their model features a stochastic component to simulate free will, that causes them to occasionally make autonomous decisions.
In our approach, scenarios are formally modeled through Hybrid Automata [4], with the addition of stochastic components. The model is formally verified through Statistical Model Checking (SMC) [3] against a set of relevant properties. The toolchain is meant to be used by professional figures with a technical background, though not necessarily in robotics nor in formal models. Therefore, the entry point is a user-friendly representation of the key parameters of the scenario that the application designer can smoothly produce and refine. The tool processes this input, automatically generates the complete formal models, and performs verification through the Uppaal tool [11, 24].
We have selected a use case from the healthcare domain to evaluate the effectiveness of the approach. Some significant experimental results drawn from the use case are presented in the paper. The experiments are run using a publicly available prototype of the tool [1].
The paper is structured as follows: Section 2 surveys related works; Section 3 outlines the background; Section 4 presents the approach; Section 5 describes the formal model; Section 6 presents the use case and the experimental results; finally, Section 7 concludes. Appendix A presents additional details about the formal model, and Appendix B describes additional experiments.

2 Related Work

Formal modeling and verification of systems that involve interactions between humans and robots is a long-standing issue. The work by Webster et al. [31]

analyses the dynamics of a person assisted by a personal care robot in a smart home. All the elements of the scenario are modeled as *agents* using Brahms, and then verified through the SPIN model-checker: in particular, the human non-deterministically selects the action to perform from a pre-determined set, and the robot must act accordingly. Vicentini et al. [30] focus on the risk assessment procedure of industrial collaborative tasks. The authors propose a framework based on LTL formulae to verify that the overall level of risk of the system does not exceed a certain threshold. The work has been extended to also include manifestations of erroneous human behavior, handled in a black-box manner [7].
Bersani et al. [9] address applications involving robots and humans working in a shared environment, modeled as networks of Timed Game Automata. Humans are modeled as *uncontrollable* agents, to capture the uncertainty of the real world. A robot controller that also accounts for unpredictable human moves is then automatically synthesized through the Uppaal-TIGA tool.
Porfirio et al. [27] explore how formal verification can be used to ensure that robots adhere to *social* norms while interacting with humans. Norms, expressed as LTL formulae, constitute the properties to be verified, whereas the sequence of interactions is modeled as a composition of state machines. Concerning the social robotics field, Adam et al. [2] propose a framework based on the BDI architecture to make human-robot interaction feel more natural. The authors build upon models of human cognition to develop a perception and deliberation process that drives the robot towards making decisions in a human-like fashion.
Some works exploit SMC to verify robotic systems. Arai and Schlingloff [6] use SMC to make predictions on the performance of autonomous transport robots in production plants. Foughali et al. [13] apply SMC to formally verify real-time properties, like schedulability and readiness, of robotic software. Herd et al. [17] focus on multi-agent systems, and on swarm robotics in particular: in this case, SMC helps deal with the size of the problem, which cannot be handled by traditional model checking techniques.
Finally, Zhao et al. [12] perform probabilistic model-checking of UAV missions, focusing on advanced aspects of battery prognostics and health management.
As this brief survey shows, although the issue of verifying human-robot interaction has been tackled with different techniques, the combination of sound formal methods with the modeling of human behavior unpredictability is still an open question. To the best of the authors' knowledge, the work in this paper is the first attempt at formally verifying an explicit model of human-related aspects, such as free will. In addition, as mentioned in Section 1, service robots will come into contact with people with different backgrounds, who may not be able to create complex models while designing their application. Therefore, the accessibility of a tool is a key factor in assessing its effectiveness. The work presented in this paper addresses these issues and is a first step towards filling the gaps.

3 Background

The formal models presented in this work are expressed through Hybrid Automata (HA) enriched with a stochastic component, indicated as HA+ in the rest of the paper. HA are an extension of Timed Automata. In Timed Automata, time is modeled through variables (*clocks*), whose value increases linearly with time unless they are reset [5]. Conditions on clocks govern the transitions between states. In HA, locations are also endowed with sets of differential equations (*flow conditions*) that constrain the derivatives of real-valued variables of the model, thus allowing for the modeling of systems with complex dynamics [4].
Automata can be organized in *networks*, where synchronization between different automata occurs through *channels*. Given a channel e and two automata with complementary enabled transitions labelled as $e?$ and $e!$—whose guards are both satisfied—the two transitions fire at the same time and the two automata synchronously change their locations. Non-deterministic choices in the model can be refined by probabilistic distributions. These constitute the stochastic component of the formalism and are modeled via probabilistic transitions, as exemplified in Fig. 1. This type of transition is marked with a probability weight, that determines how much the system will be biased to evolve in a certain direction rather than its alternatives.
The stochastic component of the formalism enables the application of statistical techniques, and SMC in particular. As opposed to traditional model checking, SMC relies on Hypothesis Testing or Estimation to evaluate the probability that a property holds on paths starting from a generic state s of the system [3]. Therefore, it does not fully explore the state space, and is feasible also for complex systems. The properties to be checked are expressed in the PCTL logic, whose syntax, shown in Table 1, allows us to express quantitative constraints on probabilities through the $P_{\geq\theta}(\phi)$ operator. Given a HA+ automaton and a PCTL property ψ, the possible outcomes of SMC are: (a) a binary value, 1 or 0, depending on whether $P(\psi) \geq \theta$ holds or not, where $P(\psi)$ is the probability of ψ holding, and θ is a threshold; or (b) a probability interval to which $P(\psi)$ is guaranteed to belong. Section 6 presents some examples of SMC experiments.
The features of HA+ are used for the robot and human models presented in detail in Section 5. In particular, given the emphasis of the work on human physiology, the model of the human includes differential equations describing the time-dynamics of fatigue. Although human fatigue can take different forms [21], at the moment we focus on the physical strain originated from non-stop walking. According to Konz [23], the alternate fatigue/recovery cycles can be modeled as exponential curves, as in Eq.1. Low values of coefficients λ and μ mean slower

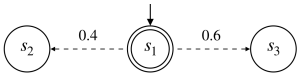


Fig. 1: Automaton with probabilistic transitions: 0.4 and 0.6 are the probabilities of reaching s_2 and s_3 starting from s_1.

$\phi ::= a \mid \neg\phi \mid \phi \vee \phi' \mid \phi \wedge \phi'$
$\psi ::= \phi \mid \mathbf{X}\phi \mid \phi \mathbf{U} \phi' \mid P_{\geq\theta}(\psi)$
$a \in AP$ atomic proposition
$\theta \in [0, 1]$ probability bound

Table 1: PCTL Syntax

fatigue accumulation and recovery, while $\mathsf{F_0}$ corresponds to the value of fatigue at the start of the current cycle (it is 0 when the human starts walking for the first time). Full exhaustion corresponds to $F(t) = 1$, full recovery to $F(t) = 0$.

$$F(t) = \begin{cases} 1 - (1 - \mathsf{F_0})e^{-\lambda t} & \text{(walking)} \\ \mathsf{F_0}e^{-\mu t} & \text{(resting)} \end{cases} \tag{1}$$

Concerning the model of robot velocity, we use the typical trapezoidal velocity profile [10] with three phases: acceleration, constant maximum speed, and deceleration. Maximum acceleration $\mathsf{a_{max}}$ and maximum velocity $\mathsf{v_{max}}$ are design parameters of the robot. For the robot battery, we assume an exponential charge/discharge cycle typical of electronic devices with lithium batteries [29]. Eq.2 shows the battery charge time-dynamics: ρ and σ represent the charge/discharge rates, whereas C_0 is the starting charge value for the current cycle. The times required for a full charge and a full discharge cycle ($\mathsf{T_{chg}}$ and $\mathsf{T_{dchg}}$) are known a-priori given the battery model, therefore rates ρ and σ are approximated with precision ϵ as in the following: $\rho = \frac{1}{\mathsf{T_{dchg}}} ln(\frac{100}{100-\epsilon})$, $\sigma = \frac{1}{\mathsf{T_{chg}}} ln(\frac{100}{100-\epsilon})$.

$$C(t) = \begin{cases} 100 - (100 - \mathsf{C_0})e^{\rho t} & \text{(discharging)} \\ 100 - (100 - \mathsf{C_0})e^{-\sigma t} & \text{(charging)} \end{cases} \tag{2}$$

4 Approach

The main contribution of this paper is a model-driven approach for the analysis of human-robot interaction through formal verification. The approach is tailored to non-industrial settings, and in particular to the healthcare environment. Indeed, we focus on scenarios in which the volatility of human behavior is at its peak, which may not be the case for industrial workers who are methodically trained to perform a set of actions during their shift. Furthermore, modeling physiology-related aspects is crucial for healthcare applications, since people in need of a medical service may find themselves in diverse, even critical, physical conditions. The toolchain is meant to be used by professionals possibly with some technical background, but not necessarily in robotics or in formal methods. Hospitals are subject to a tremendous flow of people on a day-to-day basis, and this requires dedicated professional figures to be efficiently handled. For example, clinical workflow analysts [26], who design and analyze work shifts for medical facilities, perfectly fit this profile.

The application designer is in charge of assigning each of the currently requested services to one of the available robots. Therefore, a group of humans served by a robot will constitute the atomic operational unit (referred to as the *scenario* hereinafter). Serving everybody in the group will constitute the *mission*, i.e., the high-level goal [8], for the robot. The boundaries of the mission also include any other parameters that, were they to be modified, the overall outcome would change accordingly, e.g., the enclosed environment where the agents operate or the robot starting conditions. Through the tool, the designer can try different

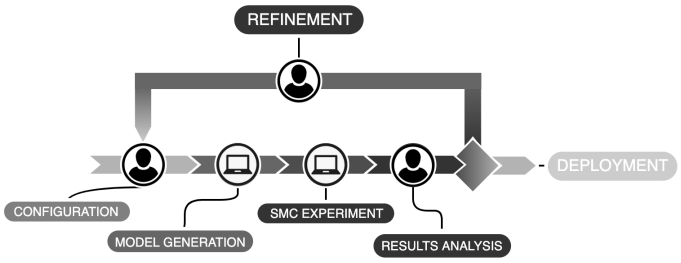


Fig. 2: Diagram representing all the phases of the approach. Different shades of gray indicate the current progress of implementation: the darker phases are fully implemented, the lighter ones are to be expanded in the future.

configurations until the estimated probability of success is reasonable. Fig. 2 depicts the steps of the design process: (i) configuration of the scenario, performed by the designer; (ii) automated model generation; (iii) execution of the SMC experiment; (iv) critical assessment of the verification results, followed by application deployment if the results are satisfactory, otherwise by model refinement. In this paper, we focus on the scenario configuration and on the formal model, whereas the deployment phase will be elaborated in future works.

4.1 Configuration

The user of the toolchain has the option to customize all the parameters of the scenario depicted in Fig. 3 and described in the following.

A scenario includes $\mathsf{N_h}$ humans to be served and a robot from the fleet that will serve them. Robots are characterized by maximum speed and acceleration ($\mathsf{v_{max}}$ and $\mathsf{a_{max}}$ in Fig. 3) and are each associated with a battery. An association between a robot and a battery constitutes a robotic system with its own id. Batteries possess an initial charge value $\mathsf{C_{start}} \in [0, 100]$, and the two parameters $\mathsf{T_{chg}}$ and $\mathsf{T_{dchg}}$ that determine the duration of full charge/discharge cycles. For each scenario, it is also essential to model the environment in which services

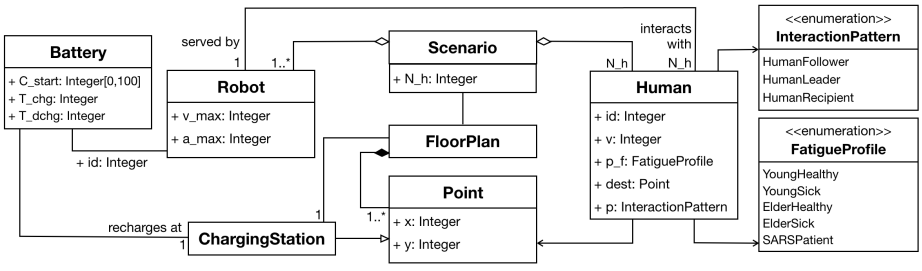


Fig. 3: Class Diagram of the user-customizable portion of the model.

Table 2: Human-Robot Interaction Patterns

Pattern	Description
Human Follower	The human follows the robot while the robot moves towards the destination. Because of *free will*, the human can decide to walk freely and whether to follow or not when the robot issues its command. If they get too far, the robot stops and waits for them.
Human Leader	The robot follows the human that moves towards the destination (unknown to the robot). The human is free to start and stop whenever they want and the robot follows accordingly.
Human Recipient	The human waits for the robot to fetch an item from a specific location. While the robot is delivering the object, the human is free to move and the robot has to track them. Synchronization occurs when the human and the robot are close and the human stops to pick up the item.

will be carried out. Specifically, the operational environment is modeled as a two-dimensional layout. The designer may specify the Cartesian coordinates of specific points of interest—e.g., wall corners and doorposts. The model capturing robot navigation takes these elements into account to drive the agent towards their correct destination and to prevent collisions with walls or humans.
Together with robots, humans are a basic component of the scenario that needs to be configured by the designer. Each human is identified by a unique id that determines the order in which they will be served. Furthermore, the user can specify the human's walking speed v and the destination point dest. A fundamental aspect of humans in the scenario is the way in which they are going to interact with the robot. We have identified a set of patterns for recurrent human-robot synchronization mechanisms, focusing on a particular subset suitable for the mobile robots with a predefined set of functionalities covered by our work. Currently implemented patterns are described in Table 2. Different patterns impact how the system will evolve while a certain service is being carried out and the condition that the system needs to verify to state that the service has been provided. As shown in Fig. 3, the designer can specify the parameter p for each human to set up how the interaction with the robot will play out.
Finally, it is also possible for the designer to customize the physiological traits of the humans in the scenario through the $\mathsf{p_f}$ parameter (see Fig. 3). This draws from a set of profiles that aggregate potential subjects by *age* and *medical condition*. Since the physiological property currently included in the model is fatigue, particular care is given to conditions that specifically target the respiratory functions of patients, thus their ability to walk. Providing the designer with this predetermined set of profiles allows them to match the specific individuals from their scenario with medically recognized significant cases. This also allows them to save time while designing the application, although the manual specification of a different set of parameters is still possible if necessary. Referring to the

model of fatigue introduced in Section 3, different profiles mean different values for parameters λ and μ. Therefore, they determine the time needed by a specific subject to reach full exhaustion and full recovery. With this feature, the professional assessing the results has the guarantee that the physical condition of the patient has been taken into account while estimating the outcome of the mission.

4.2 Model Generation and SMC Experiment

The HA+ models are handled as templates customizable by the user. This subset is, indeed, the result of the configuration phase. By doing so, the designer is only required to arrange the main elements of the application in a user-friendly format, which is more suited to their technical background than creating formal stochastic models from scratch. This input is fed to a script that automatically generates the formal model with the values laid down by the designer. Once the generation of the model is completed, the tool initiates the verification process. The user also has the option to choose whether the tool should produce a simulation of the system or estimate the probabilities of the mission ending in failure or success—i.e., a typical SMC experiment.

4.3 Result Analysis and Refinement

If the user decides to run an SMC experiment, this yields a probability value. If the probability of success is smaller than a desirable threshold, the user may refine the model in one of the following ways: (a) reduce the workload of a robot, for example, if its current battery charge value is not sufficient to carry out all the requested services; (b) change the order in which humans are to be served, which could improve the overall efficiency, e.g., by reducing robot movements between a service and the next one; (c) choose different services to be included in the scenario; (d) choose a different robot from the fleet, with different speed/acceleration parameters or with a different battery charge value. A different robot model may be useful in case the previous one moved too fast for the human, whereas issues related to the battery may involve complete discharge before the mission is done.

5 Model

In this section, for each type of component of a scenario (robots, batteries, humans) we present the corresponding HA+. The model also features an *orchestrator*, which plays the pivotal role of managing the synchronization among all other components through *channels* (see Section 3). The decisions of the orchestrator are based on the state of the system. In particular, specific features of the model capture the behavior of sensors measuring physical properties of the system, which drive the decisions of the orchestrator. Each property is modeled via a *dense counter* variable that changes over time as a result of *update* instructions

on transitions, but does not possess an explicit time-dependency. We also assume that these sensors periodically repeat the measurement: constant $\mathsf{T_{poll}}$ captures the refresh period and clock t_{upd} measures the time elapsed since the last update. As mentioned in Section 3, the robot movement follows a trapezoidal velocity profile, whereas the human moves with constant speed. As for human free will, we have chosen the straightforward approach [15] of modeling it as a random phenomenon whose behavior is comparable to a Bernoulli variable X. Finally, to dampen the complexity of the model, we assume that humans are only free to choose *when* to start or stop, and not an arbitrary trajectory. Interested readers can find additional details about the models in Appendix A.

Robot: the HA+ in Fig. 4 represents the three operating conditions of the robot, corresponding to idleness, motion and battery recharging. We introduce two time-dependent variables V and r_{dist} that model, respectively, the velocity of the robot at a generic time instant t and the distance covered since the beginning of the motion. The automaton features locations r_{idle}, r_{start}, r_{mov}, r_{stop}, and r_{rec} corresponding, respectively, to: (1) the idleness of the robot with $V = 0$; (2) the acceleration phase of the motion, thus $\dot{V} = \mathsf{a_{max}}$; (3) the travel phase with constant speed, $V = \mathsf{v_{max}}$; (4) the deceleration phase with $\dot{V} = -\mathsf{a_{max}}$; (5) the battery recharging phase, thus $V = 0$. In every location, $\dot{r}_{dist} = V$ holds. When the orchestrator fires the commands to start or stop recharging ($\mathtt{b}_{\text{start}}$ and $\mathtt{b}_{\text{stop}}$), if the robot is at the recharging station, the automaton transitions from r_{idle} to r_{rec} and back. The switch from r_{idle} to r_{start} takes place when the command to start moving ($\mathtt{r}_{\text{start}}$) is issued. Similarly, the automaton switches from r_{mov} to r_{stop} when $\mathtt{r}_{\text{stop}}$ is fired. It is also possible to start and stop the robot while it is accelerating or decelerating, so two transitions are added between r_{start} and r_{stop}. In location r_{start}, velocity is increasing linearly from 0 to $\mathsf{v_{max}}$, thus when $V = \mathsf{v_{max}}$ the automaton switches to r_{mov}. While decelerating, the robot stays in r_{stop} as long as $V > 0$ and goes back to r_{idle} when $V = 0$. We model as dense counters the Cartesian coordinates of the robot in space (r_{pos_x} and r_{pos_y}), and the angle θ_r for the orientation with respect to the x-axis. On every self-loop, and on all transitions in Fig. 4 marked with a ξ_{R}, the corresponding update instruction of Table 3 is executed.

Robot Battery: Fig. 5 shows the HA+ representing the behavior of the battery. The time-dependent variable in the model is the charge value C. The robot

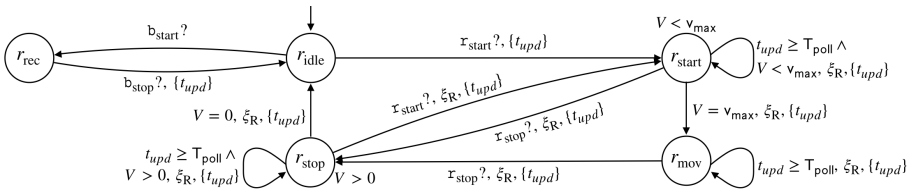


Fig. 4: Robot automaton.

battery can be in one of three states, modeled by as many locations: discharging (b_{dchg}), charging (b_{chg}), and fully discharged (b_{empty}) in which case the robot cannot move autonomously. The constraints on the derivative of charge C stemming from the model introduced in Section 3 are shown in Eq.3.

$$\dot{C} = \begin{cases} -(100 - \mathsf{C}_0)\rho e^{\rho t} & (b_{\text{dchg}}) \\ (100 - \mathsf{C}_0)\sigma e^{-\sigma t} & (b_{\text{chg}}) \end{cases} \tag{3}$$

The switch from b_{dchg} to b_{chg} and vice versa occurs when the orchestrator decides that a robot needs to start/stop recharging, firing events $\mathtt{b}_{\text{start}}$ and $\mathtt{b}_{\text{stop}}$, respectively. An additional location b_{full} is needed to model the case in which the battery is back to full charge (condition $C = 100$) and immediately stops recharging. This happens thanks to the *urgent* channel $\mathtt{b}_{\text{full}}$ [24] that fires as soon as the automaton enters location b_{full}. The sampled charge value is modeled by dense-counter b_{ch}, whose update instructions ξ_{BC} and ξ_{BD} are shown in Table 3 and correspond, respectively, to the charge and discharge operating conditions.

Human-Follower: the model is depicted in Fig. 6. The time-dependent variables are h_{dist}, which represents the distance covered by the human, and F, which corresponds to the value of the fatigue at a generic time instant. Their temporal dynamics are given in Eq.4: h_{dist} is either constant, or it increases linearly with time (with coefficient v, which is the human's constant speed) when the human is moving, whereas F adheres to the model in Eq.1.

$$h_{\text{idle}} = \begin{cases} \dot{F} = -\mathsf{F}_0 \mu e^{-\mu t} \\ \dot{h}_{dist} = 0 \end{cases} \qquad h_{\text{busy}} = \begin{cases} \dot{F} = \mathsf{F}_0 \lambda e^{-\lambda t} \\ \dot{h}_{dist} = \mathsf{v} \end{cases} \tag{4}$$

The operating conditions modeled for this component are the idleness of the human (location h_{idle}) or walking (h_{busy}). The switch from h_{idle} to h_{busy}, and back, occurs when the orchestrator orders it or as a result of the human's free will. In the first case, the orchestrator fires $\mathtt{h}_{\text{start}}$ or $\mathtt{h}_{\text{stop}}$. This leads to a probabilistic transition (the dashed arrows in Fig. 6) whose possible outcomes represent the human *obeying* the order, thus reaching the prescribed destination, or *disobeying* it, thus staying in the same location. The transition is governed by the two constant weights obey and disobey. In the second case, two additional transitions between h_{idle} and h_{busy} capture autonomous decisions as a result of free will:

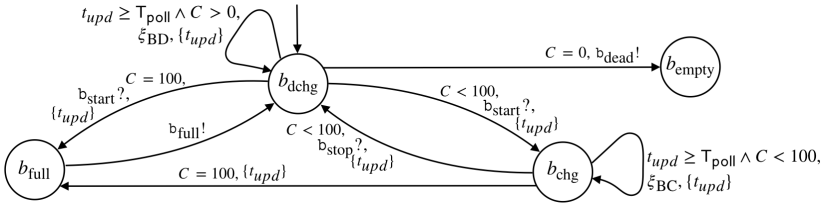


Fig. 5: Battery automaton.

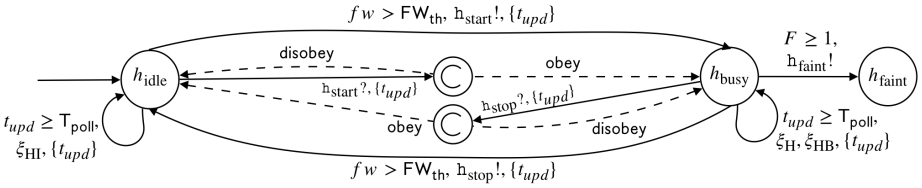


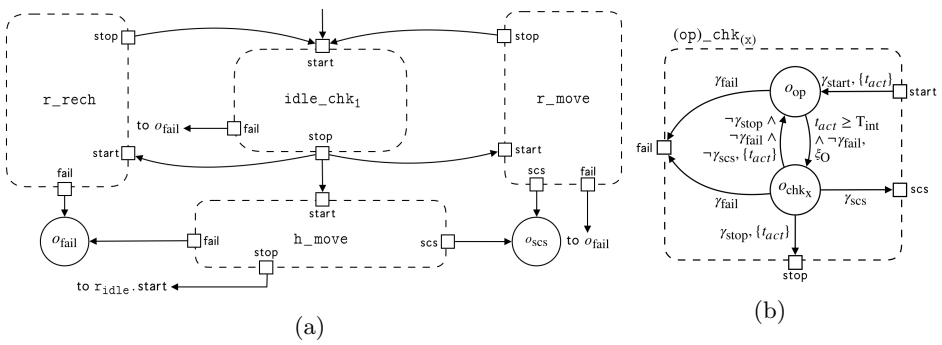
Fig. 6: Human-Follower automaton.

Table 3: Update Instructions

$\xi_{\text{R}}: \begin{array}{l} r'_{pos_x} := r_{pos_x} + V(t)\mathsf{T_{poll}} cos(\theta_r) \\ r'_{pos_y} := r_{pos_y} + V(t)\mathsf{T_{poll}} sin(\theta_r) \end{array}$	$\xi_{\text{H}}: \begin{array}{l} h'_{pos_x} := h_{pos_x} - \mathsf{vT_{poll}} cos(\theta_h) \\ h'_{pos_y} := h_{pos_y} - \mathsf{vT_{poll}} sin(\theta_h) \end{array}$
ξ_{BC}: $b'_{ch} := 100 - (100 - b_{ch})e^{-\sigma \mathsf{T_{poll}}}$	ξ_{BD}: $b'_{ch} := 100 - (100 - b_{ch})e^{\rho \mathsf{T_{poll}}}$
ξ_{HI}: $h'_{ftg} := h_{ftg} e^{\mu \mathsf{T_{poll}}}$	ξ_{HB}: $h'_{ftg} := 1 - (1 - h_{ftg})e^{-\lambda \mathsf{T_{poll}}}$

thus, the human fires $\mathtt{h}_{\text{start}}$ and $\mathtt{h}_{\text{stop}}$. Sample points of the random variable X are observations of local variable fw. A *success* is the decision of the human to start or stop freely and it occurs when $fw > \mathsf{FW_{th}}$, which is the guard condition of the transitions between h_{idle} and h_{busy}. $\mathsf{FW_{th}}$ is a constant threshold, and variable fw is updated every $\mathsf{T_{poll}}$ instants with a random value in range $[0, \mathsf{FW_{max}}]$. Therefore, the probability of making an autonomous decision is $E[X] = p = 1 - \frac{\mathsf{FW_{th}}}{\mathsf{FW_{max}}}$. If the p-value is close to 1, humans will behave more erratically, and vice versa if it is close to 0. Location h_{faint} models the case in which the human is too exhausted to proceed: it is reached when $F \geq 1$, where 1 corresponds to the maximum value of fatigue, and it causes *urgent* channel $\mathtt{h}_{\text{faint}}$ to fire immediately. The dense counters are h_{ftg} (for the fatigue), and h_{pos_x}, h_{pos_y}, θ_h for the Cartesian coordinates and orientation of the human with respect to the x-axis. Table 3 shows the update instructions (ξ_{HI}, ξ_{HB}) for the fatigue, which adhere to the model in Eq.4, and also those (ξ_{H}) of the position, where h_{pos_x}, h_{pos_y} represent the projections of the displacement since the last update.

Orchestrator: the automaton is displayed in Fig. 7a. The purpose of this component is to orchestrate the synchronization among the other agents and drive the system towards mission accomplishment. This is realized by monitoring the sensor outputs described in previous sections, and deciding whether the current state of the system requires a certain event to be fired. We have identified three operational paradigms implemented by as many sub-machines: `r_rech` controls the recharging phase of the robot; `r_move` controls the start and the end of the movement when, based on the interaction pattern between the human and the robot carrying out the service, it is initiated by the robot; `h_move` controls the dual case, in which the movement is initiated by the human. The orchestrator features both `r_move` and `h_move` since both designs can be included in the same



(a) (b)

Fig. 7: Orchestrator automaton and $\langle \texttt{op} \rangle\_\texttt{chk}_{\langle \texttt{x} \rangle}$ pattern.

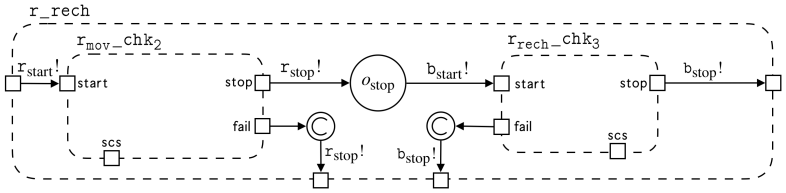
scenario. Sub-machines in Fig. 7a are endowed with *ports*: these are not part of the formalism, but a visual representation of the transitions entering and leaving the component (arrows in and out of a port constitute the *same* transition).

Figure 8 shows the details of the sub-machines. They are all built using the $\langle \texttt{op} \rangle\_\texttt{chk}_{\langle \texttt{x} \rangle}$ pattern of Fig. 7b, which includes a location modeling the current operating condition of the entire system (e.g., $o_{\mathrm{r_{rech}}}$, $o_{\mathrm{r_{flw}}}$), generically identified by o_{op}, and a location $o_{\mathrm{chk_x}}$ modeling the orchestrator monitoring the state of the system, where $\texttt{x}$ is a numerical index. Ports highlight the transitions entering and leaving the pattern sub-component, guarded by γ_{start}, γ_{stop}, γ_{fail} and γ_{scs}, where each γ condition is associated with a component-specific formula. The semantics of the pattern is described in the following.

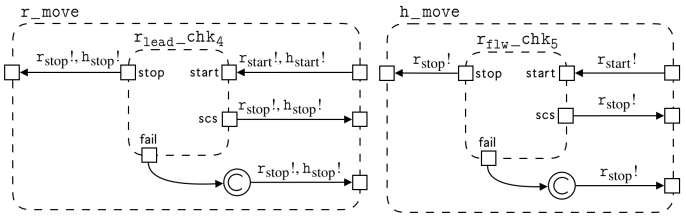
The orchestrator moves to operating condition o_{op} when the corresponding condition γ_{start} is true. The transition from o_{op} to $o_{\mathrm{chk_x}}$ is governed by clock t_{act}, and it periodically occurs every $\mathsf{T_{int}}$ time instants. Upon entering $o_{\mathrm{chk_x}}$, the orchestrator runs the monitoring routine (ξ_{O} of Fig. 8). If condition γ_{stop} holds, the orchestrator moves to the following operating condition—i.e., a different sub-component—otherwise it goes back to o_{op}. Table 4 shows the guards for each sub-machine. Locations o_{fail} and o_{scs} of Fig. 7a correspond to the end of the mission with failure or success, respectively, and are reached when conditions γ_{fail} and γ_{scs} hold. Failure occurs if the battery charge drops to 0 (event $\mathtt{b}_{\mathrm{dead}}$), hence the robot cannot recover autonomously, or if the human fatigue exceeds 1 (event $\mathtt{h}_{\mathrm{faint}}$). Location o_{scs} is reached when the mission has been successfully completed—i.e., when all humans in the scenario have been served.

The first instance of the pattern is $\texttt{idle\_chk}_1$ in Fig. 7a, which models the situation in which the system is idle and periodically checks whether an action can start. The system enters this component first when the execution starts, and returns to it whenever an action stops (and the corresponding sub-component is left). Similarly, the orchestrator exits this component if one of the γ_{start} conditions for the other sub-machines is true.

The recharging routine of Fig. 8a starts when a robot is idle and its current battery charge b_{ch} is below a threshold $\mathsf{B_{th_1}}$. Sub-machine $\texttt{r\_rech}$ models two



(a) `r_rech` sub-automaton.



(b) `r_move` sub-automaton. (c) `h_move` sub-automaton.

Fig. 8: Sub-Machines of the Orchestrator automaton.

Table 4: Orchestrator Guard Conditions

	γ_{start}	γ_{stop}
$\mathtt{idle\_chk}_1$	$\mathtt{r}_{\mathtt{rech}}.\gamma_{\text{stop}} \vee \mathtt{r}_{\mathtt{lead}}.\gamma_{\text{stop}} \vee \mathtt{r}_{\mathtt{flw}}.\gamma_{\text{stop}}$	$\mathtt{r}_{\mathtt{mov}}.\gamma_{\text{start}} \vee \mathtt{r}_{\mathtt{lead}}.\gamma_{\text{start}} \vee \mathtt{r}_{\mathtt{flw}}.\gamma_{\text{start}}$
$\mathtt{r}_{\mathtt{mov}}\mathtt{\_chk}_2$	$r_{\text{idle}} \wedge (b_{ch} < \mathsf{B}_{\mathsf{th}_1})$	$(r_{pos_x} = \mathsf{RS}_{\mathsf{x}}) \wedge (r_{pos_y} = \mathsf{RS}_{\mathsf{y}})$
$\mathtt{r}_{\mathtt{rech}}\mathtt{\_chk}_3$	$t_{act} \geq \mathsf{T}_{\mathsf{int}}$	$(b_{ch} > \mathsf{B}_{\mathsf{th}_2}) \wedge r_{\text{rec}}$
$\mathtt{r}_{\mathtt{lead}}\mathtt{\_chk}_4$	$r_{\text{idle}} \wedge \neg(h_{\text{p}} = 1)$	$(h_{ftg} > \mathsf{H}_{\mathsf{th}_1}) \vee (b_{ch} < \mathsf{B}_{\mathsf{th}_1}) \vee (h_{svd} \wedge \neg \forall_h h_{svd})$
$\mathtt{r}_{\mathtt{flw}}\mathtt{\_chk}_5$	$h_{\text{busy}} \wedge (h_{\text{p}} = 1)$	h_{idle}
$\langle \mathtt{op} \rangle\mathtt{\_chk}_{\langle \mathtt{x} \rangle}$	$\gamma_{\text{scs}} : \forall_h \; h_{svd}$	$\gamma_{\text{fail}} : \mathtt{b}_{\text{dead}}? \vee \mathtt{h}_{\text{faint}}?$

operating conditions: the movement of the robot towards the charging station ($\mathtt{r}_{\mathtt{mov}}\mathtt{\_chk}_2$), and the robot recharging its battery ($\mathtt{r}_{\mathtt{rech}}\mathtt{\_chk}_3$). Upon entering `r_rech`, the orchestrator fires $\mathtt{r}_{\text{start}}$ to instruct the robot to reach the charging station (Cartesian coordinates RS_{x} and RS_{y}), then $\mathtt{r}_{\text{stop}}$ when the dock has been reached. Location o_{stop} bridges the two operating conditions and models the deceleration phase of the robot (r_{stop}): this is why the orchestrator *waits* $\mathsf{T}_{\mathsf{int}}$ time instants before moving on, with $\mathsf{T}_{\mathsf{int}} > \mathsf{v}_{\mathsf{max}}/\mathsf{a}_{\mathsf{max}}$. When the robot has stopped completely, it can start recharging, and the orchestrator fires $\mathtt{b}_{\text{start}}$. The robot stops recharging, thus $\mathtt{b}_{\text{stop}}$ is fired, when variable b_{ch} is above a threshold $\mathsf{B}_{\mathsf{th}_2}$, then the orchestrator switches back to o_{idle}.

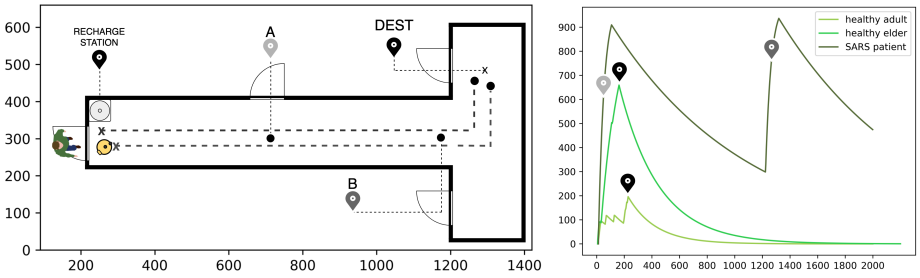
The `r_move` sub-machine of Fig. 8b is entered to initiate the robot movement when the robot is idle and the human is not a leader. Upon entering `r_move`, the orchestrator fires $\mathtt{r}_{\text{start}}$ and $\mathtt{h}_{\text{start}}$ since the human is a follower. The only operating condition modeled by this sub-component is the robot movement (component

$\mathtt{r}_{\mathtt{lead\_chk_4}}$ in Fig. 8b). The robot movement stops (events $\mathtt{r}_{\text{stop}}$ and $\mathtt{h}_{\text{stop}}$ fire, since the human is a follower) if: (a) human fatigue h_{ftg} exceeds a maximum tolerable value $\mathsf{H}_{\mathsf{th}_1}$, or (b) the battery charge drops below a value $\mathsf{B}_{\mathsf{th}_1}$ that calls for recharging, or (c) the human has been served, but they were not the last one. If the human is a leader and starts moving, hence its automaton is in location h_{busy}, the orchestrator enters sub-machine `h_move` (Fig. 8c). Upon entering `h_move`, $\mathtt{r}_{\text{start}}$ is triggered and the robot starts moving. The only operating condition is the robot following the human, modeled by $\mathtt{r}_{\mathtt{flw\_chk_3}}$. The orchestrator exits `h_move` when the human has stopped, and stops the robot with $\mathtt{r}_{\text{stop}}$.
As per Fig. 7a and Fig. 8, failure is possible for all sub-components. Instead, only `r_move` and `h_move` have outgoing transitions towards o_{scs}, since recharging the robot has no impact on service provision: this explains why the `scs` ports in Fig. 8a are not connected to the outer component.

6 Experimental Results

As previously discussed, use cases are conveniently found within the healthcare domain, specifically for the purposes of efficient patient flow handling. The experimental setting chosen to test the approach involves a human patient who needs to reach a doctor's office. The mobile robot is aware of the floor plan and the patient's characteristics, and it is able to guide the human towards the destination. When the service is successfully provided, the robot will have achieved its *mission*. Instances of classes and attributes of Fig. 3 are provided by the designer through a JSON file. The portion of the model that is not customizable by the user is stored in an XML template. The tool automatically processes the input of the user to generate a verification-ready version of the HA+ model. The tool selected for the verification is Uppaal and its extension for SMC [11, 24]. In this work we use Uppaal version 4.1.24 to implement the automata, and run SMC experiments[3] with the default set of statistical parameters. Each experiment yields the probability for mission success: formula $P^{\leq\tau}_{\geq\theta}(\diamond\ o_{\text{success}})$ is verified, with probability bound θ and time-bound τ.
With this experiment, we are able to test how different fatigue profiles impact the completion of the mission. The experimental setting features a mobile robot with $\mathsf{v_{max}} = 20\mathsf{cm/s}$, $\mathsf{a_{max}} = 5\mathsf{cm/s^2}$, with a fully charged battery ($\mathsf{C_{start}} = 100$) and approximately $2.5\mathsf{h}$ of full charge/discharge time ($\mathsf{T_{chg}} = \mathsf{T_{dchg}} = 9000\mathsf{s}$). Listing 1.2 shows the portion of the JSON file defining these parameters. The floor plan used for this experiment, depicted in Fig. 9a, reproduces a T-shaped hallway of a hospital, with doors leading to different offices. The entrance and starting point for both agents is on the left-end side (coordinates $(200, 300)$) close to the charging station ($\mathsf{RS_x} = 250, \mathsf{RS_y} = 375$). Listing 1.3 shows a snippet of the JSON file defining the coordinates for the points of the layout.
The mission for this robot is to lead a single human to their destination in $(1300, 500)$. Therefore, the interaction pattern is **Human-Follower**. To test the

[3] On a machine with 128 cores, 515GB of RAM and Debian Linux version 10.



(a) Floor plan and trajectories. (b) Fatigue curves.

Fig. 9: On the left, the floor layout for Exp A, with initial positions and destination marked by a ×, and intermediate points by a •. On the right, the fatigue curves for all the profiles tested in this experiment (colors as in the legend). Location markers indicate the time each human has reached the corresponding point of the trajectory.

effectiveness of the fatigue-related policies of the orchestrator, we repeat the experiment with three different versions of the human, with different fatigue profiles and values of walking speed. Listing 1.1 shows how these data are specified. In the first case, the human is young and in fine health ($\mathsf{p_f} = 1$). Their Maximum Endurance Time (MET)—i.e., how long they can walk non-stop before $F = 1$—is approximately 23 minutes (with reference to Eq.1, it leads to $\lambda = \mu = 0.005$) and their walking speed is $\mathsf{v} = 18\mathsf{cm/s}$. In the second case, the human is an elder in good health ($\mathsf{p_f} = 3$), with $\mathsf{MET} = 14\mathsf{min}$ ($\lambda = 0.008$, $\mu = 0.0035$) and $\mathsf{v} = 8\mathsf{cm/s}$. In the third case, the human is affected by a severe respiratory disease ($\mathsf{p_f} = 5$, $\mathsf{v} = 5\mathsf{cm/s}$) with $\mathsf{MET} = 4.6\mathsf{min}$ ($\lambda = 0.025$, $\mu = 0.001$).

Listing 1.1: Humans

```
"humans":[
 {"id": 1,
  "v": 5,
  "p": "follower",
  "p_f": 5,
  "dest": {
    "x": 1300,
    "y": 500}},[..]]
```

Listing 1.2: Robot

```
"robots": [
  {"id": 1,
   "v_max": 20,
   "a_max": 5,
   "c_start": 100,
   "T_chg": 9000,
   "T_dchg":9000},
   [..]]
```

Listing 1.3: Floor Plan

```
"floorPlan": [
  {"name":"HALL1",
   "x": 200,
   "y": 400},
  {"name":"HALL2",
   "x": 1200,
   "y": 400},
   [..]]
```

For the first case study, we have verified through the tool that $P(\diamond\, o_{\text{success}}) \in [0.717, 0.817]$ with $\tau = 300s$. With the second setting, the property that has been verified is $P(\diamond\, o_{\text{success}}) \in [0.8, 0.9]$ with $\tau = 300s$, and in the last case $P(\diamond\, o_{\text{success}}) < 0.098$ with $\tau = 1000s$. Furthermore, a simulation trace with $\tau = 2000s$ has also been produced for each test case. These results prove that in the first two cases the destination can be reached in approximately 5min with a high degree of confidence. In the last case, instead, it is practically impossible to

Table 5: Experiments Performance Data

Exp.	States	Time [min]	Virtual Memory [KiB]	Resident Memory [KiB]
$\mathsf{p_f} = 1$	212570	≈ 1	166488	120800
$\mathsf{p_f} = 3$	391311	≈ 1.5	166484	122376
$\mathsf{p_f} = 5$	2927009	≈ 11.5	166484	123068

complete the mission even in 16min. The reason behind this result is highlighted by the simulations in Fig. 9: Fig. 9a shows the trajectories of the two agents, whereas Fig. 9b shows the fatigue curves for the three test cases and the time instants in which the destination or intermediate points of the trajectory have been reached within the simulation. The orchestrator commands every agent to stop walking if human fatigue exceeds a certain threshold, set to $\mathsf{H_{th_1}} = 0.9$. In the last test case, motion stops at $t = 200s$, it resumes at $t = 1200s$ when fatigue drops to an acceptable value ($\mathsf{H_{th_2}} = 0.3$) and stops again at $t = 1400s$ when the destination is yet to be reached. This behavior is caused by the orchestrator trying to prevent human exhaustion, which inevitably slows down the entire mission. In the other two cases, thanks to the different $\mathsf{p_f}$ parameter values, when the human reaches the destination the value of fatigue is still acceptable, thus they are not stopped by the orchestrator.
Performance data for each experiment can be found in Table 5. The experiment demonstrates how the tool can predict the outcome of the mission for various scenarios with little effort on the designer-side. Moreover, it constitutes a preliminary step towards assessing the soundness of the models presented in Section 5, specifically the ability of the orchestrator to enable corrective actions if required by the state of the system.

7 Conclusion

We have presented a model-driven approach for the verification, through SMC, of human-robot interactions in healthcare scenarios. There are two main future development directions for the approach presented here. The model can be enriched with new interaction patterns to widen the range of applications that can be assessed. It is also possible to refine the model of human behavior by creating a correlation between environmental factors and the likelihood of autonomous decisions, which are a purely random phenomenon in the current version of the work. The maximum degree of scenario flexibility could also be enhanced by having the human and the robot dynamically shift from one pattern to another in particular situations as a result of the orchestrator's policies. We envisage the creation of a Domain-Specific Language that designers can use to model the mission, with finer-grained details and a richer set of physiological factors. Simultaneously, we plan on developing the deployment phase of the toolchain. Through a code generation procedure, the orchestrator could be transformed into

executable code to be deployed on real mobile platforms. This would allow us to test if the robot can effectively accommodate real people's needs and comply with the human-oriented nature of the work. The deployment-ready controller could also be simulated in a 2D/3D environment: beyond the testing purposes, this could also help professionals with a different or non-technical background in visualizing the potential capabilities of the approach.

Aknowledgements

The Italian Ministry of Education, University and Research is acknowledged for the support provided through the Project "Department of Excellence LIS4.0 - Lightweight and Smart Structures for Industry 4.0".
We would also like to thank the reviewers for their comments and suggestions for the improvement of our work.

References

1. HRI Toolchain. `https://github.com/LesLivia/hritoolchain` (2020)
2. Adam, C., Johal, W., Pellier, D., Fiorino, H., Pesty, S.: Social human-robot interaction: A new cognitive and affective interaction-oriented architecture. In: International conference on social robotics. pp. 253–263. Springer (2016)
3. Agha, G., Palmskog, K.: A survey of statistical model checking. ACM Transactions on Modeling and Computer Simulation (TOMACS) **28**(1), 1–39 (2018)
4. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical computer science **138**(1), 3–34 (1995)
5. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical computer science **126**(2), 183–235 (1994)
6. Arai, R., Schlingloff, H.: Model-based performance prediction by statistical model checking an industrial case study of autonomous transport robots. In: Concurrency, Specification and Programming (2017)
7. Askarpour, M., Mandrioli, D., Rossi, M., Vicentini, F.: Formal model of human erroneous behavior for safety analysis in collaborative robotics. Robotics and Computer-Integrated Manufacturing **57**, 465–476 (2019)
8. Askarpour, M., Menghi, C., Belli, G., Bersani, M., Pelliccione, P.: Mind the gap: Robotic mission planning meets software engineering. In: Proceedings of the 8th International Conference on Formal Methods in Software Engineering
9. Bersani, M.M., Soldo, M., Menghi, C., Pelliccione, P., Rossi, M.: Pursue-from specification of robotic environments to synthesis of controllers. Formal Aspects of Computing pp. 1–41 (2020)
10. Chettibi, T., Haddad, M., Lehtihet, H., Khalil, W.: Suboptimal trajectory generation for industrial robots using trapezoidal velocity profiles. In: IROS. pp. 729–735. IEEE (2006)
11. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B.: Uppaal SMC tutorial. STTT **17**(4), 397–415 (2015)
12. Flynn, D., Huang, X., Fisher, M., Papacchini, F., Ferrando, A.: Towards integrating formal verification of autonomous robots with battery prognostics and health management. In: Proc. of Software Engineering and Formal Methods. vol. 11724, p. 105. Springer Nature (2019)

13. Foughali, M., Ingrand, F., Seceleanu, C.: Statistical model checking of complex robotic systems. In: Proc. of SPIN. pp. 114–134. Springer (2019)
14. de Graaf, M.M., Ben Allouch, S., van Dijk, J.A.: Why would I use this in my home? a model of domestic social robot acceptance. Human–Computer Interaction **34**(2), 115–173 (2019)
15. Hadley, M.: A deterministic model of the free will phenomenon. Journal of Consciousness Exploration & Research **9**(1) (2018)
16. Halder, R., Proença, J., Macedo, N., Santos, A.: Formal verification of ROS-based robotic applications using timed-automata. In: International FME Workshop on Formal Methods in Software Engineering (FormaliSE). pp. 44–50. IEEE (2017)
17. Herd, B., Miles, S., McBurney, P., Luck, M.: Quantitative analysis of multiagent systems through statistical model checking. In: Int. Workshop on Engineering Multi-Agent Systems. pp. 109–130. Springer (2015)
18. ISO 12100: Safety of machinery - General principles for design - Risk assessment and risk reduction. ISO (2010)
19. ISO 13482: Robots and robotic devices - Safety requirements for personal care robots. ISO (2014)
20. ISO 13849-1: Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design. ISO (2006)
21. Jaber, M.Y., Givi, Z., Neumann, W.P.: Incorporating human fatigue and recovery into the learning–forgetting process. Applied Mathematical Modelling **37**(12-13), 7287–7299 (2013)
22. Jacobs, T., Virk, G.S.: ISO 13482 - The new safety standard for personal care robots. In: Intl. Symp. on Robotics. pp. 1–6. VDE (2014)
23. Konz, S.: Work/rest: Part ii-the scientific basis (knowledge base) for the guide 1. Ergonomics Guidelines and Problem Solving **1**(401), 38 (2000)
24. Larsen, K.G., Pettersson, P., Yi, W.: Uppaal in a nutshell. vol. 1, pp. 134–152. Springer-Verlag (1997)
25. Luckcuck, M., Farrell, M., Dennis, L.A., Dixon, C., Fisher, M.: Formal specification and verification of autonomous robotic systems: A survey. ACM Computing Surveys (CSUR) **52**(5), 1–41 (2019)
26. Payne, P., Lopetegui, M., Yu, S.: A review of clinical workflow studies and methods. In: Cognitive Informatics, pp. 47–61. Springer (2019)
27. Porfirio, D., Sauppé, A., Albarghouthi, A., Mutlu, B.: Authoring and verifying human-robot interactions. In: Proc. of ACM Symposium on User Interface Software and Technology. pp. 75–86 (2018)
28. Rasouli, A., Tsotsos, J.K.: Autonomous vehicles that interact with pedestrians: A survey of theory and practice. IEEE transactions on intelligent transportation systems (2019)
29. Sinkaram, C., Rajakumar, K., Asirvadam, V.: Modeling battery management system using the lithium-ion battery. In: International Conference on Control System, Computing and Engineering. pp. 50–55. IEEE (2012)
30. Vicentini, F., Askarpour, M., Rossi, M.G., Mandrioli, D.: Safety assessment of collaborative robotics through automated formal verification. IEEE Transactions on Robotics (2019). https://doi.org/10.1109/TRO.2019.2937471
31. Webster, M., Dixon, C., Fisher, M., Salem, M., Saunders, J., Koay, K.L., Dautenhahn, K.: Formal verification of an autonomous personal robotic assistant. In: AAAI Spring Symposium Series (2014)

Appendix A Additional Models

A.1 Human-Leader Model

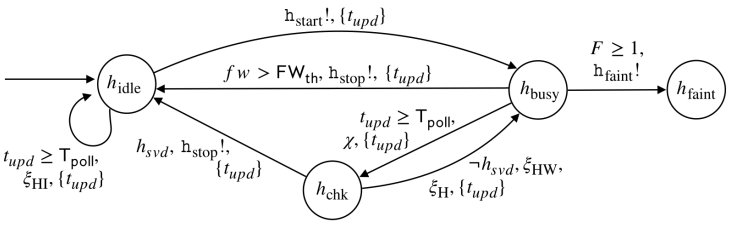


Fig. 10: Human-Leader automaton.

The automaton modeling the second pattern in Table 2 is depicted in Fig. 10. Operating conditions, time-dependent variables and dense counters are the same as the ones described for the **Human-Follower** pattern in Section 5. In this pattern, the human is leading the task, thus they are in charge of checking when the task is complete, and, in that case, end it. This checking mechanism is modeled by location h_{chk}. The automaton enters this location every time the value of clock t_{upd} equals $\mathsf{T}_{\mathsf{poll}}$. The model includes a boolean variable h_{svd}, which has value 1 when the service requested by human h has been provided, 0 otherwise. Upon entering location h_{chk}, the automaton checks whether the destination has been reached or not (update indicated by χ in Fig. 10). If this is the case, h_{svd} is set to true, $\mathtt{h}_{\text{stop}}$ is triggered and the automaton switches back to h_{idle}. Otherwise, the automaton returns to h_{busy} and the human keeps walking. If the human is a leader, their free will manifests itself by stopping even if the destination is yet to be reached, which is modeled through a transition from h_{busy} to h_{idle}, with the guard condition $fw > \text{FW}_{\text{th}}$. Free will is modeled as described in Section 5. Similarly, what is stated about update instructions in Table 3 for the **Human-Follower** pattern stands correct for this pattern as well.

A.2 Human-Recipient Model

Fig. 11 represents the automaton for the **Human-Recipient** pattern. Locations h_{idle} and h_{busy} model the same situations described for the previous two patterns. In this case, there is an additional contingency corresponding to the moment in which the human collects the item from the robot, modeled by location h_{rec}. This occurs when the robot has fetched the item and reached the human location: at this point, the orchestrator triggers $\mathtt{h}_{\text{start}}$ to prompt the synchronization. Event $\mathtt{h}_{\text{stop}}$ is then triggered to signal the end of the task. Free will is also taken into account for this pattern, with the same characteristics described in Section 5. The

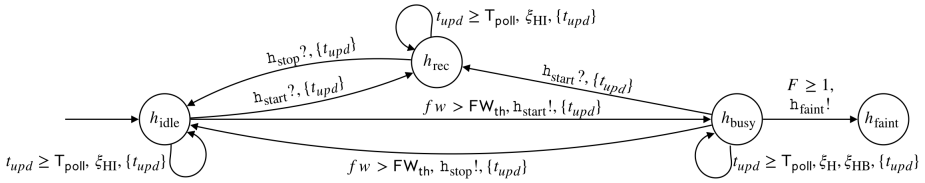


Fig. 11: Human-Recipient automaton.

human can freely decide to walk or stop while the robot is fetching the object. Therefore, two transitions are added between h_{idle} and h_{busy}, enabled by the condition $fw > \mathrm{FW_{th}}$, which trigger events $\mathtt{h}_{\mathrm{start}}$ and $\mathtt{h}_{\mathrm{stop}}$. All variables (time-dependent, dense counters and parameters) have the same semantics described for the first pattern.

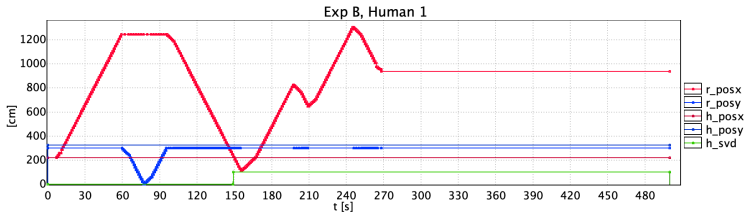
Appendix B Additional Experiments

The additional experimental setting, presented in the following, demonstrates how the two patterns in Appendix A work in practice, and how the robot battery is managed by the orchestrator. The floor layout is the same as in Fig. 9a. In this case study, the robot needs to serve two humans: one adhering to the **Human-Recipient** (see A.2) pattern, and one to the **Human-Leader** pattern (see A.1). The specific parameters for the first human are: $\mathrm{v} = 15\mathrm{cm/s}$, $\mathrm{p_f} = 3$ and $\mathrm{dest} = (1250, 100)$. Since this is a recipient pattern, in this case the dest parameter represents the location of the item needed by the human. For the second human, they are: $\mathrm{v} = 10\mathrm{cm/s}$, $\mathrm{p_f} = 1$ and $\mathrm{dest} = (1250, 500)$. The robot has the same characteristics as the one used for the experiment in Section 6, but we are going to run two versions of the experiment: one with $\mathrm{C_{start}} = 100\%$ and one with $\mathrm{C_{start}} = 11\%$.
The experiments are run on the same machine used for the experiment in Section 6, with the same version of Uppaal and the same statistical parameters. Performance data can be found in Table 6.

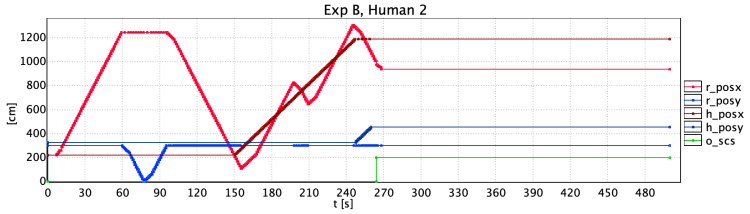
Fig. 12 and Fig. 13 show two simulation traces, one for each value of $\mathrm{C_{start}}$. When the battery is fully charged, the robot immediately moves towards the location of the object to fetch and then returns to the location of the first human, who never moves from the starting point (Fig. 12a). As in Fig. 12a, the

Table 6: Experiments Performance Data

Exp.	**States**	**Time**	**Virtual Memory** [KiB]	**Resident Memory** [KiB]
$\mathrm{C_{start}} = 100\%$	1138343	$\approx$ 4.5min	166972	126304
$\mathrm{C_{start}} = 11\%$	75339267	$\approx$ 6.5hr	166968	124752



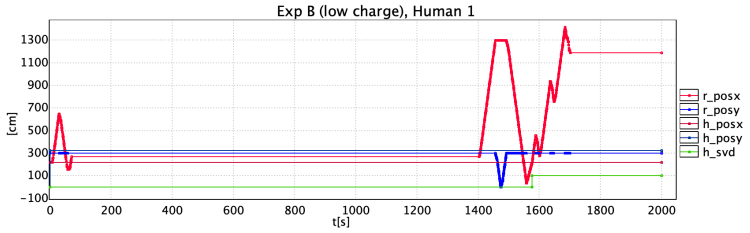
(a) First human served (recipient).



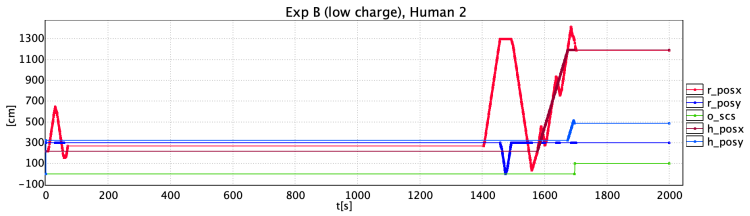
(b) Second human served (leader).

Fig. 12: Simulation trace of the experiment with $\mathrm{C}_{\mathrm{start}} = 100\%$: robot trajectory is in both plots, human 1 is in (a) and human 2 in (b).

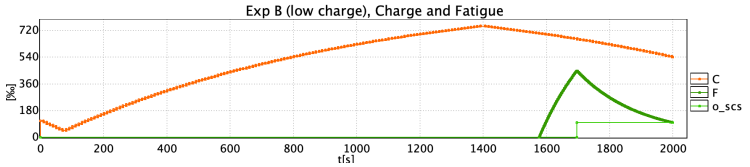
first human is served ($\mathrm{h}_{\mathrm{svd}} = 1$) at $t = 150s$. Fig. 12b shows the behavior of the second interaction pattern: since the second human is a leader, they immediately start moving as soon as their turn comes. The robot follows and, when it ends up ahead of the human (e.g., $t \approx 190s$), it steps back and resumes the trailing. The whole mission ends successfully after approximately $270s$ (see Fig. 12b).
In the second case with low battery charge, the robot initially starts moving towards the location of the object, but, as soon as $b_{ch} < \mathrm{B}_{\mathrm{th}_1}$, with $\mathrm{B}_{\mathrm{th}_1} = 10\%$, the orchestrator orders it to stop and start moving towards the recharge station instead. This is allowed since the human is a recipient, thus the robot is entitled to start and stop the action whenever necessary. The recharging phase lasts until $b_{ch} > \mathrm{B}_{\mathrm{th}_2}$, with $\mathrm{B}_{\mathrm{th}_2} = 70\%$ for this experiment ($t = 1400s$), as in Fig. 13c. When the battery has sufficiently recharged, the robot resumes all its operations: it fetches the item from point (1250, 100), brings it to the first human ($t \approx 1600s$), then follows the second human to the destination. The whole mission ends successfully at $t \approx 1700s$.
The verified property in both cases is $P^{\leq\tau}_{\geq\theta}(\diamond\, o_{\mathrm{success}})$. With $\mathrm{C}_{\mathrm{start}} = 100\%$, the property is verified with $\tau = 500s$ and $\theta = 0.9$. With $\mathrm{C}_{\mathrm{start}} = 11\%$, we have verified that $P(\diamond\, o_{\mathrm{success}}) \in [0.4, 0.5]$ with $\tau = 2000s$.
The experiment demonstrates how the orchestrator successfully manages the battery recharge policy to prevent the failure of the mission, even though this causes a general slowdown in service provision.



(a) First human served (recipient).



(b) Second human served (leader).



(c) Robot battery charge and human 2 fatigue.

Fig. 13: Simulation trace of the experiment with $C_{start} = 11\%$: robot trajectory is in both plots, human 1 is in (a) and human 2 in (b), battery charge and fatigue of human 2 in (c).