A Model-driven Approach for the Formal Analysis of Human-Robot Interaction Scenarios

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

***Abstract*—Robots are currently mostly found in industrial settings. In the future, a wider range of environments will benefit from their inclusion. This calls for the development of tools that allow professionals to set up dependable robotic applications in which people productively interact with robots aware of their needs. Given the co-existence of humans and robots, the precise analysis—e.g., through formal verification techniques—of properties related to aspects such as human needs and physiology is of paramount importance. In this paper, we present a formally-based, model-driven approach to design and verify scenarios involving human-robot interactions. Some of the features of our approach are tailored to the healthcare domain, from which our case studies are derived. In our approach, the designer specifies the main parameters of the mission to generate the model of the application, which includes mobile robots, the humans to be served, including some of their physiological features, and the decision-maker that orchestrates the execution. All components are modeled through hybrid automata to capture variables with complex dynamics. The model is verified through Statistical Model Checking (SMC), using the Uppaal tool, to determine the probability of success of the mission. The results are examined by the developer, who iteratively refines the design until the probability of success is satisfactory.**

***Index Terms*—service robots, human-robot interaction, model-driven development, statistical model checking**

I. Introduction

Over the last few decades, robots have been affecting humankind from behind the scenes. They have been mostly employed in industrial settings, contributing to applications that require a lot of strength or extreme working speed. Nonetheless, thanks to the recommendations advocated by the Industry 4.0 paradigm, robots have recently been allowed to work in close contact with human workers to support them with dull or fatiguing tasks. The next step is to let robots out of factories and into service contexts, to relieve practitioners in these areas from clerical or overly hazardous duties, improving their work conditions. This could prove a leap forward for several domains, such as healthcare and emergency response. Deploying robots in workplaces different from a factory poses a series of challenges. People do not walk around wearing protection devices in everyday situations, and they are usually not trained to safely interact with this kind of machinery. Although general safety guidelines provided by ISO 12100 [1] and ISO 13849-1 [2] still apply, ISO 13482 [3] specifically targets service robots from the following categories: mobile platforms, person carrier robots, and physical assistant robots [4]. These regulatory documents will be instrumental in bringing products to market that are designed to operate safely also in civil contexts. Once the robot is deemed fit for deployment, additional effort is required to guarantee an effective interaction with humans. Users will expect robots to respond to their needs as effortlessly as a human worker in the same position would do. Otherwise, society will perceive this injection of robotics as a step back rather than an improvement. In this paper, we present a formally-based model-driven approach for the analysis of robotic applications. Each application features robots, humans and a robot-controller, and it involves interactions among the agents in a defined environment. Though the approach is general, it is tailored towards the healthcare domain, in that it considers some physiological features of humans. We have identified a set of significant interaction patterns that capture usual coordination contingencies established between a human and a robot. The robot-controller included in the model is necessary to realize such coordination. The patterns differ from each other based on a series of criteria—e.g., who initiates synchronization or who is entitled to end it. Moreover, they are meant to be customized and reused with little effort by the designer. In fact, creating a scenario only involves the specification of how many agents are in it, their basic properties—e.g., speed and acceleration—and the floor layout. The model includes healthcare-related aspects, such as physiological features of humans. The robot-controller drives the mission to success based on efficiency-related criteria, but also on human needs. This is paramount when robots interact with humans who are in pain or in discomfort, as it often happens in healthcare settings. A key aspect of the work is that humans are not modeled as rational agents that unmistakably execute their mission; rather, they are described through a stochastic model of free will that causes them to occasionally stray from the plan. Hence, designers have a formal guarantee that the deployed solution will be able to deal with unexpected turns of events.

The approach relies on Hybrid Automata [5] to model application components. The automata are endowed with both differential equations describing the complex dynamics of the system, and stochastic elements capturing the variability of the real world. The model is formally verified through Statistical Model Checking techniques [6] against a set of relevant properties. The toolchain automatically generates the model of the scenario and performs the verification using the Uppaal tool [7] [8]. To show the effectiveness of the approach, we present a range of experiments involving different scenarios, which encompass cases in which the mission is accomplished,

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

and cases in which it fails. The latter show how the designer can iteratively refine the application model, until a satisfactory result is obtained. The experiments are run using a prototype of the tool [9], which receives in input the set of parameters, then automatically generates the model and runs the verification.
The paper is structured as follows: Section II reviews related works in the literature; Section III outlines the tools on which the work is based; Section IV presents the overall approach; Section V describes in detail the developed models; Section VI shows experimental results; Section VII concludes.

II. Related Work

Many works in literature deal with human-robot interaction modeling. Yagoda and Coovert [10] analyze the case study of a team of three individuals that operate a miniature UAV searching for survivors in the aftermath of a disaster. They select Petri nets as the modeling tool since it is a formal and graphical language, fit for analysis and explicit-state modeling. Other works implement formal languages to capture collaboration: for example, Webster et al. [11] develop Brahms models of a robotic personal assistant and then perform model-checking with the tool SPIN. Vicentini et al. [12] use LTL formulae to model collaborative tasks and develop an innovative risk assessment technique. Some works exploit learning techniques to develop their model. Amor et al. [13] identify interaction primitives through imitation learning. This type of approach is particularly useful when the robot is required to acquire a certain behavior from the human and reproduce it to make the interaction more natural. Similarly, Nikolaidis et al. [14] aim at programming robots to learn the human type of their teammate from a dataset and adopt their preferred style. However, this is only feasible in settings where the range of possible behaviors the robot can get in contact with has a limited size. Lemaignan et al. [15] present an architecture for the robot decisional layer based on models of human behavior and human preferences. In more detail, the authors build upon the Beliefs, Desires, Intentions (BDI) infrastructure to develop social robots that can work jointly with humans. Araiza et al. [16] also exploit BDI agents to model the human, the robot and their environment using a table assembly task as a possible application. In this case, models are used to automatically generate test cases through model checking.
There are also previous attempts at applying SMC to robotic systems. Arai and Schlingloff [17] exploit SMC to make predictions on the performance of autonomous transport robots in production plants. They show how the approach is beneficial when making decisions at design time. Foughali et al. [18] apply SMC to formally verify real-time properties, like schedulability and readiness, of robotic software. Herd et al. [19] explore the area of multi-agent systems, focusing on swarm robotics: in this case, SMC is needed to deal with the size of the problem which makes it unfeasible for traditional model checking techniques.
As this brief survey shows, modeling and verifying systems that involve robots coordinating with humans is a long-standing issue. Though it has been tackled through techniques from several different domains, no works combine sound formal methods with a proper model of real-world variability. Formal verification has been applied to specific manifestations of human behavior, which are handled in a black-box manner in the work by Askarpour et al. [20]. However, to the best of the authors' knowledge the work in this paper is the first attempt at analyzing human-related aspects, such as free will. In addition, as mentioned above, service robots will be used in applications designed by people with potentially many different backgrounds, who cannot be expected to be able to create complex models while designing their application. Hence, the user-friendliness of the design tool is of paramount importance. The work presented in this paper is a first step towards filling these gaps.

III. Background

In this work we use, as the modeling formalism, Hybrid Automata (HA) extended with a stochastic component (abbreviated as HA+ hereinafter). HA are themselves an extension of Timed Automata. In Timed Automata, the passage from one state to another is governed by conditions on *clocks*, whose value increases linearly with time unless they are reset [21]. HA locations are additionally endowed with sets of differential equations, called *flow conditions*. These constrain the derivatives of real-valued variables of the model, which make it possible to model systems with complex dynamics [5]. A model can include several distinct automata, that can synchronize with each other through *channels*. When two automata synchronise on a channel e, two transitions are fired at the same time, and the current location of both automata changes synchronously. In particular, the two transitions synchronise if their guards are both satisfied, and they have complementary synchronisation labels of the form $e?$ and $e!$, respectively. The stochastic behavior can be modeled through probabilistic transitions. These represent non-deterministic choices of the system, refined by probability distributions. As in Fig. 1, these transitions are marked with probability weights, that bias the evolution of the system [6]. Enriching HA with probabilistic features makes them analyzable through statistical techniques, and in particular through Statistical Model Checking (SMC). The advantage of SMC is that it does not explore the whole state space, but it applies statistical techniques to a sample set of executions to evaluate the probability that a certain property holds. This makes the analysis of the property feasible for

TABLE I: PCTL Syntax

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

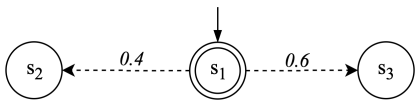


Fig. 1: Example of automaton with a stochastic transition: 0.4 and 0.6 represent the probabilities of reching s_2 or s_3 from s_1.

large and complex systems [6]. The approach also requires a formal specification of the properties to verify. The formulae are expressed in the PCTL logic, whose syntax is shown in Table I. The most significant difference with ordinary CTL is the $P_{\geq\theta}(\phi)$ operator, which allows us to express properties with quantitative constraints on probabilities [6]. 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 [6]. Section VI shows various examples of SMC experiments.

The features of HA+ are used in Section V to model the physiological properties of humans and the robot velocity profile. The human fatigue model used in our work is the one proposed by Jaber et al. [22], with alternating exponential fatigue/recovery cycles. As for the robot velocity, we implement a trapezoidal velocity profile [23] with three phases: acceleration, constant maximum speed, and deceleration. For the robot battery, we assume a typical lithium battery charge/discharge cycle [24] with an exponential and a nominal zone.

IV. Approach

The main contribution of this paper is a model-driven approach for the formal analysis of scenarios involving human-robot interaction. The toolchain is meant to be used by professional figures who may possess a technical background, but not necessarily in robotics or in formal methods. Users may be, for example, clinical workflow analysts [25] designing the following work shift for robots. The shift will involve a group of humans requesting a service that implies interaction with a robot. Serving everybody in the group will constitute the mission of the robot. Fig. 2 shows the steps, further described in the following, on which the design process is built: (A) configuration of the scenario, performed by the designer; (B) automated model generation and its verification through SMC; (C) critical assessment of the verification results, followed by application deployment if the results are satisfactory, otherwise by model refinement. In this paper, we focus mostly on the formal modeling, whereas the model generation phase will be deepened in future works.

A. Scenario Configuration

The input of the tool includes the following parameter values, depicted in Fig. 3.

A scenario may include multiple humans that need to be served and multiple available robots. Each robot moves with a fixed speed and acceleration ($\mathrm{v_{max}}$ and $\mathrm{a_{max}}$ in Fig. 3), and initial battery charge $\mathrm{C_{start}}$. An association between a robot and a battery constitutes a robotic system with its own id. Each human is served by one robot and in the model is identified by an id, their walking speed v and one interaction pattern p, which characterizes the interaction with the robot. Humans are served in ascending order of id.

The identified patterns correspond to recurrent synchronization mechanisms in case studies involving a particular subset of

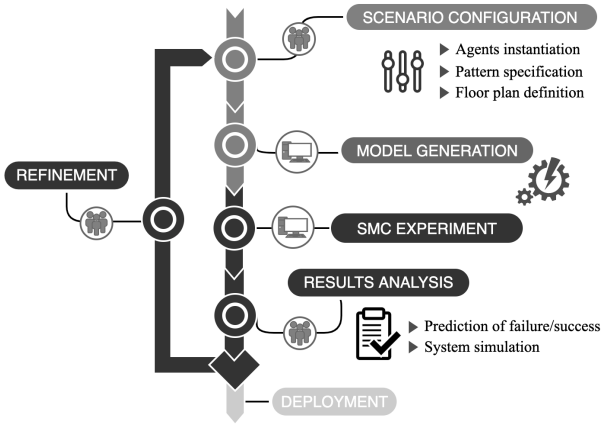


Fig. 2: Diagram representing all the phases of the approach. Different shades of gray indicate the current progress of implementation: the darker phases are fully implemented, the lighter ones are to be expanded in the future.

human-robot interaction, suitable for mobile robots with a predefined set of functionalities. The main elements that set one pattern from another are: who triggers the action, who can claim its completion and the condition that determines it, and how the state of the system evolves during the execution. In detail, the patterns are:

(i) **Human-Follower**: the human follows the robot and the robot moves towards the destination. They stop when they have reached it. For the sake of free will, the human is allowed to walk freely and choose whether to follow or not when the robot issues its command. If they get too far, the robot stops and waits for them. Robots could guide patients through an infectious ward, reducing the exposure of healthcare workers.
(ii) **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, the robot follows accordingly. If the robot ends up ahead of the human, it steps back and resumes the trailing. The robot could follow a nurse, identified by a wearable sensor, while carrying tools.
(iii) **Human-Recipient**: the human, a nurse or a doctor, waits for the robot to fetch an item, possibly a needed tool, from a certain location. While the robot is delivering the object, the human is free to move and the robot should be able to track them. The real synchronization occurs when the human and

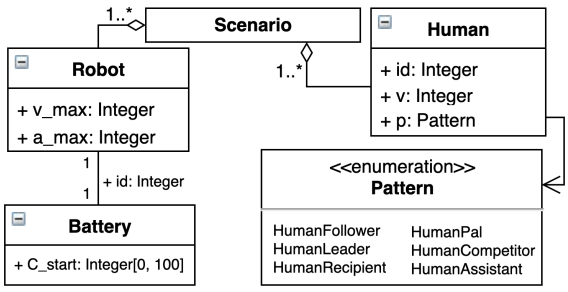


Fig. 3: Class Diagram of the user-customizable portion of the model.

the robot are close and the human stops to pick up the item.
(iv) **Human-Pal**: both the robot and the human execute an action until its ending condition, e.g., a nurse and a robot look for something and stop when either one finds it. Both the robot and the human are free to start and stop. The synchronization occurs when one signals that the task has been completed.
(v) **Human-Competitor**: the robot and the human compete over a resource. The action is over when the resource is no longer available, i.e., when either of them has reached the resource location. An example could be a doctor and a robot competing over a medical kit: ideally, both of them should be put in a position to access the item.
(vi) **Human-Assistant**: whenever a certain situation occurs, the robot stops and waits for human assistance to proceed. For example, if a closed door is detected along the trajectory, the robot emits a signal and waits for a human to open it. The human signals to the robot when their action is completed.

In this paper, we describe in detail the implementation of (i), and present experiments with (i), (ii) and (iii). Given the *plug-and-play* nature of the approach, the remaining implementations can be added to the model in a second phase without undermining the conclusions drawn about the infrastructure.
The operational environment is modeled as a two-dimensional layout. The designer only specifies the Cartesian coordinates of the points of interest, i.e., wall corners and doorposts. It is also mandatory to specify the destinations of all the humans in the model. The algorithm that simulates the robot navigation accounts for these variables to prevent collisions, which would lead to unreliable results.

B. Model Generation and SMC Experiments

After the configuration phase, the designer can automatically generate the model and run the SMC experiment. The user must additionally choose whether the experiment should involve a simulation of the system or an estimation of the probabilities of success/failure of the mission. Section VI presents three examples of experiments.

C. Results Analysis and Model Refinement

The SMC experiment yields a probability value. If the probability of success is smaller than a desirable threshold, the user may opt for one (or more) of the following corrective measures, therefore performing model refinement: (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 one service and the next one; (c) choose different services to be included in the scenario (this may not always be an option, for example if a patient has to be mandatorily served in the following shift); (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.

V. Model

We present the HA+ modeling the components of the scenario: robots, batteries, humans and the orchestrator. The latter directs the synchronization among all the other components by sending commands to the targets through *channels*. The orchestrator makes decisions based on data about the other components, and such data are modeled via *dense counter* variables. These change over time as a result of transitions *update* instructions, but they do not possess an explicit time-dependency. This mechanism simulates a sensor measuring a particular physical property (the dense counter) whose value is periodically updated. In all components, the refresh period is indicated with the constant $\mathrm{T_{poll}}$ and a clock t_{upd} measures the time elapsed since the last update. We assume the building is equipped with an Indoor Positioning System (IPS) [26] to locate both the robot and the human in the environment. It is also required that the battery charge value can be periodically measured, as it is customary with lithium batteries in electronic devices. We also assume that the human is wearing a fatigue-measuring device, as the one proposed by Dong et al. [27]. As mentioned in Section III, the robot movement follows a trapezoidal velocity profile, whereas the human moves with constant speed. As for human free will, we have opted for the straightforward approach [28] 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.

Robot: the automaton 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 $\dot{V} = \mathrm{a_{max}}$; 3) the travel phase with constant speed, $V = \mathrm{v_{max}}$; 4) the deceleration phase with $\dot{V} = -\mathrm{a_{max}}$; 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 ($\mathtt{b}_{\mathrm{start}}$ and $\mathtt{b}_{\mathrm{stop}}$), 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

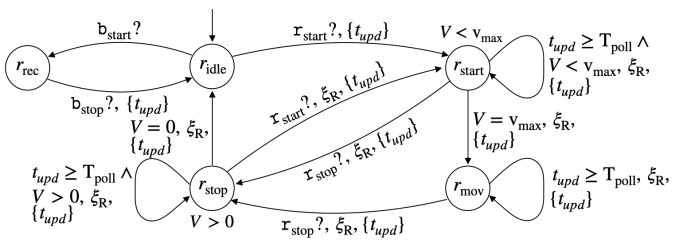


Fig. 4: Robot automaton.

($\mathtt{r}_{\mathrm{start}}$) is issued. Similarly, the automaton switches from r_{mov} to r_{stop} when $\mathtt{r}_{\mathrm{stop}}$ is fired. The robot might start or stop while it is accelerating or decelerating, so two transitions are added between r_{start} and r_{stop}. In r_{start}, velocity is increasing linearly from 0 to $\mathrm{v_{max}}$, thus when $V = \mathrm{v_{max}}$ 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 self-loops, and on all transitions in Fig. 4 marked with a ξ_{R}, the update in Eq.1 is executed.

$$\xi_{\mathrm{R}} = \begin{cases} r'_{pos_x} := r_{pos_x} + V(t)\mathrm{T_{poll}} \cdot cos(\theta_r) \\ r'_{pos_y} := r_{pos_y} + V(t)\mathrm{T_{poll}} \cdot sin(\theta_r) \end{cases} \quad (1)$$

Robot Battery: the automaton representing the behavior of the battery is pictured in Fig. 5. The time-dependent variable in the model is the charge value C. The charge and discharge cycles are approximated assuming 3 phases for each cycle. A phase is characterized by a coefficient that defines how C will change with time, and a charge threshold $\mathrm{C_{th}}$ that determines when the the following phase must start. The identified discharge phases, modeled by as many locations, are: 100% to 80% ($b_{100,80}$), 80% to 20% ($b_{80,20}$), 20% to 0% ($b_{20,0}$). The dual locations modeling the charging phases are: $b_{0,20}$, $b_{20,80}$, $b_{80,100}$. The coefficients in the linear time-dependency correspond to parameters: $-\mathrm{r}_1$ and r_1 in $b_{100,80}$ and $b_{20,80}$, $-\mathrm{r}_2$ and r_2 in $b_{80,20}$ and $b_{80,100}$, $-\mathrm{r}_3$ and r_3 in $b_{20,0}$ and $b_{0,20}$. The location b_{empty} models the situation where the battery is fully discharged and the robot cannot move again without human intervention. The switch from a *discharging* location to the corresponding *charging* one and viceversa occurs when the orchestrator decides that a robot needs to start or stop recharging, firing events $\mathtt{b}_{\mathrm{start}}$ and $\mathtt{b}_{\mathrm{stop}}$. The automaton includes a dense counter for the battery charge (variable b_{ch}). The update can be found in Eq.2, where $\mathrm{r_x}$ is the charge/discharge coefficient for the current phase.

$$\xi_{\mathrm{B}} = b'_{ch} := b_{ch} + \mathrm{r_x} \cdot \mathrm{T_{poll}}, \quad \mathrm{r_x} \in \{\pm \mathrm{r}_1, \pm \mathrm{r}_2, \pm \mathrm{r}_3\} \quad (2)$$

Human-Follower: the model of the first pattern introduced in Section IV-A is depicted in Fig. 6. We capture two operating conditions for the human: 1) location h_{idle} models the case in which the human is still; 2) location h_{busy} corresponds to the situation in which the human is moving. 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. Time-dynamics are given in Eq.3: h_{dist} is either constant or 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 introduced in [22], as explained in Section III.

$$h_{\mathrm{idle}} = \begin{cases} \dot{F} = -\mathrm{F}(\tau)\mu e^{-\mu t} \\ \dot{h}_{dist} = 0 \end{cases} \quad h_{\mathrm{busy}} = \begin{cases} \dot{F} = \mathrm{F}(\tau)\lambda e^{-\lambda t} \\ \dot{h}_{dist} = \mathrm{v} \end{cases} \quad (3)$$

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 $\mathtt{h}_{\mathrm{start}}$ or $\mathtt{h}_{\mathrm{stop}}$. 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*, thus staying in the same location. The transition is governed by the two constant weights obey and $\mathrm{disobey}$. As for the second case, sample points of the free will random variable X are observations of local variable fw. A success corresponds to the decision of the human to start or stop freely, which occurs when $fw > \mathrm{FW_{th}}$, where $\mathrm{FW_{th}}$ is a constant threshold. Since fw is updated every $\mathrm{T_{poll}}$ instants with a random value $\in [0, FW_{\max}]$, the probability of making an autonomous decision is $E[X] = p = 1 - \frac{\mathrm{FW_{th}}}{\mathrm{FW_{max}}}$. Therefore, 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. Therefore, it is reached when $F \geq 1$, where 1 corresponds to the maximum value of fatigue, and it causes the *urgent* channel [7] $\mathtt{h}_{\mathrm{faint}}$ to immediately fire. The dense counters are $h_{fatigue}$, and h_{pos_x}, h_{pos_y}, θ_h for Cartesian coordinates and orientation of the human with respect to the x-axis. The update is given in Eq.4: $h_{fatigue}$ adheres to the model in Eq.3, while h_{pos_x}, h_{pos_y} are the projections of the displacement since last update.

$$\xi_{\mathrm{H}} = \begin{cases} h'_{fatigue} := 1 - (1 - h_{fatigue})e^{-\lambda \mathrm{T_{poll}}}, & \text{if } \mathrm{h_{busy}} \\ h'_{fatigue} := h_{fatigue} e^{-\mu \mathrm{T_{poll}}}, & \text{if } \mathrm{h_{idle}} \\ h'_{pos_x} := h_{pos_x} - \mathrm{v}\mathrm{T_{poll}} \cdot cos(\theta_h) \\ h'_{pos_y} := h_{pos_y} - \mathrm{v}\mathrm{T_{poll}} \cdot sin(\theta_h) \end{cases} \quad (4)$$

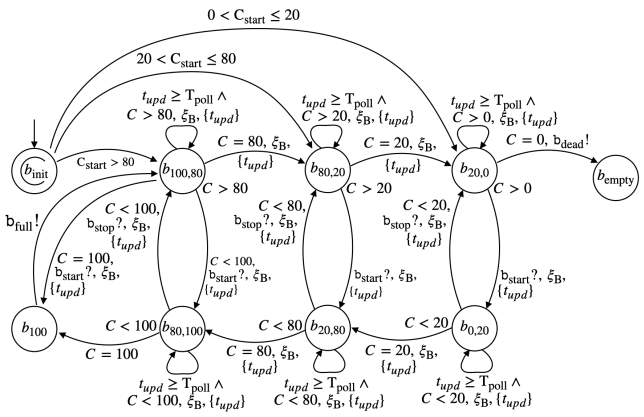


Fig. 5: Battery automaton.

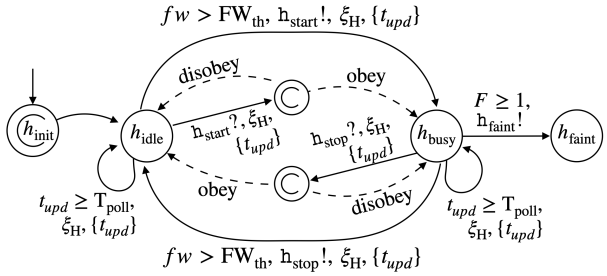


Fig. 6: Human-Follower automaton.

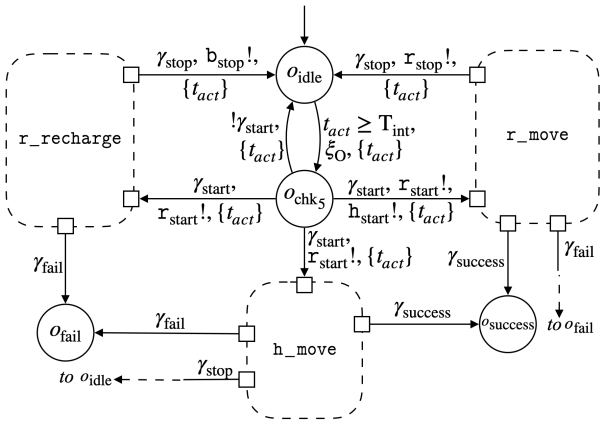


Fig. 7: Orchestrator automaton.

Orchestrator: the automaton is displayed in Fig. 7. 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_recharge` controls the recharging phase of the robot; `r_move` controls the start and the end of the robot 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 scenario. We describe the high-level behavior of the automaton, without delving into the details of the individual sub-machines. Sub-machines in Fig. 7 are endowed with *ports*: these are not part of the formalism, and should only be interpreted as visual representation of the transitions entering and leaving the component. The behavior of the orchestrator is governed by clock t_{act} that triggers the system monitoring routine (identified by ξ_{O} on the transition to o_{chk_5} in Fig. 7) every T_{int} time instants. If one of the guards γ_{start} in Eq.5 is true when the automaton is in location o_{chk_5}, the corresponding sub-automaton is entered, otherwise it goes back to o_{idle}.

$$\gamma_{\text{start}} = \begin{cases} r_{idle} \wedge b_{ch} < \mathrm{B}_{\text{th}_1} & (\mathtt{r\_recharge}) \\ r_{idle} \ \wedge \ \neg human\_leader_{id} & (\mathtt{r\_move}) \\ h_{busy} \ \wedge \ human\_leader_{id} & (\mathtt{h\_move}) \end{cases} \tag{5}$$

The recharging routine starts when a robot is idle and its current battery charge b_{ch} is below a threshold B_{th_1}. The `r_move` sub-machine is entered to initiate the robot movement, when the robot is idle and the human is not a leader. If the human is a leader and starts moving, hence its current location is h_{busy}, the orchestrator enters sub-machine `h_move`. Upon entering `r_recharge`, the orchestrator fires $\mathtt{r}_{\text{start}}$ to force the robot to reach the recharge station, then $\mathtt{r}_{\text{stop}}$ and $\mathtt{b}_{\text{start}}$ when the dock has been reached and the robot can start recharging. Upon entering `r_move`, the orchestrator fires $\mathtt{r}_{\text{start}}$, and $\mathtt{h}_{\text{start}}$ if the human is a follower. Upon entering `h_move`, $\mathtt{r}_{\text{start}}$ is triggered so that the robot can follow the human. When one of the guards γ_{stop} in Eq.6 is true, the orchestrator switches to o_{idle} from a location in the active sub-machine.

$$\gamma_{\text{stop}} = \begin{cases} b_{ch} > \mathrm{B}_{\text{th}_2} \wedge r_{rec} & (\mathtt{r\_recharge}) \\ h_{fatigue} > \mathrm{H}_{\text{th}_1} \vee b_{ch} < \mathrm{B}_{\text{th}_1} & (\mathtt{r\_move}) \\ h_{idle} & (\mathtt{h\_move}) \end{cases} \tag{6}$$

The robot stops recharging, thus $\mathtt{b}_{\text{stop}}$ is fired, when variable b_{ch} is above a desirable threshold B_{th_2}. The robot stops moving (channel $\mathtt{r}_{\text{stop}}$) when human fatigue $h_{fatigue}$ exceeds a maximum tolerable value, described by constant H_{th_1}, or when the battery charge drops below the value B_{th_1} that calls for recharging. Finally, the orchestrator exits sub-machine `h_move` when the human stops moving, and it terminates the robot motion with event $\mathtt{r}_{\text{stop}}$. The two locations o_{fail} and o_{success} correspond to the end of the whole mission with failure or success. The condition to reach o_{fail} can be found in Eq.7.

$$\gamma_{\text{fail}} = b_{ch}(t) \leq 0 \ \vee \ h_{fatigue}(t) \geq 1 \tag{7}$$

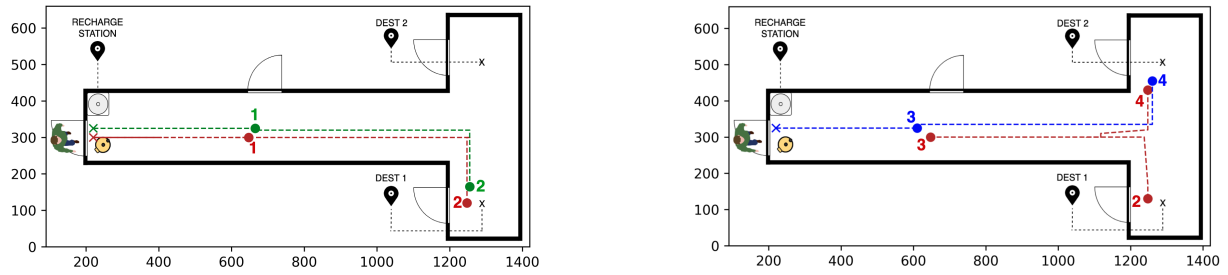
Failure occurs if the battery charge drops to 0, hence the robot cannot recover autonomously, or if the human fatigue exceeds 1. Location o_{success} is reached when condition $\gamma_{\text{success}} = \forall_h \ h_{served}$ is true. While running the system monitoring routine, the orchestrator checks whether the current service has been completed: if so, the boolean variable h_{served} is set to 1, 0 otherwise. The mission is accomplished when all humans in the scenario have been served. Therefore, only `r_move` and `h_move` have outgoing transitions towards o_{success}, since recharging the robot does not contribute to the cause.

VI. Experimental Results

We present experimental results that prove the effectiveness of our approach. We have selected the JSON format for the user input. The instances for each experiment are specified using a constructor-like notation, with the classes and attributes in Fig. 3: $Human(\mathrm{id}, \mathrm{v}, \mathrm{p})$, $Robot(\mathrm{id}, \mathrm{v}_{\max}, \mathrm{a}_{\max})$, $Battery(\mathrm{id}, \mathrm{C}_{\text{start}})$. Note that v and $\mathrm{v}_{\max}$ are expressed in [cm/s], $\mathrm{a}_{\max}$ in [cm/s^2], and $\mathrm{C}_{\text{start}}$ in percentage points. 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 [7] [8]. In this work, we use Uppaal, version 4.1.24, to implement the automata, and run SMC experiments. As for the latter, we keep the default set of statistical parameters. For each of the experiments presented in the following, we want to determine the probabilities of mission success and failure, i.e., the PCTL formulae in Eq.8 with probability bounds θ_1, θ_2 and time-bound τ which represents the maximum length of paths.

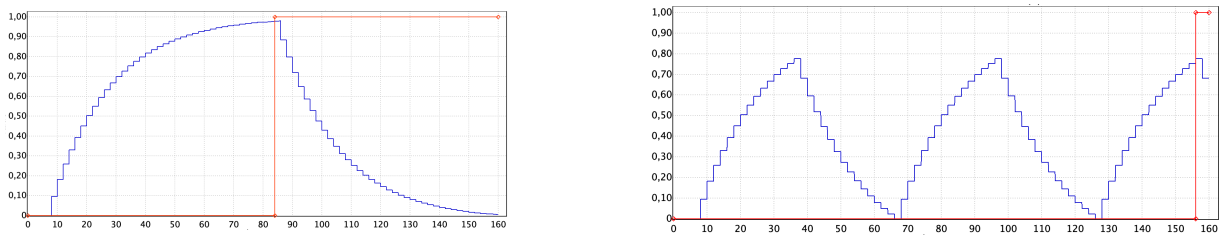
$$P^{\leq \tau}_{\geq \theta_1}(\diamond \ o_{fail}), \quad P^{\leq \tau}_{\geq \theta_2}(\diamond \ o_{success}) \tag{8}$$

The experiments are run on a machine equipped with 128 cores, 515GB of RAM and Debian Linux version 10. Performance data can be found in Table II.



(a) (b)

Fig. 8: Diagrams representing the outcome of Experiment 1 with sufficient battery charge. The red line is the robot trajectory, whereas the green and blue lines correspond to h_1 and h_2 trajectories. The x marks the initial locations. Intermediate points are marked with • and numbered in order of occurrence.



(a) (b)

Fig. 9: Plots of Experiment 2 run without fatigue monitoring in (a), and with the active orchestrator in (b). The blue line represents the human fatigue, whereas the red steps correspond to mission failure in (a), and success in (b).

The first experiment is run with the floor layout in Fig. 8, and the following configuration: (a) h_1: $\mathrm{id} = 1, \mathrm{v} = 5, \mathrm{p} = \mathrm{leader}$ with destination $(1300, 100)$; (b) h_2: $\mathrm{id} = 2, \mathrm{v} = 10, \mathrm{p} = \mathrm{leader}$ with destination $(1300, 500)$; (c) r_1: $\mathrm{id} = 1, \mathrm{v}_{\max} = 20, \mathrm{a}_{\max} = 5$; (d) b_1: $\mathrm{id} = 1, \mathrm{C}_{\mathrm{start}} = 20$. Moreover, we set $\lambda = \mu = 0.005$ (see Eq.3) for full exhaustion in approximately $15min$ of non-stop walking, and $\mathrm{r}_1 = 0.035, \mathrm{r}_2 = 0.008, \mathrm{r}_3 = 0.055$ which amount to a full discharge cycle in approximately $2.5h$. With this value of charge, the property in Eq.8 is verified with $\theta_1 = 0.9$ and $\tau = 220s$. This occurs because the battery charge drops to 0 before both humans have had the time to reach their destinations, as per the failure condition in Eq.7. In this case, the orchestrator cannot stop the motion to recharge the robot because the human is leading. The robot is free to start moving towards the recharge station (in $(250, 400)$ as in Fig. 8) only after h_1 has been served. At that point, the distance to cover exceeds the battery capacity leading to mission failure. The refinement chosen by the user could be the deployment of a different robot with more battery charge, thus: (a) b_1: $\mathrm{id} = 1, \mathrm{C}_{\mathrm{start}} = 50$, while all the other elements are left unchanged. With this modification, the property in Eq.8 is verified with $\theta_2 = 0.9$ and $\tau = 280s$. Fig. 8 depicts the resulting trajectories: h_1 reaches the destination point (point 2 in Fig. 8a), followed by the robot, and then h_2 can start. The robot then correctly starts from its current position (point 2 in Fig. 8b), and moves back towards h_2. When they meet (point 3 in Fig. 8b) the robot starts following h_2 until they reach the destination (point 4). This proves that the orchestrator is able to manage the robot recharging policy coherently with the interaction patterns. It also shows that the failure of the mission can be predicted, but the user can refine the model to obtain a successful result.

For the second experiment, we have used the same floor plan in Fig. 8 and the following setting: (a) h_1: $\mathrm{id} = 1, \mathrm{v} = 19, \mathrm{p} = \mathrm{follower}$ with destination $(1300, 500)$; (b) r_1 and b_1 are the same as for the first experiment. We set $\lambda = \mu = 0.05$, which cause full-exhaustion in about $90s$ of non-stop walking. In this case, we have tweaked the fatigue model parameters and temporarily disabled the monitoring routine of the orchestrator to show that this causes the failure of the mission due to human over-exhaustion. In particular, the property in Eq.8 is satisfied with $\theta_1 \in [0.874, 0.974]$ and $\tau = 100s$. If the orchestrator is re-activated, the property in Eq.8 is verified with $\theta_2 \in [0.77, 0.87]$ and $\tau = 160s$, thus the chances of success rise to an average of 82%. As Fig. 9(a) shows, in the first case the human is subject to a single fatiguing cycle since the robot is never forced to stop. The second plot in Fig. 9(b) shows several fatigue/recovery cycles, which means that the orchestrator stops the robot when the fatigue exceeds threshold $\mathrm{H}_{\mathrm{th}_1} = 0.7$. These results prove that the orchestrator is properly designed and that it is indispensable to achieve the goals laid down for this work.
In the last experiment, we show a successful execution of the patterns *Human_Follower* and *Human_Recipient*. The setting is the following, the floor plan is the same as in previous cases: (a) h_1: $\mathrm{id} = 1, \mathrm{v} = 15, \mathrm{p} = \mathrm{follower}$ with destination $(700, 300)$; (b) h_2: $\mathrm{id} = 2, \mathrm{v} = 10, \mathrm{p} = \mathrm{recipient}$ with object location $(1300, 500)$. All the other instances are the same as in previous experiments. The robot leads h_1 to the first

TABLE II: Experiments Performance Data

Exp.	States	Time [ms]	Virt. Mem. [KiB]	Res. Mem. [KiB]
1 (fail)	648046	153860	166040	124772
1 (scs)	730597	173610	166032	122768
2 (fail)	224826	53620	166140	120844
2 (scs)	4406482	1045290	166136	123944
3	948027	226350	166696	123636

destination, then it fetches the object in the specified location, and returns to the position of h_2, who has never moved away from the starting location. The battery charge is sufficient to execute both tasks and neither of the humans exceeds the maximum fatigue threshold. The property in Eq.8 is verified with $\theta_2 = 0.9$ and $\tau = 360s$.

In conclusion, the tool is able to provide precise predictions about the scenario based on sound mathematical techniques. The reliability of the outcome is highly dependent on the accuracy of the physiological models and on the chosen parameter values. In the future, we plan on improving this aspect of the work by introducing a data-driven model refinement loop. Given the nature of model-checking, verification times increase exponentially with complex scenarios. Nevertheless, we have tested the approach with realistic examples and obtained satisfactory results.

VII. Conclusion

In this paper we have presented a model-driven approach for the analysis of human-robot interaction scenarios in healthcare settings, based on SMC experiments. In the future, we plan to extend the model in Section V to all the patterns described in Section IV-A, to cover a wider range of applications. There are also two future development directions for the overall approach presented in Section IV. Firstly, we envisage the creation of a Domain-Specific Language (DSL) that designers can use to model the mission, with finer-grained details than what is possible at the moment. The DSL could include a more refined model of the human and a richer set of physiological factors. The designer could choose among different human profiles that determine how these variables change over time. On the other hand, we plan on developing the deployment phase of the toolchain. The robot-controller, i.e., the orchestrator, could be transformed into executable code with an automated code generation procedure. This would allow us to test how effectively the robot can interact with real people, and whether they feel like, given the human-oriented nature of the model, their needs are indeed being accomodated.

References

[1] ISO 12100, *Safety of machinery - General principles for design - Risk assessment and risk reduction*. ISO, 2010.

[2] ISO 13849-1, *Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design*. ISO, 2006.

[3] ISO 13482, *Robots and robotic devices - Safety requirements for personal care robots*. ISO, 2014.

[4] T. Jacobs and G. S. Virk, "ISO 13482 - The new safety standard for personal care robots," in *Intl. Symp. on Robotics*. VDE, 2014, pp. 1–6.

[5] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical computer science*, vol. 138, no. 1, pp. 3–34, 1995.

[6] G. Agha and K. Palmskog, "A survey of statistical model checking," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 28, no. 1, pp. 1–39, 2018.

[7] K. G. Larsen, P. Pettersson, and W. Yi, "Uppaal in a nutshell," vol. 1, no. 1-2. Springer-Verlag, 1997, pp. 134–152.

[8] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, "Uppaal SMC tutorial," *STTT*, vol. 17, no. 4, pp. 397–415, 2015.

[9] "HRI Toolchain," https://github.com/LesLivia/hritoolchain, 2020.

[10] R. E. Yagoda and M. D. Coovert, "How to work and play with robots: an approach to modeling human–robot interaction," *Computers in human behavior*, vol. 28, no. 1, pp. 60–68, 2012.

[11] M. Webster, C. Dixon, M. Fisher, M. Salem, J. Saunders, K. L. Koay, and K. Dautenhahn, "Formal verification of an autonomous personal robotic assistant," in *AAAI Spring Symposium Series*, 2014.

[12] F. Vicentini, M. Askarpour, M. G. Rossi, and D. Mandrioli, "Safety assessment of collaborative robotics through automated formal verification," *IEEE Transactions on Robotics*, 2019.

[13] H. B. Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters, "Interaction primitives for human-robot cooperation tasks," in *Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2831–2837.

[14] S. Nikolaidis, R. Ramakrishnan, K. Gu, and J. Shah, "Efficient model learning from joint-action demonstrations for human-robot collaborative tasks," in *Proc. of HRI*. IEEE, 2015, pp. 189–196.

[15] S. Lemaignan, M. Warnier, E. A. Sisbot, A. Clodic, and R. Alami, "Artificial cognition for social human–robot interaction: An implementation," *Artificial Intelligence*, vol. 247, pp. 45–69, 2017.

[16] D. Araiza-Illan, T. Pipe, and K. Eder, "Model-based testing, using belief-desire-intentions agents, of control code for robots in collaborative human-robot interactions," *arXiv preprint arXiv:1603.00656*, 2016.

[17] R. Arai and H. Schlingloff, "Model-based performance prediction by statistical model checking an industrial case study of autonomous transport robots," in *Concurrency, Specification and Programming*, 2017.

[18] M. Foughali, F. Ingrand, and C. Seceleanu, "Statistical model checking of complex robotic systems," in *Proc. of SPIN*. Springer, 2019, pp. 114–134.

[19] B. Herd, S. Miles, P. McBurney, and M. Luck, "Quantitative analysis of multiagent systems through statistical model checking," in *Int. Workshop on Engineering Multi-Agent Systems*. Springer, 2015, pp. 109–130.

[20] M. Askarpour, D. Mandrioli, M. Rossi, and F. Vicentini, "Formal model of human erroneous behavior for safety analysis in collaborative robotics," *Robotics and Computer-Integrated Manufacturing*, vol. 57, pp. 465–476, 2019.

[21] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[22] M. Y. Jaber, Z. Givi, and W. P. Neumann, "Incorporating human fatigue and recovery into the learning–forgetting process," *Applied Mathematical Modelling*, vol. 37, no. 12-13, pp. 7287–7299, 2013.

[23] T. Chettibi, M. Haddad, H. Lehtihet, and W. Khalil, "Suboptimal trajectory generation for industrial robots using trapezoidal velocity profiles," in *IROS*. IEEE, 2006, pp. 729–735.

[24] O. Tremblay, L.-A. Dessaint, and A.-I. Dekkiche, "A generic battery model for the dynamic simulation of hybrid electric vehicles," in *Vehicle Power and Propulsion Conference*. IEEE, 2007, pp. 284–289.

[25] P. Payne, M. Lopetegui, and S. Yu, "A review of clinical workflow studies and methods," in *Cognitive Informatics*. Springer, 2019, pp. 47–61.

[26] K. Kaemarungsi and P. Krishnamurthy, "Modeling of indoor positioning systems based on location fingerprinting," in *Proc. of Infocom*, vol. 2. IEEE, 2004, pp. 1012–1022.

[27] H. Dong, I. Ugaldey, and A. El Saddik, "Development of a fatigue-tracking system for monitoring human body movement," in *Int. Instrumentation and Measurement Tech. Conf.* IEEE, 2014, pp. 786–791.

[28] M. Hadley, "A deterministic model of the free will phenomenon," *Journal of Consciousness Exploration & Research*, vol. 9, no. 1, 2018.