Safety Assessment of Collaborative Robotics Through Automated Formal Verification

Federico Vicentini, Mehrnoosh Askarpour, Matteo G. Rossi, and Dino Mandrioli

***Abstract*—A crucial aspect of physical human–robot collaboration (HRC) is to maintain a safe common workspace for human operator. However, close proximity between human–robot and unpredictability of human behavior raises serious challenges in terms of safety. This article proposes a risk analysis methodology for collaborative robotic applications, which is compatible with well-known standards in the area and relies on formal verification techniques to automate the traditional risk analysis methods. In particular, the methodology relies on temporal logic-based models to describe the different possible ways in which tasks can be carried out, and on fully automated formal verification techniques to explore the corresponding state space to detect and modify the hazardous situations at early stages of system design.**

***Index Terms*—Formal methods, human–robot collaboration (HRC), model-based risk assessment, robot safety, temporal logic.**

I. Introduction

HUMAN–ROBOT collaboration (HRC) in industrial settings enhances the flexibility and adaptability of robotic systems for production. Close proximity and hybrid task assignment between humans and robots have substantial impacts on the safety of human operators. Most importantly, shared dynamic environments cannot be effectively supported by static safety analyses. In particular, the hybrid human–robot tasks generate different possible execution workflows. Therefore, various task assignments among humans and robots can also change during operations. In realistic scenarios, the environment can also change, due to mobile resources, tool changing, modification of the layout, and process locations, so that the specifications of the system need to be re-evaluated. The safety of such evolving systems should not be verified as a static property, but rather according to the behavior of the system in actual situations.

Manuscript received May 10, 2018; accepted August 19, 2019. Date of publication September 18, 2019; date of current version February 4, 2020. This article was recommended for publication by Editor A. Kheddar upon evaluation of the reviewers' comments. This work was supported in part by the European Community's Seventh Framework Programme under Grant 608849 EuRoC (European Robotics Challenges), and in part by the Fondazione Cariplo and Regione Lombardia through the AUTOVAM Project under Grant 2017-1213. *(Corresponding author: Mehrnoosh Askarpour.)*

F. Vicentini is with the Institute of Industrial Technologies and Automation (ITIA), National Research Council of Italy, 20133 Milan, Italy (e-mail: federico.vicentini@stiima.cnr.it).

M. Askarpour and M. G. Rossi are with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, 20133 Milano, Italy (e-mail: mehrnoosh.askarpour@polimi.it; matteo.rossi@polimi.it).

D. Mandrioli is with the Politecnico di Milano, 20133 Milan, Italy (e-mail: dino.mandrioli@polimi.it).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2019.2937471

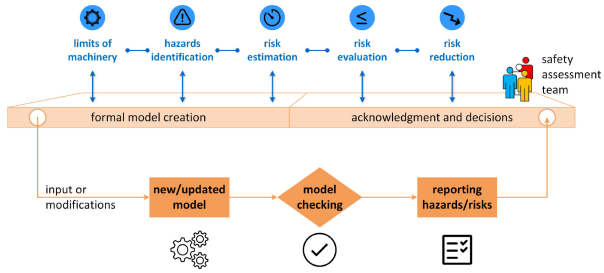


Fig. 1. Overview of the SAFER-HRC methodology (bottom part) with respect to risk assessment steps, simplified from ISO 12100 (top part). SAFER-HRC creates a formal model of the HRC application, on which a risk assessment is carried out through automated formal verification. The results are reported to a team of human assessors, who acknowledge risk evaluations and refine the design of risk reduction models, then rerun SAFER-HRC until a desired result is achieved.

A HRC safety analysis shall comply with the requirements of ISO 12100 [1], as shown in the flowchart at the top of Fig. 1. First, the procedure investigates any source of potential harm or discomfort to humans—known as *hazards*—during the course of a task execution. Then, identified hazards are evaluated in terms of *risk*—a metric, combining the intensity and likelihood of hazards, for the potential hazard to become an actual harm—and mitigated by introducing suitable *risk reduction measures (RRM)*. A hazard is just a potential danger, and the goal of safety is achieved when *high risks* are resolved using different strategies [2].

For changing scenarios, safety analysis needs to be comprehensive and exhaustive, and it should facilitate the exploration of different variants and configurations of the system (e.g., to support design updates). Manual risk assessments applied to HRC cases are necessarily static and pose significant challenges to experts in identifying all possible hazardous situations due to the high level of unpredictability of typical human behaviors. As applications become more and more complex and human–robot interactions expand, some risks could be overlooked because they are hidden in the evolution of the system. The main resulting limitation is the likely selection of ineffective or overconservative strategies (e.g., *always* keep a low speed).

This article introduces a rigorous, iterative methodology—based on formal methods—to guarantee the safety of physical HRC systems. It is structured as follows. Section II discusses the state of the art in the field of safety analysis of HRC applications, and Section III gives an overview of our own contributions to it. Section IV introduces the background in formal verification

relevant for safety assessment, highlighting models, and safety properties. HRC system modeling is described in Section V, which introduces the formalization of all relevant elements in terms of temporal logic, including the modeling of actions, which is used to identify safety-critical situations. Section VI describes how the formal approach supports the key modules for safety analysis: extraction of hazards from the formal model, risk estimation, and the verification of risk reduction upon the introduction of models for mitigation measures. Section VII reports on a use case of collaborative task where the method is applied and illustrated. Finally, Section VIII discusses several aspects of the methodology, the obtained results, and some open issues.

II. State of the Art

There is a rich literature about safety analysis of systems where humans and machines interact in a critical way. We can coarsely categorize it as follows:

a) general, mostly informal, and safety analysis techniques;
b) techniques and standards that more specifically address human interaction with systems;
c) methods, techniques, and tools that exploit—at some level—the mathematical formalisms.

Unsurprisingly, some works found in literature exhibit features that belong to several of the abovementioned categories.

Among traditional, general-purpose, safety-centered techniques, FMEA/FMECA [3], failure analysis technique (FTA) [4], HAZard and OPerability study (HAZOP) [5], and systems-theoretic accident model and processes (STAMP) [6] are very well known. FMEA/FMECA are bottom-up analytical methods that determine and evaluate the effects of known failures, then introduce actions to eliminate or reduce the chance of such potential failure effects, documenting the whole process. In FMECA, remedial actions are applied proportionally to the criticality of failures. FTA is a deductive failure analysis technique, which exploits Boolean algebra and probability theory. It finds out the causes of failure behavior via diagrams and logic models. HAZOP identifies hazards and operability problems and investigates how a system can deviate from the intended design. HAZOP should be carried out early in the design stage to allow for effective design adjustments with minimum additional costs. A multidisciplinary team (HAZOP Team) uses brainstorming meetings to determine if the design intents are satisfied, using guide-words that provide qualitative measures of design deviations.

Unlike other methods, STAMP views accidents as control problems, which involve complex dynamic processes (e.g., component failure accidents, unsafe interactions among components, design errors) and not simply as a chain of failures. STAMP-based process analysis (STPA, [7]) is a powerful hazard analysis technique based on STAMP, which is used in different industry sectors.

All of these methods are general and domain-agnostic, and thus, applicable also to human–robot interaction and collaboration. For example, authors in [8] have applied HAZOP to risk analysis of an assistive service robot.

Works specifically addressing human–machine interaction [including category (b) introduced above] have always dedicated particular attention to the active presence and decision making of human operators, or various temporal combinations of human and machines (or robots).

For example [9] deals with unpredictable environments and the various temporal combinations of their evolution, capturing hazards due to human factors; the authors in [10] focuses on the combination of different hazards, avoiding to create an excessive amount of redundant information; the authors in [11] argues for producing results that are not generic, (semi)-automated, and reusable for different scenarios. Approaches belonging to these categories are complemented with dedicated expertise for modeling and analyzing hazards associated with physical human–robot interaction, e.g., accidental or purposeful contacts.

Formal methods [12]—those of type (c) in the classification mentioned above—are a set of well-defined mathematical languages and techniques for accurate modeling, specification, and automated verification of systems. The formal model of an application can capture all of its associated workflows. Additionally, verification can be done via automated techniques that are capable of exploring all such workflows, thus, eliminating any chance of overlooking any of them.

In robotics, formal modeling and verification techniques have been mostly used for defining robot behavior (e.g., controllers, motion planning, fault detection). Formal verification has been also exploited for exhaustive and comprehensive safety analysis. A number of works developed methods that employ (semi-) formal[1] approaches to address safety. For example, Guiochet [13] merges the semiformal UML notation [14] with HAZOP first to describe test-case scenarios through sequence and state machine diagrams, and then to identify hazards and analyze their risks. The authors in [15] and [16] model an assistant robotic system through the Brahms language [17] and verify it against safety requirements via the SPIN model checker [18]. The authors in [19] and [20] also employ formal verification to analyze the safety of assistant robots. The authors in [21]–[23], instead, focus on nonrobotic collaborative applications and use simplified cognitive architectures that require large manual customizations, but mostly ignore human fallibility and plausible errors. These examples of formal modeling have their major focus on the plausibility and correctness of robot (or device) programmed behavior, whose failure may have negative consequences on trustworthiness and—indirectly—safety due to the creation of potentially harmful situations. No specific modeling is provided for mechanical hazards (e.g., crushing, trapping, entanglement, abrasion, and shearing) derived from the physical interaction between humans and robots in industrial applications, as normatively reported in domain-specific international standards such as ISO 10218-2 [24] and ISO/TS 15066 [25], or for situations and hazards reported for personal care robots in ISO 13482 [26].

[1]The term "semiformal" denotes a notation whose syntax—i.e., the rules for text or diagram composition—is mathematically defined, but whose semantics is left to the experience and intuition of writers and readers.

To this end, task-centric approaches usually rely on decompositions of the tasks to be executed, from which hazards derive. For instance, hierarchical task analysis (HTA [27]) offers an exhaustive description of tasks by decomposing them in a hierarchy of goals, subgoals, operations, and plans. Many task-centric approaches, e.g., [28], rely on HTA to analyze the safety hazards that the task itself might create, not only those due to system failures [as in category (a) mentioned above]. For capturing the nature of hazards in human–robot physical interaction, nonetheless, models should describe not only the task to be executed with its goals and workflow, but also, as pointed out by the authors in [29], the various components of the system: physical and functional elements of the work domain (e.g., objects, geometries), human–machine interaction, as in [30], human judgments in different situations, the configuration of the control system, etc. This entails the necessity to create a well-integrated model of the overall system model, where human behavior plays a key role with distinguishing features. The authors in [31] identify the following four categories of techniques for the modeling of the human behavior (many of which based on formal methods).

1) *Human-device interface (HDI)* models, extensively studied in [32], focus on the interaction via interface tools and do not deal with the *copresence* of human and device [33].
2) *Cognitive* models describe the knowledge behind human observable behavior [32] and encompass a wide range of models, from classic (e.g., SOAR [34] and ACT-R [35]) to more recent ones (e.g., OCM [36] and PUM [37]). PUM is the best-suited for formal verification because of its general and accurate semantics. Examples of PUM-based techniques are [38] and [39].
3) *Human mental* models distinguish the machine model from the user's perception and cognition. The authors in[21], [40]–[42] introduce several formalisms for building such models. These models are too detailed to suit an exhaustive state-space and risk exploration because of state-explosion issues.
4) *Task-analytic* models are based on hierarchical structures of tasks [31]; examples are OFM [43], UAN [44], ConcurTaskTrees [45], and EOFM [46].

The authors in [47] and [48] combine cognitive models and probability distributions. However, the authors of these works mention that, since real data on human behavior is not available, they had to rely on self-generated fictional data that does not necessarily reflect or abstract the actual behavior of operators. This significantly lessens the reliability of these approaches for what concerns safety analysis.

Unlike interface models, cognitive and task-analytic ones are able to represent the collaboration between human operators and robots. However, they have been mostly used in contexts that are not fully formal, or focus on a limited set of *intended behaviors* (i.e., regular use of the system, standard and intended instructions of the application), ignoring instead the *unintended* ones (i.e., any deviation from the intended use), which are a major source of hard-to-predict hazards [49]. HRC applications require a more comprehensive effort in the modeling of the human behavior than what is currently used in the domains of HDI, with respect to hazards deriving from interaction with machinery. In HRC for both industrial and service purposes, robots are embodied and powerful.

In summary, the state of the art of safety analysis in HRC offers a rich range of approaches and methods, each one with its own merits and specificities. In the following section, we introduce our own contribution in the framework of formal verification *and* direct safety modeling (e.g., physical hazards to humans).

III. Contributions

This article proposes a methodology called Safety Assessment through Formal Verification in HRC (SAFER-HRC) for supporting dynamic safety assessment in HRC applications. SAFER-HRC uses model-based formal verification to achieve the following:

1) exhaustively explore all possible execution workflows of an application;
2) automatically identify hazards within those workflows and determine their associated risk, in compliance with ISO 12100 and ISO 10218-2;
3) introduce provisions for risky hazards and iteratively verify the effectiveness of such protective measures by computing residual risks.

We formally model the system through a temporal logic language. We opted for a logic-based approach over other formal methods—such as, for example, state-based formalisms—because of its descriptive nature, which makes it more suitable to abstract away from irrelevant details; furthermore, many variants of temporal logic are decidable and support automated formal verification techniques. In fact, our model is formally verified by a model checker against a desired safety property, i.e., the *negligibility of risks* (a hazard is harmless if its associated risk score is below a certain threshold).

In our approach, the informal and goal-oriented description of an application is translated into a logic model that captures the key components of a HRC system (operators, robots, and the layout), complemented with formalized descriptions of hazardous situations. Additional modules formalize, in the same logic, risk estimation procedures (as in ISO/TR 14121-2 [50]) and protective measures. These modules include predicates and formulae about humans, robots, environment, hardware properties, spatial and functional relationships among elements, and safety protection factors (e.g., power limits). The whole logical model is *discretized* to abstract away from excessive details and to avoid state-space explosion issues [51].

The unpredictability of human behavior during the execution of a HRC task could generate a range of unintended, potentially hazardous, situations. Thus, a sufficiently accurate model of the system requires also to consider human factors and the observable manifestation of the operator's physical presence and activities. SAFER-HRC captures the effects of human behavior on safety properties to be verified, rather than cognitive-based mechanisms behind such behaviors. A functional model—rather than a cognitive one—replicates human errors independently from attitudes of operators (e.g., being tired or absent-minded) leading to such errors.

The selection of applicable risk mitigation strategies is not fully automated. As shown in Fig. 1, our methodology does not replace the human risk assessor, but provides an automated assistant that formalizes the standard risk evaluation procedure of ISO 12100 applied to collaborative robotics. The human risk assessor is in charge of, first, tailoring the model for the system, and, second, confirming the chosen mitigation action (e.g., limit power, reduce speed, change path) after running the verification process again.

IV. Background: Temporal Logic and Automated Formal Verification

The ultimate goal of the presented methodology is to comprehensively explore and verify all possible interactions between robots and human operators, with respect to *risks*. Recall that risks, not hazards, are the key notion of safety. The goal is then achieved by combining two elements: first, a formal model that captures the sequences of actions performed by robots and operators in their fulfillment of assigned tasks; second, mechanisms to exhaustively analyze these sequences of actions, and to detect those that are deemed unacceptable if some hazards, when present, display an intolerable risk level.

A. Modeling Actions Along Time

The first requirement (modeling actions) is accomplished using a temporal logic language, called TRIO [52], which is capable of expressing the evolution over time of phenomena of interest (e.g., "the operator enters a robotic cell *before* a robot moves") and is general enough to describe decidable and undecidable models, over discrete and continuous time. TRIO is amenable to automated formal verification performed by Zot [53], which is a bounded satisfiability checker that exhibits comparable performance with respect to well-known bounded model checkers such as NuSMV [54]. Other logic languages, such as MTL [55], which can also be automatically analyzed through the Zot tool [56], would have suited our needs, though TRIO was developed specifically with the idea of allowing practitioners to build formal specifications of critical systems [57].

Additionally, the TRIO language features a quantitative notion of time, so that it is possible to render properties of tasks such as "the robot *always* reaches the working pose 3 time units *after* leaving its homing position."

We rely on a fully logic-based approach to model actions because this allows us great flexibility and modularity in building models. For example, adding or removing actions to/from workflows simply involves adding/removing the corresponding logic formulae, and similarly for other elements of the system, such as RRMs.

TRIO formulae are built out of propositions and predicates describing the basic phenomena of the system (e.g., $\mathrm{turn\_on}, \mathrm{off}$) and the usual propositional connectives—"and" ($\wedge$), "not" ($\neg$), etc.—and first-order quantifiers—"forall" ($\forall$), "exists" ($\exists$)—as well as a single basic modal operator, called Dist, that relates the *current time* to another time instant. Then, if ϕ is a TRIO formula and d is a time distance, $\mathsf{Dist}(\phi, d)$ holds at time t if, and only if, ϕ holds at time $t + d$. While TRIO can exploit both discrete and dense sets as time domains (e.g., the set of nonnegative integers $\mathbb{N}$, or the set of nonnegative real numbers $\mathbb{R}_{\geq 0}$), in this article, we assume $\mathbb{N}$ as discrete time domain. For convenience in the writing of specification formulae, TRIO defines a number of *derived* temporal operators from the basic Dist, through propositional composition and first-order logic quantification. Table I reports some of the most significant ones, including those used in this article. For example, consider the following two generic predicates: $\mathsf{leave\_home}$, to describe when the robot departs from a homing position—i.e., $\mathsf{leave\_home}$ is true at time t if the robot departs at that time—and $\mathsf{at\_target}$, to capture the fact that the robot is operational in a position. Then, the situation informally introduced above is formally captured by the TRIO formula $\mathsf{Alw}(\mathsf{leave\_home} \Rightarrow \mathsf{Dist}(\mathsf{at\_target}, 3))$. Interested readers can find more details about the TRIO language in [52]. A formula ϕ describing some situation in the TRIO language is evaluated over *histories*, which formally capture the notion of "behavior" of a system.

TABLE I
List of Derived TRIO Operators; ϕ, ψ Denote Propositions, and d Is a Constant Value

TRIO Operator	Definition	Meaning
$\mathsf{Past}(\phi, d)$	$d > 0 \wedge \mathsf{Dist}(\phi, -d)$	ϕ occurred d time units in the past
$\mathsf{Futr}(\phi, d)$	$d > 0 \wedge \mathsf{Dist}(\phi, d)$	ϕ occurs d time units in the future
$\mathsf{Alw}(\phi)$	$\forall t(\mathsf{Dist}(\phi, t))$	ϕ always holds
$\mathsf{Som}(\phi)$	$\exists t(\mathsf{Dist}(\phi, t))$	ϕ occurs sometimes
$\mathsf{Until}(\phi, \psi)$	$\exists t(\mathsf{Futr}(\psi, t) \wedge (\forall t'(0 < t' < t) \Rightarrow \mathsf{Dist}(\phi, t')))$	ψ will eventually occur and ϕ will hold till then
$\mathsf{Until}_{\mathsf{w}}(\phi, \psi)$	$\mathsf{Until}(\phi, \psi) \vee \mathsf{Alw}(\phi)$	weak until: ψ may never occur in the future
$\mathsf{WithinF}(\phi, d)$	$\exists t(0 < t < d \wedge \mathsf{Dist}(\phi, t))$	ϕ will occur within d time units
$\mathsf{Lasted}(\phi, d)$	$\forall t(0 < t < d \rightarrow \mathsf{Dist}(\phi, -t))$	ϕ held for the last d time units

Definition 1: Given a predicate pred, a *history* H for pred is a set of its values along the temporal domain $\mathbb{N}$; we indicate the values taken by pred in history H as ${}^{H}\overline{\mathsf{pred}}(0), {}^{H}\overline{\mathsf{pred}}(1), {}^{H}\overline{\mathsf{pred}}(2), \ldots$. This definition is extended in a natural way to sets of predicates $\mathcal{S}$ and to formulae ϕ.

For example, consider the predicate $\mathsf{leave\_home}$ introduced above: a history H_1 might be such that $\mathsf{leave\_home}$ is true at time 8 (denoted as ${}^{H_1}\overline{\mathsf{leave\_home}}(8)$). Another possible history H_2 might be such that $\mathsf{leave\_home}$ is true at time 10, but not at time 8.

Definition 2 (Semantics of Dist): Let ϕ be a TRIO formula, $\mathbb{N}$ the temporal domain and $t \in \mathbb{N}$ the current time instant. Then, ${}^{H}\overline{\phi}(t)$ indicates the value of formula ϕ at instant t for history H, which is either *true* or *false*. Let d be a (positive or negative) time distance—i.e., $d \in \mathbb{Z}$. Formula $\mathsf{Dist}(\phi, d)$ holds at time t for

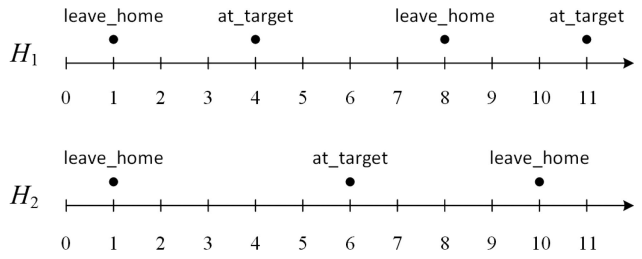


Fig. 2. Two examples of histories for formula Alw(leave_home⇒ Dist(at_target,3)), one that satisfies the formula (history H_1) and one that does not satisfy it (H_2).

history H if, and only if, ϕ holds at time $t + d$, or, equivalently, ${}^{H}\overline{\mathsf{Dist}(\phi, d)}(t)$ is true if, and only if, ${}^{H}\overline{\phi}(t + d)$ is true.

For example, consider history H_1 (see Fig. 2) such that both ${}^{H_1}\overline{\mathsf{leave\_home}}(8) = \text{true}$ and ${}^{H_1}\overline{\mathsf{at\_target}}(11) = \text{true}$ hold; then, formula $\phi_{ex} := \mathsf{leave\_home} \Rightarrow \mathsf{Dist}(\mathsf{at\_target}, 3)$ holds at time 8 in H_1, that is, ${}^{H_1}\overline{\phi_{ex}}(8) = \text{true}$.

B. Verification of Properties

A history H is said to *satisfy* a TRIO formula ϕ if ϕ holds for H in the origin of the time domain—i.e., if ${}^{H}\overline{\phi}(0)$ is true. In general, a formula ϕ is *satisfiable* if there exists a history H for which it is satisfiable; if no such history exists, ϕ is *unsatisfiable*. For example, consider formula $\mathsf{Alw}(\phi_{ex})$, where ϕ_{ex} was introduced above, and the histories H_1 and H_2 of Fig. 2. History H_1 satisfies $\mathsf{Alw}(\phi_{ex})$, because every time leave_home is true in the history (i.e., at instants 1 and 8), after 3 time units at_target is also true; hence, ${}^{H_1}\overline{\mathsf{Alw}(\phi_{ex})}(0) = \text{true}$. History H_2, on the other hand, does not satisfy $\mathsf{Alw}(\phi_{ex})$, because leave_home is true at time 1, but there is no corresponding at_target at time 4; hence, ϕ_{ex} is not true in 1 and ${}^{H_2}\overline{\mathsf{Alw}(\phi_{ex})}(0) = \text{false}$.

Satisfiability checking is the problem of determining whether a TRIO formula ϕ is satisfiable or not. In general, checking the satisfiability of TRIO formulae is an undecidable problem. However, this article uses a decidable subset of the language, that can be handled by automated tools, in order to build the system model and to express its properties. In particular, the Zot bounded satisfiability checker [53] implements several techniques for checking the satisfiability of TRIO formulae in an automated manner [56], [58]. More precisely, given an input TRIO formula ϕ, Zot returns either a history satisfying ϕ or the value UNSAT, which stands for "unsatisfiable." Satisfiability checking is the underlying mechanism for verifying the safety properties of an application. An application corresponds to a set of histories. Let ϕ_S be a TRIO formula that formalizes the possible behaviors of a system. Then, the histories that satisfy ϕ_S are exactly those that capture all possible sequences of phenomena of interest occurring in the application. Let ϕ_P be a TRIO formula that captures a safety requirement—e.g., "the risk value is always less than a given tolerable threshold" Then, the desired property to be verified holds for the application if there are no histories of the system described by ϕ_S that violate the property described by ϕ_P. Formally, there are no behaviors (histories) of the system for which "not ϕ_P" holds.

Definition 3 (Safety verification): Given a formula ϕ_S capturing the behaviors of the system and a formula ϕ_P formalizing the safety property, the system is safe if, and only if, $\phi_S \wedge \neg\phi_P$ is unsatisfiable.

Operationally, an application is verified to be safe or not by inputting a suitable TRIO formula (i.e., $\phi_S \wedge \neg\phi_P$) in the Zot bounded satisfiability checker. The outcome UNSAT proves the application being safe. Otherwise, Zot returns a history that not only satisfies ϕ_S, but also violates ϕ_P (because it satisfies $\neg\phi_P$); this is a *counterexample* witnessing why the application is not safe.

Definition 4: Under the generic notion of risk, the specific *safety property* ϕ_P to be used in Definition 3 becomes

$$\mathsf{Alw}(\mathsf{risk} \leqslant \tau) \tag{1}$$

where τ is a tolerable value of a quantitative notion of risk (to be detailed in Section VI-B).

The description of an application is made of a set—or a conjunction—of formulae ϕ_S, to which Definition 3 collectively applies. An instance of the formal model captures all possible behaviors of the system in a given application—i.e., the histories of values taken by temporal logic predicates. Different histories from the same model instance are possible and are due to different timings for the modeled actions (e.g., an operator might reach a target in a nondeterministic way). Whenever an application is changed, a new instance of the TRIO model is produced, in turn corresponding to many histories. Changes might be due to variations in the nominal workflow when the functional purpose of the application is modified, or to variations in the properties of elements (e.g., safety limitations are enabled in robots), a rearrangement of the layout, the introduction of new devices, etc. Upon reinstantiation of the model—regardless of the reason—the safety condition captured by formula (1) is checked again for satisfiability. This automatic mechanism is very useful whenever the conditions of the application change because a planning action is done (e.g., task change) or some safety functions (e.g., power limitation) are manually enabled by the safety person(s) in charge of the tool.

V. Methodology of Formalization of HRC Applications

This section provides the modeling elements of human–robot collaborative tasks—i.e., the components of the system (see Section V-A)—and how such tasks are organized in actions (see Section V-B). The model described in this section and in Section VI constitutes the temporal logic formulae denoted by ϕ_S in Definition 3.

A. Elements of HRC Systems: Operators, Robots, and Layouts

The formal model of a HRC system involves three main classes: operators ($\mathcal{O}$), robots ($\mathcal{R}$), and layouts ($\mathcal{L}$).

Definition 5 (Layout ($\mathcal{L}$)): The workspace *layout* is partitioned in locations $\mathcal{L} = \{L_1, \ldots, L_n\}$, each of which is specified by a tuple of characteristics that help identify hazards and

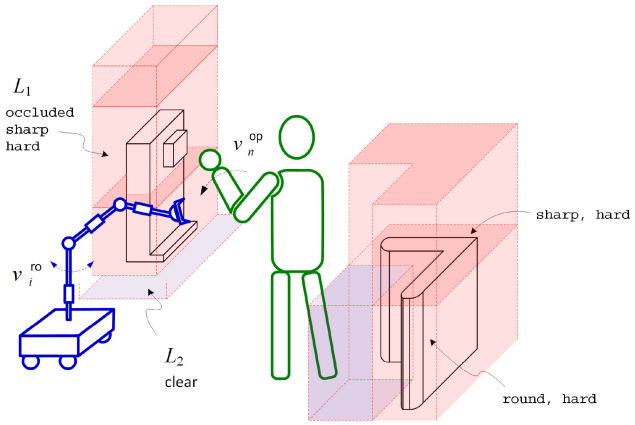


Fig. 3. Example HRC layout modeled with a manipulator, an operator, and some fixtures. Some locations L (shaded regions) are represented together with the values of their attributes $obst$, $shape$, and $material$.

estimate their risks as

$$\begin{aligned} L_k &= \langle shape_k, material_k, obst_k \rangle \\ shape_k &\in \{\texttt{sharp}, \texttt{chamfer}, \texttt{fillet}, \texttt{round}\} \\ material_k &\in \{\texttt{soft}, \texttt{hard}\} \\ obst_k &\in \{\texttt{occluded}, \texttt{clear}, \texttt{free}, \texttt{warning}\} \end{aligned} \tag{2}$$

$shape_k$ and $material_k$ are abstract representations of the form and substance of objects present in each location and take one of the mentioned discrete values. The attributes of a location L are intended to render some safety-related properties of the environment, in order to underpin the identification of potential hazards, and some information useful for the risk estimation of such hazards (see Section VI-A for details).

In particular, attribute $obst$ is closely related to the workspace of the robot element. Depending on where the robot is based, for each location L that the robot cannot reach (see Fig. 3), it holds that $obst = \texttt{free}$. Assignments $obst = \{\texttt{occluded}, \texttt{clear}\}$ are, instead, defined for all locations within the reach of the robot, depending on the presence or absence of constraining objects. Assignment $obst = \texttt{warning}$ is defined for regions outside the robot workspace, which are relevant for safety measures (e.g., slow down the robot).

For example, the layout in Fig. 3 reports three regions around the left (hazardous) fixture, of which location L_1 is labeled as $L_1 = \langle \texttt{sharp}, \texttt{hard}, \texttt{occluded} \rangle$, while L_2 location is further away from the fixture and is labeled as $L_2 = \langle \texttt{sharp}, \texttt{hard}, \texttt{clear} \rangle$.

Remark 1: The three-dimensional (3-D) shape of locations $\mathcal{L}$ is application-oriented and is defined at modeling time. The workspace is partitioned starting from a tiling of the layout in planar geometries according to the presence of objects, e.g., contouring a fixture. Then, planar tiles are projected along their normal direction for a given height (e.g., up to human hip, then up to human shoulders, where actual measurements and percentiles are taken from [59]). $\mathcal{L}$ results in a set of stacked volumes (see Fig. 3) sharing surface boundaries, i.e., preventing gaps or overlaps between regions.

Definition 6 (Robot ($\mathcal{R}$)): The *robot* model $\mathcal{R}$ is composed of the formalization of its n elements (e.g., links)—that is, $\mathcal{R} = \{R_1, \ldots, R_n\}$—and several kinematic constraints that capture the correct movements of manipulators. Each individual element R_j is specified by a tuple consisting of both invariant—i.e., constant over time—attributes (mass m_j and shape $shape_j$) and variable—whose values change over time—attributes (current values of position on the layout p_j, velocity v_j, and force f_j):

$$\begin{aligned} R_j &= \langle p_j, v_j, f_j, m_j, shape_j \rangle \\ p_j &\in \mathcal{L}, v_j, f_j \in \{\texttt{none}, \texttt{low}, \texttt{mid}, \texttt{high}\} \\ m_j &= \{\texttt{low}, \texttt{mid}, \texttt{high}\} \\ shape_j &\in \{\texttt{sharp}, \texttt{chamfer}, \texttt{fillet}, \texttt{round}\}. \end{aligned} \tag{3}$$

The position and velocity of each element are expressed with reference to a specific *point of interest* (POI) of the element (e.g., the position and velocity of the end-effector are defined with respect to its center of mass). As shown above, the values of velocity when translating from one location to another, and of the force exchanged between a robot element and the environment are expressed through quantized discrete values. For example, assuming that R_1 and R_2 are two consecutive parts, $\mathsf{Alw}(p_1 = p_2 \vee \mathsf{Adj}(p_1, p_2))$ states that they—indeed, the corresponding POIs—are always in the same location, or in adjacent ones. Similarly, there are constraints on velocity and force values to exclude discontinuities. For example, $\mathsf{Alw}(v_j = \texttt{none} \Rightarrow \mathsf{Futr}(v_j = (\texttt{none}|\texttt{low}), 1))$ asserts that if the velocity is equal to `none`, it cannot jump to `mid` or `high` at the next time instant and it can only be equal to `low` or `none`. Another example is $\neg \mathsf{moving}_{\mathcal{R}} \Leftrightarrow \forall j (v_j = \texttt{none})$, which states that a robot is not moving when all its elements have null velocity. The representation of position and velocity of robot $\mathcal{R}$ elements through corresponding POIs in the operational space allows us to overcome joint space representation issues due to singularities or null-space movements of redundant axes.

Remark 2: One of the criteria to choose an appropriate granularity level is to assume that the position p_j of each robot element is equal to a single location of the layout—i.e., the location sizes depend also on the expected velocity of R_j in that region. Definition 6 uses a single POI to represent each robot element, thus, adopts this assumption, and excludes the occupation by a robot element of multiple locations at the same time. We claim this is the case for most common applications, including example 1 in Section VI-A and the case study in Section VII, as we have divided the layout into functional volumes that can be measured and are related to the programmed movements of the robot system. In fact, volumes close to fine manipulation positions should be smaller than those where large displacements are expected/planned. However, this assumption can sometimes be relaxed into a multilocation occupation model (for example, by considering multiple POIs for each robot element), especially for large human body parts that can extend over many adjacent small regions L, which entails a greater accuracy in the modeling of the positioning of agents, but also a more complex set of spatial interactions (e.g., concurrent occupancy).

Definition 7 (Operator ($\mathcal{O}$)): The model of a human *operator* is composed of a set of formalized definitions of 11 body regions $\mathcal{O} = \{O_{\text{head}}, \ldots, O_{\text{leg}}\}$ [25] and a set of constraints introduced for the sake of the consistency of the body model. Every body part O_i is defined by a tuple including both constant values, which are relevant for the mechanics of contacts (mass m_i and stiffness k_i) and the dynamic values of position and velocity (which, as in the robot model $\mathcal{R}$, refer to a POI of the body region) as

$$\begin{aligned} &O_i = \langle p_i, v_i, m_i, k_i \rangle \\ &p_i \in \mathcal{L}, v_i \in \{\texttt{none}, \texttt{low}, \texttt{mid}, \texttt{high}\}. \end{aligned} \tag{4}$$

For example, $\mathsf{Alw}(p_{\text{head}} = p_{\text{arm}} \vee \mathsf{Adj}(p_{\text{head}}, p_{\text{arm}}))$ is a constraint that asserts that consecutive body parts are always in locations that are the same or adjacent, thus, ruling out unrealistic configurations such as head and arms being in opposite corners of the layout.

For the sake of compatibility with relevant standards in the field, we have chosen to adopt the human body analysis of ISO/TS 15066, which identifies 11 body regions-29 specific localizations—according to their different pain tolerance on the basis of biomechanical studies in [60] on impacts (by robots). The 11 body regions are head, face, neck, shoulders, chest, belly, pelvis, upper and lower arms, hands, thigh, and legs.

B. Formalization of Tasks

In a given history H, the *state* s of the system at a given time t is the set of values taken in H by the variables, parameters, and properties defined in the model: we indicate it by ${}^{H}\overline{s}(t)$. For example, $\neg\mathsf{moving}_{R_1} \wedge \mathsf{moving}_{R_2} \wedge (p_{R_1} = L_1) \wedge p_{\text{head}} = L_2$ is a constraint that predicates on the state of the model at a given time; for the constraint to hold at time t in a history H, then it must be that $v_1 = \texttt{none}$ holds at t, i.e., ${}^{H}\overline{v_1}(t) = \texttt{none}, v_2 \neq \texttt{none}$, etc. In this article, we are interested in studying the behavior of HRC applications over *bounded histories*. More precisely, a bounded history of length K is such that only time instants $t \leq K$ are taken into account, with K sufficient to reach the conclusion of each history. The evolution of the model from one state to another (i.e., its dynamics) derives from *actions*, which compel attributes to change values. For example, an operator is required to move from one location (say, L_1) to another (L_2), as part of a task, while the robot is not working. The execution of this action, taking x time units, modifies the state from ${}^{H}\overline{s}(t)$, which is compatible with constraint $p_i = L_1 \wedge \neg\mathsf{moving}_{\mathcal{R}}$ to state ${}^{H}\overline{s}(t+x)$, which is instead compatible with constraint $p_i = L_2 \wedge \neg\mathsf{moving}_{\mathcal{R}}$. There might be intermediate states between instants t and $t+x$—e.g., the operator passes through other possible locations L_i and L_j to reach L_2.

Definition 8: Every task is composed of a set of atomic *actions*. An action a_i is defined by a tuple of the following form, where a_i^{exeT} is its required time for termination, a_i^{agent} is its expected executor agent (operator or robot), $preC_i$ and $posC_i$ are two sets of formulae that enable/control its actual execution, and a_i^{sts} defines its executional state, which will be explained later on

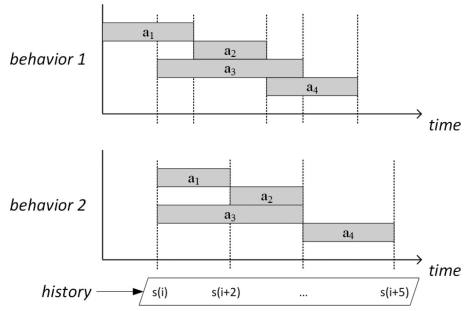


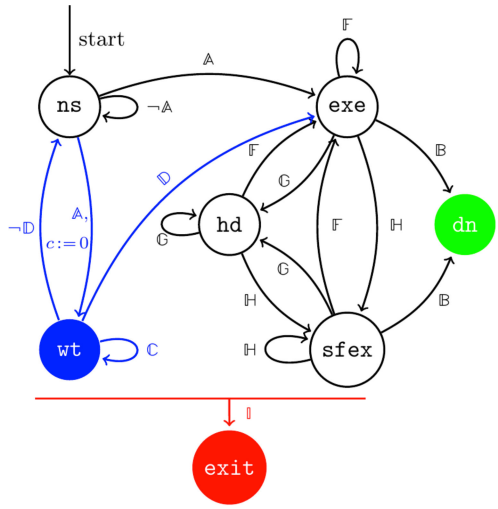
Fig. 4. Sequences of actions can differ according to evaluation of their preconditions.

$$\begin{aligned} a_i &= \langle a_i^{exeT}, a_i^{\text{agent}}, preC_i, posC_i, a_i^{sts} \rangle \\ &a_i^{exeT} \in \mathbb{N}, \qquad a_i^{\text{agent}} \in \{\texttt{op}, \texttt{ro}\} \\ &a_i^{\text{agent}} = \texttt{op} \rightarrow a_i^{sts} \in \{\texttt{ns}, \texttt{wt}, \texttt{exe}, \texttt{sfex}, \texttt{hd}, \texttt{dn}, \texttt{exit}\} \\ &a_i^{\text{agent}} = \texttt{ro} \rightarrow a_i^{sts} \in \{\texttt{ns}, \texttt{exe}, \texttt{sfex}, \texttt{hd}, \texttt{dn}, \texttt{exit}\}. \end{aligned} \tag{5}$$

Fig. 4 shows the workflow of actions. For example, actions a_2 and a_4 could start only after termination of a_1 and a_2, respectively; hence, $a_1^{sts} = \texttt{dn} \in preC_2$ and $a_2^{sts} = \texttt{dn} \in preC_4$. However, preconditions do not necessarily specify a workflow as a whole: other actions may execute between a_2 and a_4, and a_2 may start on different time instants in two different histories of the model.

1) Model of Robot Actions: The semantics of our proposed model allows actions to have a set of executional states and defines the transitions among them, as depicted in Fig. 5. However, this semantics does not capture the algorithm implemented by the robot controller, nor the operational state of the controller (e.g., idle, busy); rather, it is the abstraction of the conditions of robot actions. Transitions between different states of the action can also be triggered by cross-action predicates that qualify the presence of hazards, risks, and RRMs. A robot action can be in one of the following states.

1) `ns` (not started): initial default state that holds until all preconditions become true.
2) `exit`: given a safety property that the system should satisfy, such as the one in Definition 4, the task execution aborts upon its violation, which corresponds to the detection of hazardous situations with high risk during the lifetime of any of the actions.
3) `exe` (executing): running state which is triggered when all preconditions are satisfied.
4) `sfex` (safe executing): extension of `exe` state with at least one active RRM in order to keep the risk level acceptable.
5) `hd` (hold): exception state, entered upon an explicit suspension of execution due to a request from the operator. The state is used when the execution is momentarily paused, although some safety RRMs may be enabled.
6) `dn` (done): regular termination state which is triggered when all postconditions are satisfied.



Label	Full Formula	Definition
𝔸	$preC$	$preC$ holds.
𝔹	$posC$	$posC$ holds.
ℂ	$\mathbb{A} \wedge \neg$opStarts	an operator action does not start before opStarts.
𝔻	$\mathbb{A} \wedge$ opStarts $\wedge\, c \leqslant \Delta$	operator should start within Δ time units.
𝔼	$\exists i,j,k,x(\mathsf{RRM}^{x}_{ijk})$	a RRM of type x, involving O_i, R_j and L_k, is active.
𝔽	$\neg\mathbb{E} \wedge \neg posC$	$posC$ are not fulfilled yet and no RRM is active.
𝔾	$\exists i,j,k(\mathsf{RRM}^{hold}_{ijk})$	a RRM of type hold is active.
ℍ	$\mathbb{E} \wedge \neg\mathbb{G} \wedge \neg\mathbb{B}$	$posC$ are not fulfilled, yet, and some RRM is active, which is not of type hold.
𝕀	$\exists i,j,k(\mathsf{risk}_{ijk} > \tau)$	risk exceeds the threshold.

Fig. 5. Semantics of human and robot actions. For readability, edges (labeled 𝔸–𝕀) and an auxiliary formula 𝔼 are explained in the table mentioned above. For operator actions the blue part replaces the direct transition from `ns` to `exe`. c is a clock that counts the time spent in state `wt`. To simplify the figure a single red line, meaning that risk is too high and execution must be aborted, replaces a collection of red edges from all states but `dn` to `exit` state.

If a task contains a loop of n iterations, n separate sets of actions are independently instantiated—i.e., a_i and a_{i+n} are two separate actions with identical definitions. Note that n is a constant parameter whose value is determined before the safety analysis is carried out.

2) Model of Operator Actions: The behavior of the operator is nondeterministic. It is unrealistic to impose that the operator starts execution of an action as its preconditions are satisfied. Indeed, the execution of actions by the operator might not be as temporally accurate as for the robot. Delays could affect the correct timing of the task without compromising its completion (e.g., slight distraction, plain waiting). Hence, in addition to preconditions, an operator action depends also on a new predicate $\mathsf{o}pStarts$, which captures the time operator actually starts to act. As shown in formula (6), a timeout equal to Δ time units is assigned to every operator action, which starts being counted as soon as all preconditions of the action hold. The transition `wt` $\rightarrow$ `exe` is triggered when the operator starts executing the action ($\mathsf{o}pStarts$) within Δ time units. Otherwise, the action returns to state `ns`

$$\begin{pmatrix} a_i^{\text{agent}} = \mathtt{op} \wedge \\ a_i^{sts} = \mathtt{wt} \end{pmatrix} \Rightarrow \begin{pmatrix} \mathsf{WithinF}(a_i^{sts} = \mathtt{exe}, \Delta) \vee \\ \mathsf{Futr}(a_i^{sts} = \mathtt{ns}, \Delta + 1) \end{pmatrix}. \quad (6)$$

Another assumption included in the model states that, unlike the robot, the operator is able to execute up to two actions at a time. It is stated by the formula mentioned below, which defines that there cannot be three different operator actions executing simultaneously, thus, keeping the maximum number of concurrently executing operator actions to two as

$$\neg\exists i,j,k \begin{pmatrix} i < j < k \wedge a_z^{\text{agent}} = \mathtt{op} \\ \bigwedge_{z \in \{i,j,k\}} \left(a_z^{sts} = \mathtt{exe} \vee a_z^{sts} = \mathtt{sfex}\right) \end{pmatrix}. \quad (7)$$

If multiple operator actions are in state `wt`, at most two of them will nondeterministically—and not necessarily simultaneously—enter state `exe` within Δ time units. Moreover, a human operator could manifest erroneous behavior that affects the completion of the task. This includes: repetitions of the same action, omissions, reversal of order, replacement, insertion [61]. The inclusion of state `wt` helps us to generate some of these error types (omission, insertion). This issue has been tackled more thoroughly in [62].

VI. Risk Assessment Modules

In order to automate the risk assessment procedure of ISO 12100, the formal verification routine of SAFER-HRC explores all histories H of the model in search for hazards and their associated risks. Each phase of risk assessment is modeled by one of the modules explained in the following. If a history of the model follows the definition of a hazardous situation declared in Section VI-A, then its risk is estimated by the formula included in Section VI-B. If the estimated risk value is above a predefined threshold, the mechanisms introduced in Section VI-C would find it and, thus, highlight the necessity of some RRM. Then, a suitable constraint as defined in Section VI-D is introduced in the model.

The formal model of RRMs provides an abstraction of safety functions [63]. A safety function is deployed in practice to "solve" a nonnegligible risk and can be implemented in different ways, using various input, output, and logic components (e.g., sensors, controllers, motors, and mechanical devices) whose reliability level is identified by functional safety requirements. The behavior (correct or faulty) of such hardware/software components is modeled through the information they generate: for example, a correct detection by a position sensor is captured by a predicate representing the information "operator one is in region L_1" being true. Depending on the hazards detected by the verification runs, different selections of RRMs are possible, according to the chosen policy for solving risks.

TABLE II
PREDICATES USED TO FORMALIZE THE DEFINITION OF HAZARDOUS SITUATIONS BETWEEN R_j, O_i, AND L_k

Predicate Formula	Definition
$\mathsf{Sep}_{ij} \in \{\texttt{close}, \texttt{mid}, \texttt{far}\}$	a discrete value to represent the distance of R_j and O_i. $\mathsf{Sep}_{ij} = \texttt{close}$ means that the two elements are in he same location ($p_i = p_j = L_k$); $\mathsf{Sep}_{ij} = \texttt{mid}$ means they are in adjacent locations; and $\mathsf{Sep}_{ij} = \texttt{far}$ means there is at least one location in between locations of i and j.
$\mathsf{InSameL}_{ijk} \Leftrightarrow \mathsf{Sep}_{ij} = \texttt{close}$	R_j and O_i are both located in L_k.
$\mathsf{ArrivedBefore}_{ijk} \Leftrightarrow \mathsf{InSameL}_{ijk} \wedge \mathsf{Past}(p_i = L_k, 1) \wedge \mathsf{Past}(p_j \neq L_k, 1)$	O_i entered L_k right before R_j.
$\mathsf{Reach}_{ijk} \Leftrightarrow \mathsf{InSameL}_{ijk} \wedge \mathsf{Past}(\mathsf{Sep}_{ij} > close, 1)$	R_j and O_i just arrived in L_k together.
$\mathsf{Leave}_{ijk} \Leftrightarrow \mathsf{Past}(\mathsf{InSameL}_{ijk}, 1) \wedge p_i \neq L_k \wedge p_j \neq L_k$	R_j and O_i both just left L_k.
$\mathsf{Contact}_{ijk} \Leftrightarrow \mathsf{InSameL}_{ijk} \wedge \mathsf{moving}_{R_j}$	contacts occur when R_j is functioning in close proximity with O_i in L_k ($\neg(L_{k,obst} = \texttt{free})$).

If the model fails to satisfy a certain safety property, it means the model requires some refinements in terms of adding new RRMs, modifying properties of $\mathcal{O}$, $\mathcal{R}$, and $\mathcal{L}$ or pre/postconditions of actions.

A. Hazard Identification Module

This article focuses on mechanical hazards such as impacts, shearing, crushing, and cutting (see ISO:12100 Annex B). The principle is nevertheless extensible to other types of hazards, once they are formalized through their properties and relevant attributes. Each mechanical hazard has a formal definition and is identified by a Boolean predicate hzd^{*}_{ijk}, where $*$ is the placeholder for the hazard-type label to be qualified, and ijk are, respectively, the identifiers of human body part, the robot part, and location involved in the hazardous situation. The formal model includes a finite set of hazard definitions. Whenever one of them is satisfied in a state, the corresponding predicate hzd^{*}_{ijk} is true, which captures the fact that the hazardous situation occurs in the system. Mechanical hazards arise from the relative positions of body parts, robot parts (see Section V-A), and objects, as well as from the dynamics of such mutual relationships among $\mathcal{O}$, $\mathcal{R}$, and $\mathcal{L}$. Hence, some predicates have been introduced in the formal model in order to express these spatial configurations and to capture the definitions of hazards. These predicates are described in Table II for given $O_i \in \mathcal{O}$, $R_j \in \mathcal{R}$, and $L_k \in \mathcal{L}$. For compatibility with recent standardization efforts in industrial robotics [25], contact hazards involving operators and collaborative robots are classified into two groups: first, fast, impact like contacts, where body parts are hit and then recoil because of the kinetic energy transferred to the body, defined as *transient* (Tr) contacts; second, sustained contacts of body parts against a constraining object with continuous energy flow from the robot, defined as *quasistatic* (Qs). In Qs-type situations, robot forces distribute as pressure on affected body areas with a slow/negligible displacement, which depends on the mechanical impedance of the body part. Thus, Tr and Qs hazards between R_j, O_i in L_k are formalized as follows.

Definition 9: As defined by the next formula, a *Quasistatic hazard* $\mathsf{hzd}^{\texttt{Qs}}_{ijk}$ occurs when two elements approach each other in an occluded location. It can also happen in a nonocclusion region when multiple links of a robot are in the same location and fold in a way that could entrap a human body part as

$$\begin{aligned}&\mathsf{hzd}^{\texttt{Qs}}_{ijk} \Leftrightarrow \\ &\mathsf{Contact}_{ijk} \wedge \mathsf{Reach}_{ijk} \wedge \mathsf{ArrivedBefore}_{ijk} \wedge \\ &\big(obst_k = \texttt{occluded} \vee \exists R_m \in \mathcal{R}(\mathsf{InSameL}_{imk} \wedge i \neq m)\big).\end{aligned} \tag{8}$$

Transient contacts are, instead, revealed by a nonsustained contact in a region without occlusions.

Definition 10: The next formula defines that a *Transient hazard* $\mathsf{hzd}^{\texttt{Tr}}_{ijk}$ occurs when two elements O_i and R_j reach a clear location shown as

$$\begin{aligned}&\mathsf{hzd}^{\texttt{Tr}}_{ijk} \Leftrightarrow \\ &\mathsf{Contact}_{ijk} \wedge \mathsf{Reach}_{ijk} \wedge obst_k = \texttt{clear} \wedge \\ &\mathsf{Futr}(\mathsf{Leave}_{ijk}, 1) \wedge \nexists R_m \in \mathcal{R}(\mathsf{Reach}_{imk} \wedge j \neq m).\end{aligned} \tag{9}$$

Depending on the geometry and kinematics of the robot, the granularity with which the layout $\mathcal{L}$ is partitioned is essential to guarantee that the conditions in formula (8) and (9) capture real hazardous situations.

Example 1 (Hazard identification): Fig. 3 represents a typical HRC scenario in which the robot is supposed to execute a_n by R_j ($j =$ tool), whereas operator executes a_m by O_i ($i =$ hand). In fact the robot and operator need to *concurrently* move from L_2 to L_1 and, thus, $a^{sts}_m = \texttt{exe} \in preC_n$. Recall that the layout is divided in regions $L_1 = \langle \texttt{round}, \texttt{hard}, \texttt{occluded} \rangle$ and $L_2 = \langle *, *, \texttt{clear} \rangle$, and also that $a^{exeT}_n = T_n$ and $a^{exeT}_m = T_m$ hold.

Consider a history H in which the precondition of action a_m holds at time t. The operator has Δ time units to start executing a_m and consequently different contact combinations might happen. If the timeout expires, $a^{sts}_m = \texttt{ns}$ becomes true and a_n cannot start cause its precondition does not hold. Otherwise (the operator timely starts a_m at $t < t_1 < t + \Delta$), one of the following situations occurs.

1) At $t_2 > t_1$, O_i and R_j both reach L_2 before reaching the target (L_1). Thus, the configuration of the state at time t_2—i.e., ${}^{H}\overline{s}(t_2)$—matches with formula (9) and ${}^{H}\overline{\mathsf{hzd}^{\texttt{Tr}}_{ij2}}(t_2)$ holds.
2) At $t_2 > t_1$, O_i arrives at L_1 before R_j. Thus, ${}^{H}\overline{s}(t_2)$ matches with formula (8), and ${}^{H}\overline{\mathsf{hzd}^{\texttt{Qs}}_{ij1}}(t_2)$ holds.

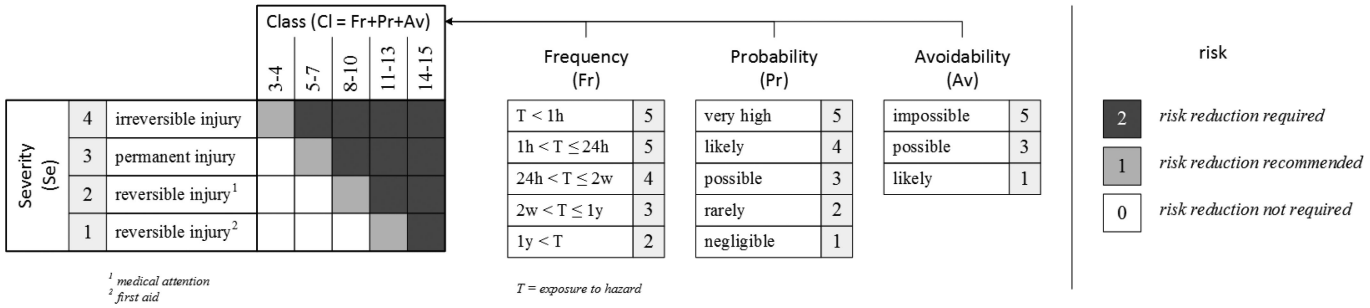


Fig. 6. Reference matrix for the hybrid method of risk estimation [50, Sec. A.7, example Table A.19].

3) O_i reaches L_1 after R_j. Thus, $t_1 < t_i < t_1 + \max\{T_n, T_m\}$ does not match any contact conditions.

B. Risk Estimation Module

Once the hazards are identified, their associated risks need to be computed according to the normative principles of risk estimation [1, Sec. 5.5.2]. The risk value of hzd^*_{ijk} is represented by risk_{ijk}, which is a compound notion of *severity* and *probability of occurrence*. Severity states the effects of hazards in terms of lesions or damages, while probability of occurrence is an indicator of the hazard's likelihood in time and space, or the possibility to being exposed to hazards due to errors and failures, or the chance to avoid hazards due to layout conditions. In case the risk is estimated to be nonnegligible, RRMs should react by reducing one of the two factors—or even both, if necessary.

SAFER-HRC adopts the hybrid risk estimation method defined in ISO/TR 14121-2 [50, Sec. 6.5] that combines a risk matrix approach with quantitative risk factors: severity Se, frequency Fr, probability Pr, and avoidability Av. As shown in Fig. 6, the latter three factors are combined to set an identifier ($Cl = Fr + Pr + Av$) that, together with the severity value, determines the final risk. The depicted matrix in Fig. 6 replaces continuous figures (e.g., probability distributions) with qualitative notions (*likely*, *possible*, *rarely*, etc.) whose values correspond to positive integer numbers. It is noteworthy that probability scores are rarely derived from a known distribution,[2] but, rather, they are associated with statistical data about accidents in similar situations and frequent human misbehavior.

1) Initial Risk Estimation: In HRC applications, it is very common to have permanent exposure to robot systems due to the sharing of the same workspace as part of the intended use—i.e., the frequency value Fr is assumed to be always maximum and equal to the initial one Fr^0 (where the $\{\cdot\}^0$ superscript denotes the initial value of risk factors).

The probability factor Pr considers the human behavior, the reliability of components, the history of accidents, etc. The Pr score is then very much influenced by both the quality of the safety-related equipment and the skill/awareness level of the operator. In the absence of any active mean intended to solve machine failures and human errors, the default score for the probability is assumed to be $Pr^0 \geq 2$.

TABLE III
VALUES IN INITIAL RISK ESTIMATION

	Se^0_{ijk}	Fr^0_{ijk}	Pr^0_{ijk}	Av^0_{ijk}	risk^0_{ijk}
initial value	$\geqslant 3$	5	$\geqslant 2$	5	2 (non-neg.)

Similarly, without alerting or training, nor active safety, the default chance of avoidance is negligible—i.e., $Av^0 = 5$.

The starting estimate of risks associated with hazards does not take into account any risk mitigation factor and assigns a high severity value due to the presence of moving machines (default hypothesis in safety of machinery). As a result, the initial risk is very often nonnegligible (see Table III), due also to its occurrence factor—i.e., for all hazard hzd^*_{ijk}, it holds that $Cl^0 \geq 12$. Instead of adopting safeguarding RRMs, which remove hazards, but prevent the collaborative sharing of spaces and functions, HRC hazards are further analyzed in terms of their effects (i.e., the severity factor) with specific focus on physical human–robot interactions. In the following, we provide an example of more detailed model of severity of contact hazards, depending on the involved physical quantities (shape of the contact surfaces, velocity, force, etc.).

2) Risk Estimation: Pain and Pressure: The general principle in HRC physical interaction [64] is that the operator should never get injured. However, ISO/TS 15066 claims that occasional accidental contacts would produce just a sensation of limited pain that is not different from the sensation generated by contacts in activities of daily life and could be captured by $Se < 3$. There have been several studies [60], [65]–[67] to set empirical injury scales. Experiments on dummies and cadavers are reported in [64]. Although the biomechanics of pain is far from being fully understood, the primary figure for painful interaction is known to be pressure, i.e., the contact force distributed through the interaction area.

Pain derives[3] from the energy or power flux density from the robot into tissues of different masses, stiffnesses, and sensitivities to pain, depending on the dynamics of the contact. In particular, flux densities are dominated by kinetic energy transfer in transient contacts or by mechanical work in quasi-static constraining. The formal model contains an attribute called

[2]This is one of the main reasons why *quantitative* methods in ISO/TR 14121-2 [50] are hardly used in manual assessments.

[3]From standardization perspective only. Cited ongoing studies are dedicated to investigate pain onset and bio-mechanical implications.

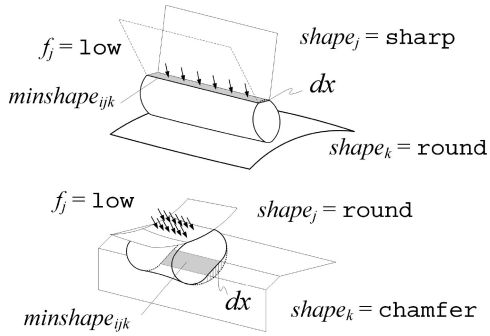


Fig. 7. Pressure resulting from force f_j on surfaces $minshape_{ijk}$ (shadow area) for quasi-static contacts—O_i is always between R_j and L_k. The length of arrows is proportional to the force magnitude. The deformation of the body part under load dx is proportional to the pressure.

TABLE IV
MAPPING BETWEEN CONCRETE AND DISCRETE f_j VALUES, VALID FOR HUMAN HAND PALM

interaction area cm^2	$minshape$	$f_j :=$ low	$f_j :=$ mid
$5 \leqslant A < 15$	fillet	800N	3900N
$1 \leqslant A < 5$	chamfer	400N	–
$A < 1$	sharp	70N	–

$minshape_{ijk} \in \{\texttt{sharp}, \texttt{chamfer}, \texttt{fillet}, \texttt{round}\}$, which is a combinatorial value of shapes of R_j, O_i, and L_k and symbolizes the surface of the contact area. Once $minshape_{ijk}$ is known, the effect of the distribution of forces over surfaces and the caused pain are computable. Fig. 7 shows a few examples in which the exposed area to the contact ($minshape$) affects the pressure value.

When a Qs hazard occurs, the force f_j applied by the robot causes a pressure, captured by a set of discrete values $P \in \{\texttt{low}, \texttt{mid}, \texttt{high}\}$. The only case for negligible pain, formalized below, is $(P = \texttt{low})$ because it could only cause $Se < 3$. The force values in formula (10) should be initialized based on experimental pain studies. However, additional limitations could be considered if deemed necessary by the safety team. Table IV draws an example of a mapping between actual force values and the discrete values of the formal model, in the case of the palm of a human hand as

$$P_{ijk}^{\texttt{Qs}} = \texttt{low} \Leftrightarrow$$
$$((f_j = \texttt{low} | f_j = \texttt{mid}) \wedge (minshape_{ijk} = (\texttt{round}|\texttt{fillet})))$$
$$\vee (f_j = \texttt{low} \wedge (minshape_{ijk} = (\texttt{chamfer}|\texttt{sharp}))). \tag{10}$$

When a Tr hazard occurs, the actual pain is believed to originate from the pressure on body tissues, whose deformation dx increases as the stiffness k_i of body part O_i decreases. Under the conservative hypothesis[4] that the kinetic energy is transformed into elastic energy, and that the momentum is preserved through the instantaneous impact, the transient contact is a totally inelastic impact, starting at relative velocity v_{ij} between R_j and O_i, and can be written as

$$E = \frac{(P_i A)^2}{k_i} = \frac{m' v_{ij}^2}{2}, \qquad v_{ij} = \frac{P_i A}{\sqrt{k_i \frac{m'}{2}}} \tag{11}$$

where k_i is the stiffness of the approximate spring-damped model of the O_i.

$P_{ijk}^{\texttt{Tr}}$ captures the pain effects on the human body as a result of the compression during the impact and, like $P_{ijk}^{\texttt{Qs}}$, is discretized and formalized through a set of formulae; for example, the only case with a negligible risk value, which corresponds to $Se < 3$ captures the following formula:

$$P_{ij}^{\texttt{Tr}} = \texttt{low} \Leftrightarrow (\mathsf{m}' = \texttt{low} \wedge (v_{ij} = \texttt{mid}|\texttt{high})) \vee$$
$$((\mathsf{m}' = \texttt{mid}|\texttt{high}) \wedge v_{ij} = \texttt{low}) \vee (\mathsf{m}' = \texttt{low} \wedge v_{ij} = \texttt{low}). \tag{12}$$

In conclusion, key elements for computing severity are robot force and velocity (v_j, f_j) in each state s, and the geometries of $\mathcal{R}$ and $\mathcal{L}$ rendered by $shape_j$, $shape_k$ attributes and ultimately expressed through predicate $minshape_{ijk}$. According to Fig. 6, even the occurrences of low-severity hazards should be analyzed carefully because their combination with high probability yields nonnegligible risk—i.e., $\mathsf{risk} > \tau$. A typical example is the continuous exposure to unwanted contacts, which are not tolerable for ergonomics and health reasons [68].

```
----------------------------time 10----------------------------
********************** Executing state **********************
Actions 1,2,3 are terminated
Action 4 is executing: operator is moving to the pallet (L_1_3)
Actions 5,6,7,8,9,10,11,12,13,14 are not started
********* Human state ***************** Robot state *********
Operator is moving          **  Robot is not moving
Head_area is in L_2_2       **  Link1 is in homing position (L_0)
Arm_area is in L_2_2        **  Link2 is in homing position (L_0)
Leg_area is in L_2_1        **  EndEff is in homing position (L_0)
*** Relative properties ********* Safety state ***
EndEff is far               **  detected hazards: none
Link1 is far                **  Risk is 0
Link2 is far                **
Relative velocity is high   **
Relative force is none      **
```

Fig. 8. Example of a report on system state produced by the tool at a given time instant ($t = 10$).

C. Risk Evaluation

A system designer should decide how conservative the system should be, and define a safety property accordingly. The global state of the model in every instant of a history should be checked against this property through an automated formal verification tool—Zot [53] in our case—to determine whether the property is violated or not. Fig. 8 shows the customized output of Zot, which shows the state of the system at a time instant.

A simplified example of a safety property is provided in formula (1). Here, given the explanation of our formalism, we introduce the actual safety property used in our approach, which captures the complexity of HRC contacts more properly. The formulae of the model explicitly state that, when the risk of a

[4]Under the assumption that the system is isolated, any energy generated/dissipated by robot control is not modeled, so that the energy exchange between body and robot includes all the energy.

hazard reaches the acceptability threshold (τ), the associated RRM should be triggered. The purpose of defining the formula below is to check if each risk is suitably handled by at least one RRM, and it does not stay above the acceptability threshold for longer than one instant,[5] shown as follows:

$$\mathsf{Alw}\left(\begin{array}{l}\mathsf{risk}_{ijk} \leq \tau \vee \\ \\ \mathsf{risk}_{ijk} > \tau \wedge \exists y \left(\begin{array}{l}\mathsf{RRM}^{y}_{ijk} \wedge \\ \mathsf{Futr}(\mathsf{risk}_{ijk} \leq \tau, 1)\end{array}\right)\end{array}\right). \quad (13)$$

If formula (13) is not satisfied, it means that there is an RRM, which is not strong enough, or that there is a hazard that has been left without mitigation. Thus, to satisfy this formula, we need to define RRMs. They could be initially defined and included in the model, as we will see in Section VI-D, or eventually come to our attention after several iterations of verification process, as it will be discussed in Section VI-E.

D. Risk Reduction

The safety assessment approach presented in this article supports a dynamic mechanism for activating/deactivating RRMs, so that they reduce the application's risks only when necessary. Sometimes RRMs may unacceptably downgrade the *entire* performance of the application. In general, RRMs mitigate risky hazards by the following:

1) reducing their probability (e.g., changing plan or workflow, preventing failures, etc.);
2) reducing their avoidability providing escaping conditions;
3) reducing or eliminating the lesion on the human body in case of hazardous event.

An RRM is "immediately" applied when a high level of risk is detected, and consequently, the risk value should go below the threshold "right after" the RRM starts. This is captured by the following formula where y identifies the type of the employed RRM. In our formalization "immediately" and "right after" are formalized as "simultaneously"—i.e., $\mathsf{risk}_{ijk} > \tau \wedge \exists y(\mathsf{RRM}^{y}_{ijk})$—and "at the next instant"—i.e., $\mathsf{Futr}(\mathsf{risk}_{ijk} \leq \tau, 1)$. Alternatively, one could have employed the more sophisticated notion of *zero-time transitions* [69], [70], where phenomena occur at vanishingly close time instants, which are still separate

$$\mathsf{risk}_{ijk} > \tau \Rightarrow \exists y(\mathsf{RRM}^{y}_{ijk}). \quad (14)$$

The rest of this section introduces formalizations for *some* RRMs, particularly relevant for HRC situations.

Definition 11 (Speed and Separation Monitoring (SSM)): Contact-less collaborative modes can be used for protecting the operator by tuning the robot speed to maintain a minimum safe distance [24] from the operator as in the following formula. The safety team can purposefully select an RRM that tunes distance and/or velocity according to the following formula:

$$\begin{array}{l}\mathsf{RRM}^{\mathsf{SSM}}_{ijk} \Rightarrow \\ (\mathsf{Sep}_{ij} < \mathsf{Sep}_{\text{min}} \Rightarrow \mathsf{Futr}(v_{ij} = \texttt{none}, 1)) \wedge \\ (\mathsf{Sep}_{\text{min}} \leq \mathsf{Sep}_{ij} \leq \mathsf{Sep}_{\text{mid}} \Rightarrow \mathsf{Futr}(v_{ij} \leq \texttt{mid}, 1))\end{array} \quad (15)$$

where $\mathsf{Sep}_{\text{min}}$ and $\mathsf{Sep}_{\text{mid}}$ are two thresholds on the distance between O_i and R_j, moving at relative speed v_{ij}. $\mathsf{Sep}_{\text{mid}}$ is used as a threshold for slowing down R_i, so to maintain a $\mathsf{Sep}_{\text{min}}$ threshold. When $\mathsf{Sep}_{\text{min}}$ is violated, a protection stop is triggered.

The details about the computation of relative velocity and separation distance in continuous kinematics are available in ISO/TS 15066 [25, Sec. 5.5.4]. For the consistency of formula (15) in implementations, it is essential that, first, the position of both O_i and R_j is measured by monitoring all the $\mathcal{L}$ locations, and second, the robot is equipped with some safety function able to limit its velocity. However, the formula is general enough to be independent from the specific implementation of the SSM strategy on actual hardware. It is able to capture either the case of keeping a constant distance through proximity sensing, considering a default human speed of 1.6 m/s ISO/TS 15066, ISO 13855 [71], or the case of dynamically updating the separation distances through online human tracking [72]. Mutual positions can be measured through any kind of environmental sensors (e.g., laser scanners) or on-board proximity sensing, while the ability to limit the velocity to a given value is part of the functions offered by the safety-related part of the robot control system (e.g., EN 61800-5-2 safely-limited speed). During the discretization of velocities for formula (15), a numerical correspondence with current technology is straightforward: robot speeds exceeding 600–800 mm/s can reasonably be defined as fast for lightweight low-payload robots, whereas 2 m/s can be assumed as the fast threshold for traditional industrial medium/high-payload robots. Similarly, $v_j = \texttt{low}$ may be assigned to $\leq$250 mm/s, corresponding to a well-established value of low speed for all classes of robots.

Avoiding collisions may be hardly practicable in crowded/narrow environments, or accidental minor contacts could be unavoidable. An alternative strategy is to reduce the severity Se through, e.g., the power and force limiting (PFL) collaborative mode [25, Sec. 5.5.5], derived from the recent experience in industrial domains.

Definition 12 (PFL quasi-static): The suitable RRM for a quasi-static hazard $\mathsf{hzd}^{\mathtt{Qs}}_{ijk}$ is a force-limiting PFL ($\mathtt{PFL}_f$ for short) protection measure as

$$\mathsf{RRM}^{\mathtt{PFL}_f}_{ijk} \Rightarrow P^{\mathtt{Qs}}_{ijk} = \texttt{low}. \quad (16)$$

The effect of $\mathtt{PFL}_f$ is to reduce the pressure on a body part until a safe threshold is reached, i.e., attaining the condition in formula (10). The continuous model of a constrained contact is depicted in Fig. 9, with the RRM-enabling condition represented by Ⓓ. The RRM is enforced by control functions able to detect and limit the force f_i on robot links, or by intrinsically safe features (e.g., compliant actuation), or both.

[5] As a consequence of changing the property to be verified from formula (1) to formula (13), Fig. 5 needs to be adjusted. More precisely, the exit constraint $\mathbb{I}$ is replaced by formula $\mathsf{risk}_{ijk} > \tau \wedge \neg\exists y \left(\mathsf{RRM}^{y}_{ijk} \wedge \mathsf{Futr}(\mathsf{risk}_{ijk} \leq \tau, 1)\right)$.

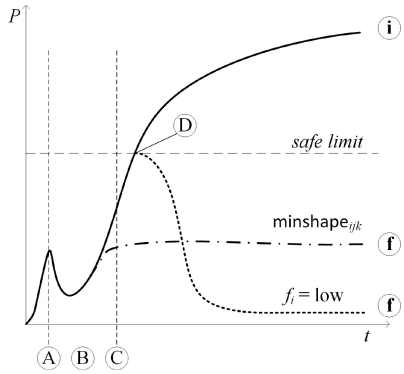


Fig. 9. Typical pressure pattern (solid line) on a body part of a given surface and stiffness, in case of contact: at Ⓐ the body loses contact because of the impact, recoils in Ⓑ, before any control action can take place after Ⓒ (reaction time). From starting condition ⓘ, the RRM leading to safe ⓕ condition can be done by limiting/detaching the robot force/torque (dotted line) or passively distributing the pressure on larger surfaces (dot-dashed line). Intrinsically safe (low-power or compliant) robots behave as the latter case.

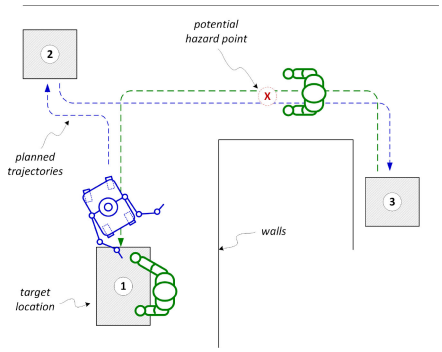


Fig. 10. Example of a task involving a mobile dual-arm robot, (self) planning its motion/mission in an environment shared with human operators.

Definition 13 (PFL dynamic): The suitable RRM for a transient hazard $\mathsf{hzd}^{\mathtt{Tr}}_{ijk}$ is a speed-limiting $\mathtt{PFL}$ ($\mathtt{PFL}_v$ for short) protection measure as

$$\mathsf{RRM}^{\mathtt{PFL}_v}_{ijk} \Rightarrow v^{\mathtt{Tr}}_{ij} = \mathtt{low}. \tag{17}$$

The effect of $\mathtt{PFL}_v$ is to reduce the momentum transfer, and consequently the impact power through the interacting surface on body parts. Given that the reduced masses are constant, the RRM is obtained by tuning v_i, and consequently v_{ij}, as in formula (12).

E. Iterations in Estimation and Reduction of Risks

When necessary upon risk estimation, one or more RRMs are implemented[6] by updating the properties of the system components. These updates entail that a new model is produced, corresponding to a new set of histories; hence, a new formal verification activity is performed. The process explained above is done systematically in a so-called push-button way, which requires minimum manual interference to refine the model at each iteration.

Example 2: Pursuing example 1, the results of the risk analysis for identified hazardous cases can be reported as the following.

1) The RRM introduced for the hazard detected in the first case of the example (i.e., ${}^{H_1}\overline{\mathsf{hzd}^{\mathtt{Tr}}_{ijk}}(t_2)$) is $\mathsf{RRM}^{\mathtt{SSM}}_{ijk}$, which tackles the hazard by reducing the probability of its occurrence.
2) The hazard detected in the second case (i.e., ${}^{H_2}\overline{\mathsf{hzd}^{\mathtt{Qs}}_{ijk}}(t_2)$) is dealt with through $\mathsf{RRM}^{\mathtt{PFL}_v}_{ijk}$. This RRM reduces the risk by weakening its severity.

[6]Functions implementing RRMs shall comply with functional safety requirements in [24, Sec. 5.5.2]. It is, however, out of the scope of the model and the formal verification tool to verify the compliance of safety-related parts of the control systems and all other safety functions to the functional safety standards (e.g., [63]). Each RRM is assumed to be safely deployed. See also Section VIII-A.

Let us emphasize that the aforementioned types of RRMs of Section VI-D are only some examples among several other possible ones, which have different effects on the risk levels and on the performance of the application.

The selection of RRMs in different situations depends on the safety strategy chosen by a safety analyzer or even by an automated reasoner. The result could lead to different courses of execution: one model instance may be forced to completely prevent physical contacts by implementing, e.g., collision avoidance strategies, while other options may allow physical contacts (e.g., due to the instructions of the executing action), but restraining the relative physical parameters. The incremental nature of the formal verification allows us to model both strategies (contact-tolerant versus contact-less), to verify the formal model in either case, and to enable the identification of the best strategy for various situations.

A related point to consider when choosing RRMs, in fact, is the required efficiency in terms of feasibility and execution time of a given action under the effects of an RRM. For example, an action might demand a full speed performance by the robot (e.g., no constraints are possible on velocity), or it might tolerate some level of delay caused by constraining the physical parameters of the system. The effects of the safety strategy and associated RRM selection on the verification outcome is shown through the following example.

Example 3: Fig. 10 shows an environment in which two human operators and a dual-arm robot mounted on a mobile platform work together. Assume that the robot, which is now located in location ①, should move to location ②, and then from there to ③. A collision between human (the one who is not in ①) and robot could appear at location Ⓧ while robot is moving from ② to ③ and the operator is moving from ③ to ①. There are three possible safety strategies—and corresponding RRMs—to deal with this collision.

a) Contact-less safety strategy: If contacts are to be avoided, whenever the robot and operator's distance goes below a certain threshold, the RRM in formula (15) slows down or stops the robot. Thus, robot and operator will never meet whilst the robot is moving too fast.

b) Contact-tolerant safety strategy: Considering the layout in the figure, it seems hard to avoid contacts. Alternatively, the

RRM in formula (17) could reduce the average velocity along trajectories of predictable human movements (e.g., around Ⓧ) so that harmless physical contact occurs along colliding patterns.

c) Safety by planning/control: Another way of avoiding potentially frequent contacts, which might compromise the efficiency of the task execution, is to plan the robot to stop in ② until the operator arrives at ①. Then, it starts moving toward ③. This way the collision hazard at Ⓧ never appears, but the execution time might considerably increase. This strategy belongs to a class of planning activities because it is a purposeful rearrangement of the sequence of actions, redefining the intended use. The selected RRM can be modeled as a new precondition for action "move to ③" of the robot. If we indicate "robot moves from ② to ③" as a_i, and "operator moves from ③ to ②" as a_j then the risk reduction can be formalized as $preC_i \Rightarrow a_j^{sts} = \texttt{dn}$.

VII. Experimental Case Study

This section illustrates how SAFER-HRC works in practice by applying it to a realistic case study of collaborative fixture assembly in a flexible manufacturing system (FMS). The preparation of machine tool pallets—i.e., setting jigs and mounting (dismounting) workpieces into (from) fixtures before (after) machining—is a skillful and heavy job. Hence, collaborative robots are used to assist operators in tasks such as carrying/loading tools or containers, supporting workpieces during assembly, applying tools.

We consider a scenario in which the robotic system provides all services that improve the ergonomics of manual operations and release the operators from repetitive or heavy tasks. The robotic system is a manipulator arm mounted on a mobile unit (base) that automatically relocates within a workspace as the one shown in Fig. 11(a). Regularly, the robot is positioned in front of the two assembly stations ① and ②, or close to an instrumented inspection station ③. The robot unit can nonetheless travel and access the whole workspace, including a loading/unloading area for raw materials and finished parts, and can be manually adjusted by operators around its programmed positions (e.g., to compensate positioning/reaching limits or errors).

Two human operators ($\texttt{op}_1$ and $\texttt{op}_2$) are employed in the application. $\texttt{op}_1$ is mostly present on stations ① and ②, while $\texttt{op}_2$ works mainly on ③ or executes auxiliary manual tasks on the workbench in ④. Both operators can freely hold and resume their tasks, swap posts, or join one another in some area. The main robot-assisted intended tasks are: pallet assembly at stations ① and ②, including bin-picking from local storage carried by the mobile unit; pallet disassembly at ① and ②, including bin-dumping; pallet inspection at station ③; lead-through programming of assembly, disassembly, and inspection tasks (trajectories, parameters, etc.) at stations ①, ②, and ③; material handling on load/unload areas.

Other manual tasks by $\texttt{op}_1$ and $\texttt{op}_2$ include manual loading of parts/boxes; (additional) visual inspection of pallet at stations ①, ②, and ③; manual assembly/disassembly of pallet at stations ① and ②; manual measurements of parts at station ④; cleaning pallets at stations ① and ②; kitting of tools and parts at stations ①, ②, and ③; general supervision (programming other tasks at HMI during operations, consulting production data at HMI, etc.) at stations ①, ②, and ③. Note that *all* combinations of robot/manual task assignments are admitted (e.g., robot holds and $\texttt{op}_1$ screw-drives jigs and vice versa, switching tasks on the fly, quitting a manual task and assigning the robot to proceed autonomously). Frequently, robot base and operators move side-to-side across the central aisle, or other operators transit along the aisle because the target area is part of a larger plant and access to it is not restricted. See Fig. 11(b) for reference layout and situations.

A. Model Generation

First, two instances of $\mathcal{O}$ and a $\mathcal{R}$ model with four main parts ($R_1, R_2, R_{ee}, R_{base}$) are initialized. The second step to apply our methodology is the creation of $\mathcal{L}$, which provides a discretized and abstract representation of the workspace, as in Fig. 11(c). The partitioning of the layout in regions is a design step that is configured manually.

The size of the regions is relative to the task and the surrounding environment; more precisely, starting from the established layout, boundaries are set for separating zones dedicated to process, free motions, etc. The location of the boundaries is relative to the size and expected (discretized) speed of the robot system: in our case study, a manipulator able to reach up to 1.5 m is used for lightweight tasks moving in the speed range of 500–1000 mm/s; hence, an area of 500-mm depth (i.e., one third of the robot reach) around a fixture or fixed obstacle is on average considered enough for taking into account potential close-collisions or crushing. In the open side of a working location, instead, the same robot system may enjoy regions twice or three times as larger (e.g., up to the full extension of the robot arm) for free movements. Regions where the operator is moving as part of the intended use, may be up to 1 m wide—i.e., 0.5 s of fast pace walking at 1600 mm/s—while locations for standing manipulation are in general smaller because all the process and potential interactions are likely to happen in narrow spaces.

For example, red regions in Fig. 11(c) are smaller, since planned movements in front of stations ① and ② involve small displacements. Conversely, regions far from the stations are larger, because larger displacements are programmed there.

The 3-D model of the $\mathcal{L}$ workspace is rendered with regions of three stacked layers: lower, middle, and upper regions. The mobile robot base has mobility in the low layers of the regions, while the manipulator arm can reach middle and upper layers, in each region. The human body is discretized in head, chest, leg, arm, and fingers segments. Assuming, e.g., a person standing, the lower sections reach up to operators' hips, the middle sections cover torsos up to shoulders (where chest, arm, and fingers segments are expected), and the upper sections are supposed to account for the position of operators' heads. The head can (hazardously) get down to the middle layers. Naturally, arms and fingers are the most mobile human parts.

The third step is to model the tasks, e.g., to create activity diagrams of the whole application, identify atomic actions, and formalize their pre/postconditions. In the following, the model

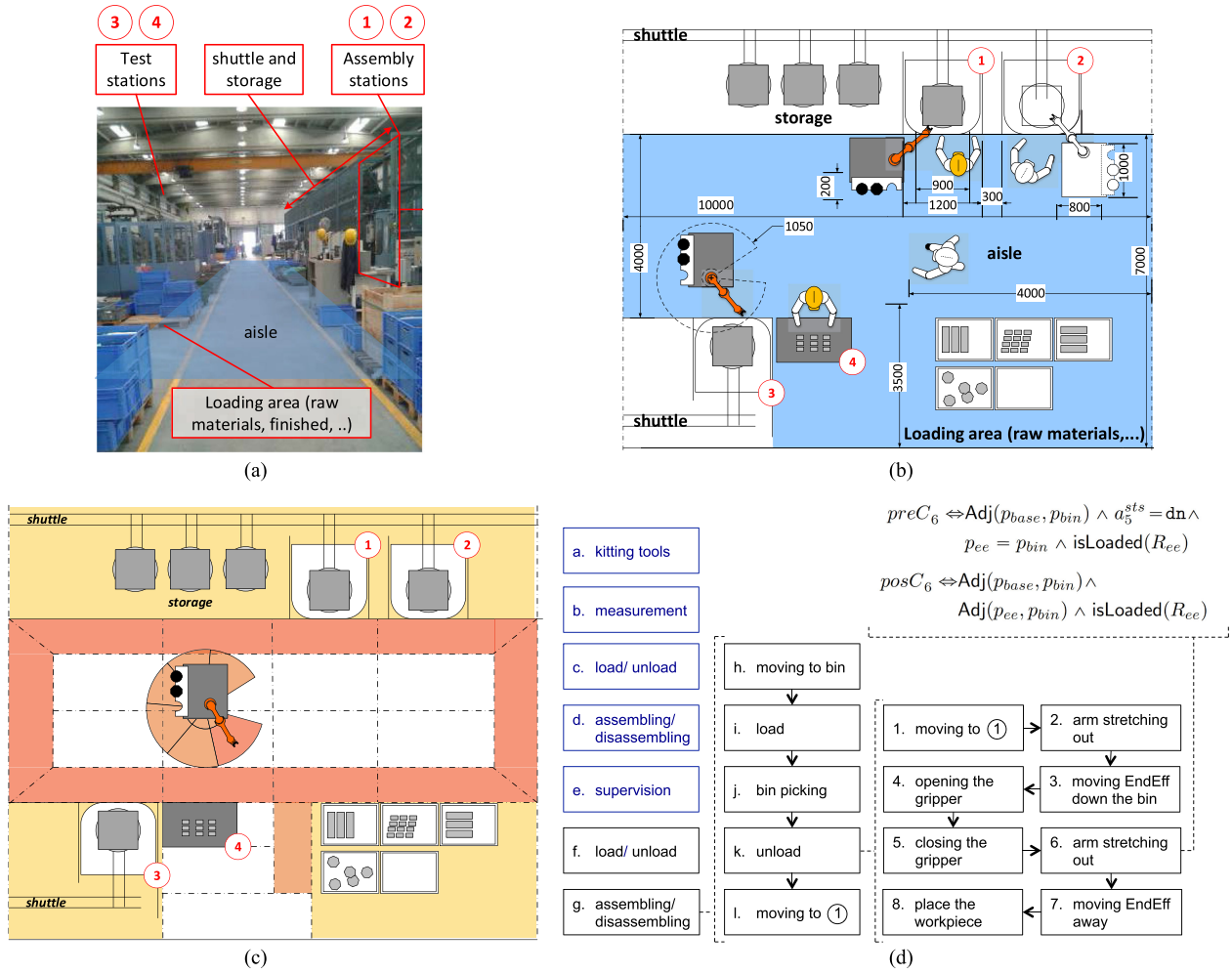


Fig. 11. (a) The case study is carried on a real workplace, where two assembly stations and one test station are operational. The mobile robot is intended to (un)load workpieces and move across the aisle (blue area). (b) Abstract sketch of the workspace with all the sizes and possible positions of the robot and operators. (c) Discretized representation of the layout according to the characteristics of locations. $\mathcal{L}$ areas are colored according to their *obst* value: `occluded` (red), `clear` (orange), `free` (yellow). White areas do not have a static *obst* value, but *obst* attribute can assume `occluded`, `clear`, or `free` values depending on the robot movements. The radial areas around the robotic arm actually turn around with the robot and, at any time, define the *obst* of the areas they coincide with. (d)The case study application is composed of several tasks done by operator (a)–(e) and robot (f)–(g). The predicate $\mathsf{isLoaded}(R_{ee})$ means that the gripper end-effector is loaded with a workpiece. Only the detail of task g is shown as an example. It is broken down into subtasks (h)–(l). Then, each of them in turn is a sequence of atomic actions. The actions of subtask k are shown, with their intended order. The formulae shown at the top right of the figure are an example of pre/postconditions of an action (in particular, of action 6).

of tasks and their actions are described with informal language, and a sample of detailed formulae corresponding to conditions of individual actions is shown in Fig. 11(d).

As stated in Definition 8, the smallest executable units of each task or application are atomic actions. These actions are derived from a finite set of activities (e.g., move, grab), which are executed in different regions or on different parts. The ultimate size of a task depends on the number of atomic actions required to execute it, regardless of further grouping into subtasks. Fig. 11(d) shows that the application is nominally divided in operator [i.e., (a)–(e)] and robot tasks [i.e., (f)–(g), but different combinations of human and robot tasks can achieve the same effects. For example, executing task "g," which is done by the robot, is equivalent to executing tasks "a" and "d" by the operator. Some tasks can also be switched between the operator and the robot.

Table V reports a significant subset of the formal model of the studied application. It provides constraints regarding layout regions, robot and human body part kinematics, and their discretized velocity values, a few examples of pre/postconditions of actions and discretized pressure values. Finally, the table briefly presents an example of the formal verification procedure and the RRMs that could be introduced to mitigate the risk.

B. Model Verification and Risk Reduction

As a result of a formal verification round, several samples of the SAFER-HRC output (graphical and textual traces) are shown in Fig. 12. In terms hazard identification, those corresponding to time steps 17 and 24 are examples of hazards due to intended uses, while the other two show hazards caused by unintended or erroneous behavior of the operator. Time step 8 reports the

TABLE V
SELECTED FORMULAE OF THE MODELED CASE STUDY IN FIG. 11

Model Elements	Formulae	Description
$\mathcal{L}$	$\forall O_i \in \{$ head, chest, leg, arm, fingers$\} : [$ $\exists L_k \in \mathcal{L} : (p_i = L_k)$ $\wedge \ \forall L_k \in \mathcal{L} : (p_i = L_k \Leftrightarrow \forall_{k' \neq k} L_{k'} \in \mathcal{L} : (p_i \neq L_{k'}))]$	Every element always occupies a location. Every element occupies only one location.
$\mathcal{R}$	$\forall R_j \in \{R_1, R_2, R_{ee}, R_{base}\} : [\exists L_k \in \mathcal{L} : [$ $v_j = \texttt{none} \Leftrightarrow \mathsf{Lasted}\,(p_j = L_k, 2)$ $\wedge \ v_j = \texttt{low} \Leftrightarrow \mathsf{Lasted}\,(p_j = L_k, 1) \wedge \mathsf{past}\,(p_j, 2) \neq L_k$ $\wedge \ v_j = \texttt{mid} \Leftrightarrow \mathsf{past}\,(p_j, 1) \neq p_j$ $\wedge \ v_j = \texttt{high} \Leftrightarrow \mathsf{past}\,(p_j, 2) \neq \mathsf{past}\,(p_j, 1) \neq p_j$ $\wedge \ \mathsf{moving}_{base} \Rightarrow \mathsf{moving}_{R_j}$ $\wedge \ \mathsf{past}\,(p_j, 1) = p_j \vee \mathsf{Adj}(\mathsf{past}\,(p_j, 1), p_j)]]$ $p_{ee} = p_2 \vee \mathsf{Adj}(p_{ee}, p_2)$	The velocity of each robot part is: `none`, if it has not displaced for the last 2 time units. `low` , if it has displaced the last time unit. `mid`, if it has moved. `high`, if it has displaced for two time units. If R_{base} displaces, so does every part mounted on it. The robot parts move smoothly inside the layout. R_{ee} is attached to the end of R_2, so are their positions.
$\mathcal{O}_{Alice}$	$\mathsf{Sep}_{fingers,arm} \in \{\texttt{close}, \texttt{mid}\}$ $p_{leg} < 24$ $\forall O_i \in \mathcal{O}_{Alice} : \mathsf{past}\,(p_i, 1) = p_i \vee \mathsf{Adj}(\mathsf{past}\,(p_i, 1), p_i)$	Attached body areas should always be close or adjacent. Alice's legs are always in bottom locations ($L_1 ... L_{23}$). $\mathcal{O}_{Alice}$'s body parts move smoothly inside the layout.
$\mathcal{O}_{Bob}$	The formulae for $\mathcal{O}_{Alice}$ are replicated for $\mathcal{O}_{Bob}$.	
Example of Actions	a_1 : $preC_1 \Leftrightarrow p_{base} = \text{initial\_point} \wedge \forall x_{\neq 1} : a_x^{sts} = \texttt{ns}$ $posC_1 \Leftrightarrow p_{base} = p_{bin}$	**Action 1.** R_{base} moves to bin from an initial point. **Pre-condition**:R_{base} is at initial point, other actions are `ns`. **Post-condition**: R_{base} is arrived at the front of the bin.
	a_2 : $preC_2 \Leftrightarrow a_1^{sts} = \texttt{dn} \wedge \neg\mathsf{moving}_{base} \wedge \neg\mathsf{moving}_{ee}$ $posC_2 \Leftrightarrow \neg\mathsf{moving}_{base} \wedge p_{ee} = \text{bin}$ $a_2^{sts} = \texttt{exe}\|\texttt{sfex} \Rightarrow \neg\mathsf{moving}_{base}$	**Action 2.** Robot arm is stretching out towards bin. **Pre-condition**: a_2 is done (i.e., base is in place and still). **Post-condition**: the robot is still and R_{ee} is in the bin. **Extra-condition**: R_{base} is still while $a_2^{sts} = \texttt{exe}$.
	a_{23} : $preC_{23} \Rightarrow p_{fingers_A} = p_{ee} = \textcircled{4} \wedge \mathsf{isLoaded}(R_{ee})$ $posC_{23} \Rightarrow p_{ee} = \textcircled{4} \wedge \mathsf{isLoaded}(R_{ee})$	**Action 23.** Alice inspects the workpiece (wp) at $\textcircled{4}$. **Pre-condition**: Alice and R_{ee} are at $\textcircled{4}$, R_{ee} holds wp. **Post-condition**: wp is still held by the gripper.
Pressure values (used in Risk Est--imation in addition to formulae (10) and (12))	$\forall O_i \in \mathcal{O},\ R_j \in \mathcal{R},\ L_k \in \mathcal{L} : [$ $\wedge \ P_{ij}^{\texttt{Tr}} = \texttt{mid} \Leftrightarrow (\mathsf{m}' = \texttt{low} \vee v_{ij} = \texttt{low}) \wedge \neg(\mathsf{m}' = \texttt{low} \wedge v_{ij} = \texttt{low})$ $\wedge \ P_{ij}^{\texttt{Tr}} = \texttt{high} \Leftrightarrow P_{ij}^{\texttt{Tr}} \neq (\texttt{low}\|\texttt{mid})$ $\wedge \ P_{ijk}^{\texttt{Qs}} = \texttt{mid} \Leftrightarrow (f_j = \texttt{high} \wedge (minshape_{ijk} = (\texttt{round}\|\texttt{fillet}))) \vee (f_j = \texttt{mid} \wedge (minshape_{ijk} = (\texttt{chamfer}\|\texttt{sharp})))$ $\wedge \ P_{ijk}^{\texttt{Qs}} = \texttt{high} \Leftrightarrow f_j = \texttt{high} \wedge (minshape_{ijk} = (\texttt{chamfer}\|\texttt{sharp}))]$	
Verification Example	Consider the following state of the model highlighted by the formal verification procedure: R_{ee} releases the wp (i.e., a_{24}), while Alice is inspecting it manually with fingers (i.e., $a_{23}^{sts} = \texttt{exe}$ holds). A quasi-static hazard is detected because R_{ee} endangers fingers of Alice ($fingers_A$) in an `occluded` area. Hence, . $\mathsf{hzd}^{\texttt{Qs}}_{fingers_A, ee, \textcircled{4}}$ holds, as defined by formula (8), where $i = fingers_A, j = ee$ and $k = \textcircled{4}$ Now the risk value of the hazard is estimated, which depends on: - the pressure value, which is `high` because R_{ee} is sharp and its force is high (i.e., $P^{\texttt{Qs}}_{fingers_A, ee, \textcircled{4}} = \texttt{high}$ as defined in the above row); - the sensitivity of the body part, which is also high. Hence, $\mathsf{risk}_{fingers_A, ee, \textcircled{4}} = 2$ holds and an RRM shall be introduced, alternatively selected between the following: - setting pressure low by $\mathsf{RRM}^{\mathtt{PFL}f}_{fingers_A, ee, \textcircled{4}}$ (formula (16)) which, by formula (10) and shape of R_{ee}, means $f_{ee} = \texttt{low}$; - modifying $posC_{23}$ and $preC_{24}$ so that a_{23} does not terminate if R_{ee} and fingers are close ($preC_{24} \Rightarrow a_{23}^{sts} = \texttt{dn}$), and R_{ee} does not release the wp before the inspection terminates by introducing a button to be pushed when the inspection is completed ($posC_{23} \Rightarrow p_{fingers} \neq p_{ee} \wedge \mathsf{buttonPushed}$). The next iteration of the formal verification procedure shows that the risk associated with this hazard is reduced by either of the two proposed RRMs.	

The source code of the full formal model can be found at https:github.com/Askarpour/SAFER-HRCgithub.com/Askarpour/SAFER-HRC. In the formulae, function $\mathrm{past}v, d$—all lowercase, unlike temporal operator $\mathrm{Past}\phi, d$—returns the value that variable v had d time units in the past.

occurrence of a `Tr` hazard on operator's head when he/she mistakenly bends down to execute an inspection task. Time step 20 depicts a hazardous situation resulting from a wrong manual execution (e.g., due to not following assigned instructions). For instance, the operator might confuse her/his duties and mistakenly starts to execute the task, which has been assigned to the robot. Since the task is to pick pieces from the local bin, he/she is exposed to impacts.

Table VI provides a numerical report of the performance of the SAFER-HRC procedure applied to the use case, which can be classified as "regular" in terms of difficulty of analysis from a safety technology and risk assessment expertise standpoint, and "challenging" in terms of extension and size of situations to consider. Unlike the general practice of risk assessment, SAFER-HRC reports *all* occurrences of hazards, including repeated/similar situations along a history: 60 instances of 27 individual hazards are found, combining the 2 base types of contact hazards (`Qs` and `Tr` impacts) along the tasks in the different required locations. In addition to being more complete than an averagely accurate risk assessment, the precise identification of hazard instances would allow a control system to trigger RRMs in due situations, at the right time.

Another benefit of the SAFER-HRC method is that a relatively large number of `Tr` hazards is identified (73% of total hazards):

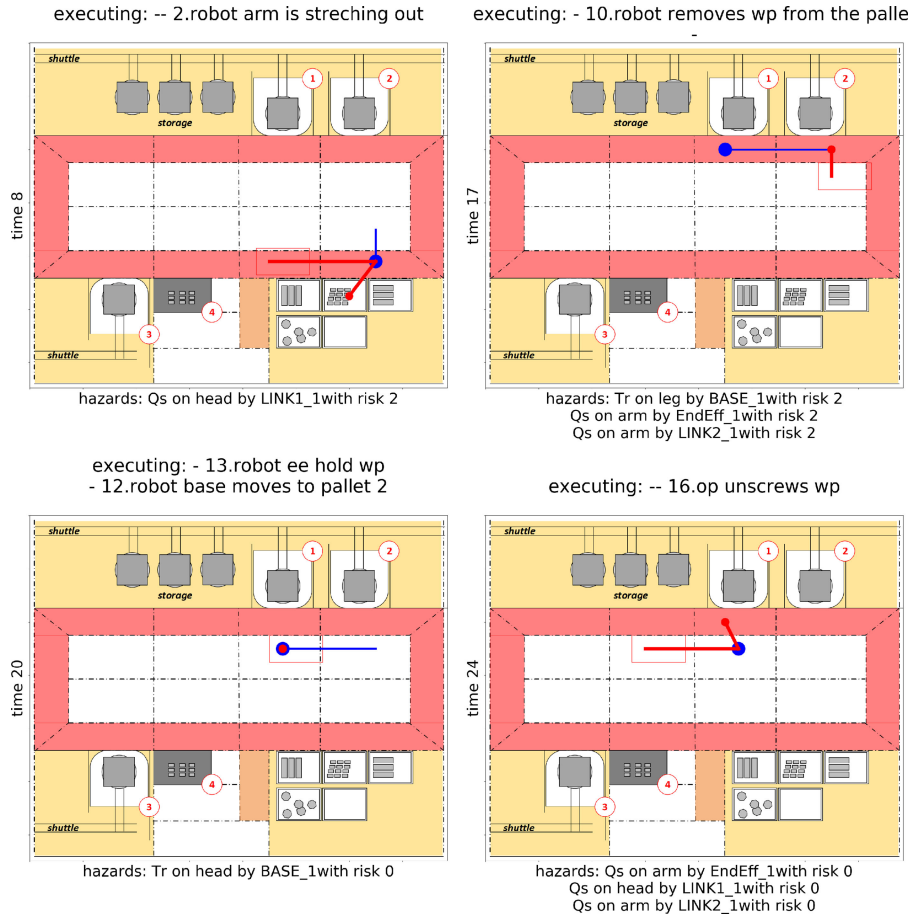


Fig. 12. Examples of outputs of the tool: snapshots of different time instants.

TABLE VI
POSTEVALUATION ANALYSIS OF ROUNDS OF MODEL CHECKING AND MODEL UPDATE: MAIN PERFORMANCE FIGURES IN THE *HAZARD IDENTIFICATION* GROUP REFER TO THE FIRST ROUND (NO RRMS), WHILE FIGURE ABOUT RRMS IN *RISK REDUCTION* GROUP REFER TO THE APPLICATION OF MODEL UPDATES

Hazard Identification	# hazards total (individual)	60 (27)
	# hazard types	2
	# Qs hazards	16 (27%)
	# Tr hazards	44 (73%)
	multiple instances	yes
	# locations	69
	# exposed body parts	5
	support unintended use	yes
	rate of unintended use instances	45%
Risk Reduction	# RRM types	5
	# hazard/RRM combinations	20
Total required time	20 man-hrs	

crushing and entanglement hazards (Qs) are somehow easier to track in a HRC application because they are associated with physical locations used during task execution. Accidental dynamic impacts may instead happen all along robot movements, in any region. Such high numbers in hazard finding are due to the ability of exploring many individual locations (69) to be combined with the five segments used in discretizing the operators' body. Since the method encompasses also unintended behaviors, many identified hazards are derived from misuses or errors, which are an essential part of accurate risk assessments in HRC. This is highlighted by the relatively high rate of detected unintended uses (45%).

Risk reduction is achieved by manually selecting (see liability of human assessor in Section VI-E) one of five possible RRMs, suitable to solve up to 20 different subcases of hazardous situations.

The total running time for all rounds of model verification is in the order of minutes/hours using commodity hardware,[7] to which we estimate to add several other days of model development—which might be anyway largely reused in similar applications. As a result, SAFER-HRC requires less effort than what is in general[8] necessary for completing a thorough safety assessment.

VIII. SUMMARY AND DISCUSSION

The methodology introduced in this article incorporated, automated, and improved existing standardized procedures for

[7]The experiments were done on a Linux desktop machine with a 3.4 GHz Intel Core i7-4770 CPU and 16 GB RAM of which only 235 MB have been used in the worst case. The average experiment time for each iteration was 19 min and 37 s.

[8]According to the experience of some of the authors in performing risk assessments in HRC.

the safety assessment of HRC applications by using automated formal verification techniques. In particular, the methodology relied on the TRIO temporal logic for system modeling, and on the Zot formal verification tool to iteratively identify hazardous situations associated with nonnegligible risks, and to mitigate them by introducing RRMs in the model.

The methodology improved manual risk assessments in terms of exhaustiveness, fallibility, and accuracy, while keeping a human-in-the-loop [73] standpoint. The last feature tagged this methodology as semiautomated due to the involvement of a human safety analyzer in the process of model refinement (i.e., explicitly introducing or acknowledging risk reduction models).

A. Discussion

This section examines in some depth the obtained results from the point of view of two relevant issues; then it addresses another important one, which has been left for future research.

1) Granularity and Scalability of the Model: The precision of the model is crucial for a robust identification of hazards. It might seem that the discretization of the model parameters, which is necessary for handling the size and complexity of the model, jeopardizes its precision. However, the discretization is done according to the functional specifications and some knowledge of the task (see, e.g., the examples of levels of speed for velocity discretization in Section VI-D, depending on the size and payload of robots). Other notable examples of the effects of discretization concern the partitioning of layout locations, which are identified—and sized—by their application-dependent features. The discretization is designed to prevent the possibility of overlooking relevant situations (e.g., instantaneous swapping of presence in adjacent regions) and consequently missing some hazards. In Example 1, the layout region L_1 is large enough to capture the proximity of the robot and the operator—i.e., detecting the contact situations—when they move in opposite directions. This approach is generalized to any situation where the model discretization involves some critical effects [52]. However, a finer granularity increases the model complexity and verification time. Issues such as not only granularity, but also the number of layout locations, the number of elements of the robot system, the size of the task (number of actions and number of possible execution variations) can greatly increase the complexity of the model. As the model gets bigger, the formal verification tool needs more time to exhaustively explore the corresponding state-space—leading to the well-known state-space explosion issue of formal verification approaches. Finding the right level of detail for models representing systems is an open issue, which inevitably involves using some level of abstraction [74].

The main consequence of increasing the complexity of the model—hence, verification times, which can become hours—is the inability of executing the verification online, while the system is running. However, our experiments are based on realistic case studies, on which verification times were acceptable. Hence, we claim that the approach is effective also from the scalability point of view.

On the other hand, since this article mainly focuses on design time, the issue remains one of "trading off precision for exhaustiveness." For example, one could prefer to use a *precise simulator* that gradually generates multiple small and short traces of a system, rather than to exhaustively explore all the possible traces of a simplified *abstract system*. Simulation tools for HRC applications (e.g., Tecnomatrix, Siemens PLM, Siemens, Germany) usually provide a physical engine for the representation of human and environment. Physical simulators are not alternative to formal verification. Both techniques have benefits and drawbacks already known to the scientific community and could be used as complementary approaches.

2) Functional Safety: The presented approach involves the use of RRMs in a purely abstract definition—i.e., as any sort of hazard-specific safety functions whose effect is to "solve" nonnegligible risks (reduce the risk score). This is done for decoupling the functional safety requirements on RRMs and the actual implementation of the safety functions that underpin RRMs. This approach has benefits in terms of formal verification, without affecting the consistency of the results obtained. For instance, the pain-pressure reduction RRM formula (10) is based on a safety function for force limitation, whose action is simply captured in the model by the condition $f_j = \texttt{low}$. Such function (i.e., formal predicate) is nonetheless implemented very differently—e.g., by tuning the maximum generated joint torque in commercial controllers, or by limiting the actuators power, or by inherently safe design of joints (e.g., through elements of passive compliance, like SEAs). Any implementation has the effect of lowering the Se factor of the risk estimation (i.e., limit the consequences of the harm). As part of a safety function, the reliability of the risk reduction is reported by the improved probability factor, e.g., $Pr < 2$ in risk estimation. A failure in the safety function is then excluded, i.e., f_j cannot assume $\{\texttt{mid}, \texttt{high}\}$ values in the safety function predicate. When no RRM is applied, instead, failures of the system or of control functions can still be taken into account by altering the properties of the model to be verified. In the current version of the model, no requirements of functional safety are specified—e.g., no required performance level PL_r as for ISO 13849 is assigned, and it is assumed that any RRM is implemented in the real system with the correct and suitable $\text{PL} = \text{PL}_r$. The verification of the functional safety requirements of RRMs would be nested inside the described effects of selection and application of RRMs: assigning a PL_r would be a matter of defining the exact probabilities of occurrence (derived from initial value of Pr) and gravity (derived from initial values of Se) of potential failures of the RRM. Verifying the implemented PL is a matter of computing the failure rate of the actual devices composing the safety function, together with the architecture and the diagnostic coverage of such function. This computation is nested because it would produce the evidence that the safety function is in fact able to improve the risk estimation factors, as captured by the formal model.

3) Stochastic Modeling: Risk estimation in HRC applications is affected by probabilistic distributions of unintended human behaviors or errors, which cumulate with distributions about failure rates or misfunctions of devices and concurrent failures/errors. In this article, we abstracted away from probabilistic analysis in the sense that we adopted some discrete values,

at some level of granularity, as proposed by hybrid methods [50], estimated *a priori* on the basis of previous experience and common sense. As pointed out in previous item Section VIII-A2, the Pr factor for failsafe components or functions, if deemed necessary as part of RRMs, is transformed from "high" to "low" (instead of being a numerical expression of, e.g., the ISO 13849-1 parameter PFH_d of a safety function). Associating probability distributions with detected hazard occurrences, and specifically human errors, would make the risk estimate *more accurate*, but it would not necessarily fundamentally alter the proposed approach. Indeed, probabilities could be associated with hazards after they are detected by the formal verification runs, to better estimate the actual need to introduce RRMs. In particular, we considered in [62] a series of common human errors derived from previous analyses based on psychological operator models [75]. We consider deterministic facts resulting from errors (e.g., being late, being in the wrong location, doing the wrong sequence, and not respecting a postcondition). Introducing a continuous probability of such operator's errors could be the result of a suitable formalization of a parameter associated with erroneous conditions. Considering the present literature on modeling human behavior in HRC applications (e.g., the probability of repeating the same error twice, the probabilities of errors for tired, unexperienced operators with respect to expert, concentrated operators), a stochastic model of errors could help in assigning more accurate Pr factors to error appearances, but would not be formalized as a further cause-effect relationship accounting for cognitive phenomena of any type. An objective of ongoing research is to embed complementary stochastic modeling for error parametrization into our approach based on deterministic formal verification.

REFERENCES

[1] *Safety of Machinery—General Principles for Design—Risk Assessment and Risk Reduction*, ISO 12100, International Organization for Standardization, Geneva, Switzerland, 2010.

[2] P. A. Lasota, T. Fong, and J. A. Shah, "A survey of methods for safe human-robot interaction," *Found. Trends Robot.*, vol. 5, no. 4, pp. 261–349, 2017.

[3] *Analysis Techniques for System Reliability—Procedure for Failure Mode and Effects Analysis (FMEA)*. IEC 60812, International Electrotechnical Commission, Geneva, Switzerland, 2006.

[4] *Fault Tree Analysis (FTA)*. IEC 61025, International Electrotechnical Commission, Geneva, Switzerland, 2006.

[5] *Hazard and Operability Studies (HAZOP studies)—Appl.ication Guides*. IEC 61882, International Electrotechnical Commission, Geneva, Switzerland, 2016.

[6] N. Leveson, "A new accident model for engineering safer systems," *Saf. Sci.*, vol. 42, no. 4, pp. 237–270, 2004.

[7] N. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA, USA: MIT Press, 2011.

[8] J. Guiochet, D. Martin-Guillerez, and D. Powell, "Experience with model-based user-centered risk assessment for service robots," in *Proc. IEEE 12th Int. Symp. High Assurance Syst. Eng.*, Nov. 2010, pp. 104–113.

[9] B. S. Dhillon and A. R. M. Fashandi, "Safety and reliability assessment techniques in robotics," *Robotica*, vol. 15, no. 6, pp. 701–708, 1997.

[10] G. Joshi and H. Joshi, "FMEA and alternatives v/s enhanced risk assessment mechanism," *Int. J. Comput. Appl.*, vol. 93, pp. 33–37, 2014.

[11] A. A. Baig, R. Ruzli, and A. B.Buang, "Reliability analysis using fault tree analysis: A review," *Int. J. Chem. Eng. Appl.*, pp. 169–173, 2013.

[12] J. M. Wing, "A specifier's introduction to formal methods," *Computer*, vol. 23, no. 9, pp. 8–23, Sep. 1990.

[13] J. Guiochet, "Hazard analysis of human-robot interactions with HAZOP-UML," *Saf. Sci.*, vol. 84, pp. 225–237, 2016.

[14] OMG-UML2, *OMG Unified Modeling Language(OMG UML), Superstructure, V2.1.2*, 2007.

[15] M. Webster *et al.*, "Formal verification of an autonomous personal robotic assistant," in *Proc. Formal Verification Model. Human-Mach. Syst.: Papers AAAI Spring Symp.*, 2014, pp. 74–79.

[16] M. Webster *et al.*, "Toward reliable autonomous robotic assistants through formal verification: A case study," *IEEE Trans. Human-Mach. Syst.*, vol. 46, no. 2, pp. 186–196, Apr. 2016.

[17] M. Sierhuis, W. J. Clancey, and R. J. V. Hoof, "Brahms: A multi-agent modelling environment for simulating work processes and practices," *Int. J. Simul. Process Modelling*, vol. 3, no. 3, pp. 134–152, 2007.

[18] SPIN model checker. 2002. [Online]. Available: http://spinroot.com/spin/whatispin.htmlspinroot.com

[19] C. Dixon, M. Webster, J. Saunders, M. Fisher, and K. Dautenhahn, ""The fridge door is open"—temporal verification of a robotic assistant's behaviours," in *Proc. 15th Annu. Conf. Adv. Auton. Robot. Syst.*, 2014, pp. 97–108.

[20] R. Stocker, L. Dennis, C. Dixon, and M. Fisher, "Verifying brahms human-robot teamwork models," in *Proc. Eur. Conf. Logics Artif. Intell.*, 2012, pp. 385–397.

[21] J. Bredereke and A. Lankenau, "Safety-relevant mode confusions modelling and reducing them," *Rel. Eng. Syst. Saf.*, vol. 88, no. 3, pp. 229–245, 2005.

[22] R. Butterworth, A. Blandford, and D. Duke, "Demonstrating the cognitive plausibility of interactive system specifications," *Formal Aspects Comput.*, vol. 12, no. 4, pp. 237–259, 2000.

[23] J. Fu and U. Topcu, "Synthesis of shared autonomy policies with temporal logic specifications," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 1, pp. 7–17, Jan. 2016.

[24] *Robots and Robotic Devices—Safety Requirements for Industrial Robots—Part 2: Robot Systems and Integration*. ISO 10218-2, International Organization for Standardization, Geneva, Switzerland, 2011.

[25] *Robots and Robotic Devices—Collaborative Robots*. ISO/TS 15066, International Organization for Standardization, Geneva, Switzerland, 2016.

[26] ISO 13482, *Robots and Robotic Devices Safety Requirements for Personal Care Robots*. ISO 13482, International Organization for Standardization, Geneva, Switzerland, 2014.

[27] N. A. Stanton, "Hierarchical task analysis: Developments, applications, and extensions," *Appl. Ergonom.*, vol. 37, no. 1, pp. 55–79, 2006.

[28] J. A. Marvel, J. Falco, and I. Marstio, "Characterizing task-based human-robot collaboration safety in manufacturing," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 2, pp. 260–275, Feb. 2015.

[29] G. Jamieson, J. Andersson, A. Bisantz, A. Degani, and M. Lind, "Model-based approaches to human-automation systems design," in *Proc. ASME Biennial Conf. Eng. Syst. Des. Anal.*, 2012, pp. 871–880.

[30] K. W. Wong, R. Ehlers, and H. Kress-Gazit, "Correct high-level robot behavior in environments with unexpected events," *Robot., Sci. Syst. X*, 2014.

[31] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, "Generating phenotypical erroneous human behavior to evaluate human-automation interaction using model checking," *Int. J. Human-Comput. Stud.*, vol. 70, no. 11, pp. 888–906, Nov. 2012.

[32] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, "Using formal verification to evaluate human-automation interaction: A review," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 3, pp. 488–503, May 2013.

[33] M. L. Bolton and E. J. Bass, "Formally verifying human–automation interaction as part of a system model: Limitations and tradeoffs," *Innov. Syst. Softw. Eng.*, vol. 6, no. 3, pp. 219–231, 2010.

[34] J. E. Laird, *The Soar Cognitive Architecture*. Cambridge, MA, USA: MIT Press, 2012.

[35] J. R. Anderson, "Act: A simple theory of complex cognition," *Amer. Psychologist*, vol. 51, pp. 355–365, 1996.

[36] P. A. Lindsay and S. Connelly, "Modelling erroneous operator behaviours for an air-traffic control task," in *Proc. Australas. UI Conf.*, 2002, pp. 43–54.

[37] R. M. Young, T. R. G. Green, and T. J. Simon, "Programmable user models for predictive evaluation of interface designs," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 1989, pp. 15–19.

[38] P. Curzon, R. Rukšėnas, and A. Blandford, "An approach to formal verification of human–computer interaction," *Formal Aspects Comput.*, vol. 19, no. 4, pp. 513–550, 2007.

[39] B. Werther and E. Schnieder, "Formal cognitive resource model: Modeling of human behavior in complex work environments," in *Proc. Int. Conf. Comput. Intell. Modelling, Control Autom., Int. Conf. Intell. Agents, Web Technol. Internet Commerce*, 2005, pp. 606–611.

[40] J. Bredereke and A. Lankenau, "A rigorous view of mode confusion," in *Proc. Int. Conf. Comput. Saf., Rel. Secur.*, 2002, pp. 19–31.

[41] W. D. Gray, "The nature and processing of errors in interactive behavior," *Cogn. Sci.*, vol. 24, no. 2, pp. 205–248, 2000.
[42] F. E. Ritter and R. M. Young, "Embodied models as simulated users: Introduction to this special issue on using cognitive models to improve interface design," *Int. J. Human-Comput. Stud.*, vol. 55, no. 1, pp. 1–14, 2001.
[43] C. M. Mitchell and R. A. Miller, "A discrete control model of operator function: A methodology for information display design," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, no. 3, pp. 343–357, May 1986.
[44] H. R. Hartson, A. C. Siochi, and D. Hix, "The UAN: A user-oriented representation for direct manipulation interface designs," *ACM Trans. Inf. Syst.*, vol. 8, no. 3, pp. 181–203, Jul. 1990.
[45] F. Paternò, "Formal reasoning about dialogue properties with automatic support," *Interacting Comput.*, vol. 9, no. 2, pp. 173–196, 1997.
[46] A. J. Abbate, A. L. Throckmorton, and E. J. Bass, "A formal task-analytic approach to medical device alarm troubleshooting instructions," *IEEE Trans. Human-Mach. Syst.*, vol. 46, no. 1, pp. 53–65, Feb. 2016.
[47] S. Junges, N. Jansen, J. Katoen, and U. Topcu, "Probabilistic model checking for complex cognitive tasks—A case study in human-robot interaction," *CoRR*, vol. abs/1610.09409, 2016.
[48] R. Ruksenas, J. Back, P. Curzon, and A. Blandford, "Verification-guided modelling of salience and cognitive load," *Formal Aspects Comput.*, vol. 21, no. 6, pp. 541–569, 2009.
[49] M. Askarpour, "SAFER-HRC: A methodology for safety assessment through formal verification in human-robot collaboration," Doctoral Dissertation, DEIB, Polytechnic Univ. Milan, Milan, Italy, 2018.
[50] *Safety of Machinery—Risk Assessment—Part 2: Practical Guidance and Examples of Methods*. ISO/TR 14121-2, International Organization for Standardization, Geneva, Switzerland, 2012.
[51] R. Pelánek, "Fighting state space explosion: Review and evaluation," in *Proc. Int. Workshop Formal Methods Ind. Crit. Syst.*., 2008, pp. 37–52.
[52] C. A. Furia, D. Mandrioli, A. Morzenti, and M. Rossi, *Modeling Time in Computing* (Series Monographs in Theoretical Computer Science. An EATCS Series). New York, NY, USA: Springer, 2012.
[53] "Zot: A bounded satisfiability checker," 2012. [Online]. Available: http://github.com/fm-polimi/zotgithub.com/fm-polimi/zot
[54] A. Cimatti *et al.*, "NuSMV 2: An opensource tool for symbolic model checking," in *Proc. 14th Int. Conf. Comput. Aided Verification*, 2002, pp. 359–364.
[55] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Syst.*, vol. 2, no. 4, pp. 255–299, 1990.
[56] M. Pradella, A. Morzenti, and P. San Pietro, "Bounded satisfiability checking of metric temporal logic specifications," *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 3, 2013, Art. no. 20.
[57] E. Ciapessoni, P. Mirandola, A. Coen-Porisini, D. Mandrioli, and A. Morzenti, "From formal models to formally based methods: An industrial experience," *ACM Trans. Softw. Eng. Methodol.*, vol. 8, no. 1, pp. 79–113, 1999.
[58] L. Baresi, M. M. P. Kallehbasti, and M. Rossi, "Efficient scalable verification of LTL specifications," in *Proc. 37th Int. Conf. Softw. Eng.*, 2015, pp. 711–721.
[59] *Basic Human Body Measurements for Technological Design – Part 2: Statistical Summaries Body Measurements from National Populations*. ISO/TR 7250-2, International Organization for Standardization, Geneva, Switzerland, 2010/Amd1:2013.
[60] M. Melia *et al.*, "Measuring mechanical pain: The refinement and standardization of pressure pain threshold measurements," *Behav. Res. Methods*, vol. 47, no. 1, pp. 216–227, 2015.
[61] P. Curzon and A. Blandford, "From a formal user model to design rules," in *Proc. Int. Workshop Interactive Syst., Des., Specification, Verification*, 2002, pp. 1–15.
[62] M. Askarpour, D. Mandrioli, M. Rossi, and F. Vicentini, "Modeling operator behavior in the safety analysis of collaborative robotic applications," in *Proc. Int. Conf. Comput. Safety, Rel., Secur.*, 2017, pp. 89–104.
[63] *Safety of Machinery – Safety-Related Parts of Control Systems – Part 1: General Principles for Design*. ISO 13849-1, International Organization for Standardization, Geneva, Switzerland, 2006.
[64] S. Haddadin, "Physical safety in robotics," in *Proc. 1st Int. Summer School Methods Tools Des. Digit. Syst. Formal Model. Verification Cyber-Phys. Syst.*, 2015, pp. 249–271.
[65] Y. Yamada, Y. Hirasawa, S. Huang, Y. Umetani, and K. Suita, "Human-robot contact in the safeguarding space," *IEEE/ASME Trans. Mechatronics*, vol. 2, no. 4, pp. 230–236, Dec. 1997.
[66] D. Mewes and F. Mauser, "Safeguarding crushing points by limitation of forces," *Int. J. Occupational Saf. Ergonom.*, vol. 9, no. 2, pp. 177–191, 2003.
[67] S. Haddadin, A. Albu-Schffer, and G. Hirzinger, "Requirements for safe robots: Measurements, analysis and new insights," *Int. J. Robot. Res.*, vol. 28, no. 11-12, pp. 1507–1527, 2009.
[68] *Machinery Directive of the EC*, MD 2006/42/EC, International Organization for Standardization, Geneva, Switzerland, 2006.
[69] E. Carpanzano, L. Ferrucci, D. Mandrioli, M. Mazzolini, A. Morzenti, and M. Rossi, "Automated formal verification for flexible manufacturing systems," *J. Intell. Manuf.*, vol. 25, no. 5, pp. 1181–1195, 2014.
[70] M. Rossi, D. Mandrioli, A. Morzenti, and L. Ferrucci, "A temporal logic for micro- and macro-step-based real-time systems: Foundations and applications," *Theor. Comput. Sci.*, vol. 643, pp. 38–64, 2016.
[71] *Safety of Machinery—Positioning of Safeguards with Respect to the Approach Speeds of Parts of the Human Body*. ISO 13855, International Organization for Standardization, Geneva, Switzerland, 2010.
[72] F. Vicentini, M. Giussani, and L. M. Tosatti, "Trajectory-dependent safe distances in human-robot interaction," in *Proc. IEEE Emerg. Technol. Factory Autom.*, Sep. 2014, pp. 1–4.
[73] P. Fung, G. Norgate, T. Dilts, A. Jones, and R. Ravindran, "Human-in-the-loop machine control loop," U.S. Patent 5116180 A, Nov. 6, 2018.
[74] J. Guiochet, M. Machin, and H. Waeselynck, "Safety-critical advanced robots: A survey," *Robot. Auton. Syst.*, vol. 94, pp. 43–52, 2017.
[75] Siemens, "RobotExpert extension of tecnomatix process simulate suite," [Online]. Available: https://docs.plm.automation.siemens.com/tdoc/tecnomatix/14.1.2/RBX#uid%:indexSiemens/RobotExpert, Accessed on: Jan. 2018
[76] E. Hollnagel, *Cognitive Reliability and Error Analysis Method (CREAM)*. Amsterdam, The Netherlands: Elsevier, 1998.

Federico Vicentini received the M.Sc. degree in mechanical engineering and the Ph.D. degree in mechanical system engineering from the Politecnico di Milano, Milano, Italy, in 2003 and 2007, respectively.

He is currently a Researcher with the National Research Council (CNR), Institute of Intelligent Industrial Systems and Technologies for Advanced Manufacturing (STIIMA), Milan, Italy. His research interests include industrial robot safety, human–robot interaction, and validation procedures.

Mr. Vicentini serves as a member of national and international standardization committees for robot and machine safety.

Mehrnoosh Askarpour received the M.Sc. degree in computing systems engineering and the Ph.D. degree in information technology from the Politecnico di Milano, Milano, Italy, in 2013 and 2018, respectively.

She is currently a Postdoc Researcher with the Politecnico di Milano, Milan, Italy. Her current research interests include verification of safety-critical system properties, application of formal methods in industrial robotics, and model checking for safe collaborative robotics.

Matteo G. Rossi received the M.Sc. degree in computer science and automation engineering and the Ph.D. degree in computer engineering and automation from the Politecnico di Milano, Milano, Italy, in 1999 and 2003, respectively.

She is currently an Assistant Professor of Software Engineering and Formal Methods with the Politecnico di Milano, Milan, Italy. His research interests include formal methods for safety-critical and real-time systems, architectures for real-time distributed systems, and transportation systems both from the point of view of their design, and of their application in urban mobility scenarios.

Dino Mandrioli received the M.Sc. degree in electronic engineering from the Politecnico di Milano, Milano, Italy, in 1972.

He is currently a Full Professor of Computer Science with the Politecnico di Milano, Milan, Italy. His research interests include formal methods for critical systems, formal languages and their characterization in mathematical logic, automatic verification, and model checking.