SAFER-HRC: Safety Analysis through Formal vERification in Human-Robot Collaboration

Mehrnoosh Askarpour[1], Dino Mandrioli[1], Matteo Rossi[1], and Federico Vicentini[2]

[1] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milan, Italy
{mehrnoosh.askarpour,dino.mandrioli,matteo.rossi}@polimi.it
[2] Consiglio Nazionale delle Ricerche, Istituto di Tecnologie Industriali e Automazione, via Bassini 15, 20133 Milan, Italy
federico.vicentini@cnr.itia.it

Abstract. Whereas in classic robotic applications there is a clear segregation between robots and operators, novel robotic and cyber-physical systems have evolved in size and functionality to include the collaboration with human operators within common workspaces. This new application field, often referred to as Human-Robot Collaboration (HRC), raises new challenges to guarantee system safety, due to the presence of operators. We present an innovative methodology, called SAFER-HRC, centered around our logic language TRIO and the companion bounded satisfiability checker Zot, to assess the safety risks in an HRC application. The methodology starts from a generic modular model and customizes it for the target system; it then analyses hazards according to known standards, to study the safety of the collaborative environment.

Keywords: Safety Analysis, Formal Verification, Safety Rules, Human-Robot Collaboration.

1 Introduction

In Human-Robot Collaboration (HRC) applications, close proximity and direct interaction between robot and operator are unavoidable, so providing safety for the operator requires more effort and a more rigorous approach. Thus, the safety of machinery community has published several standards [11,15], which include a list of significant *hazards* in HRC applications, potential sources of harms for the operator, their likely origins, and safety regulations for guiding the design and deployment of robotic solutions. In particular, ISO standard 10218-2 [11] identifies four possible collaborative modes between humans and industrial robots. Of these, Power and Force Limitation (PFL) is the one involving actual physical contact, and it is associated with strict safety requirements in terms of pressure and force thresholds, in order to limit the effects on the human body.

Our focus in this work is on collaborative robots that are considered in the PFL category.

HRC applications must be evaluated through the analysis laid out in ISO standard 12100 [12], to identify existing hazards and unwanted situations due to intentional *misuses* or unconscious *errors* of the operator; and to prevent their consequences, which are measured in terms of quantified *risk* values.

Figure 1(a) shows the stages of a standard iterative risk analysis (resulting in the CE marking in the case of European directives).
(i) **Limits of Machinery**: The desired tasks of the robot and its machinery regulations and constraints are determined.
(ii) **Hazards Identification**: The existence of hazards (and combinations thereof) listed in product-specific standards such as ISO 10218-2 is identified.
(iii) **Risk Estimation**: The risk values associated with hazards identified in the previous step are measured. Many risk-scoring methods are introduced in [14], all of which combine the severity of a harm with its likelihood.
(iv) **Risk Evaluation**: The significance of each hazard is evaluated. The methods reported in [14] help determine the range of acceptability for the risk scores.
(v) **Risk Reduction**: If the risk value is not negligible, appropriate measures are (iteratively) introduced to reduce each risk, either by redesigning the system to eliminate the hazard , or through the introduction of a safety function (e.g. "full stop in case of a signal from a protection sensor"), which needs to be verified against suitable requirements of reliability and availability. We refer the reader to [13] for a complete discussion about functional safety.
The process continues iteratively until no new risk is identified and the residual risk value is acceptable. New risks may appear due to hazards related to risk reduction measures, or to operator behaviors in reaction to such measures. Devices and protection measures can alter the course of actions (use and misuse).

In this paper we introduce the SAFER-HRC (Safety Analysis through Formal vERification in HRC applications) methodology, which provides a technique to comprehensively identify hazards through the exhaustive exploration, rooted in formal methods, of the behavior of the target system. Among the different types of hazards (e.g., electrical, ergonomic, ...), this work addresses operational hazards with a specific focus on those that are caused by human-robot interactions. Although we do not claim that SAFER-HRC guarantees that all possible hazards in a system are found, we argue that the exhaustive exploration on which the methodology is based helps increase the confidence that no significant hazardous situations are left unconsidered. To achieve an exhaustive analysis of the system model, we rely on the state-space exploration capabilities of formal verification techniques. Due to the impossibility of foreseeing all possible behaviors of the operator, it cannot be claimed that all possible interactions of operator and system are taken into account; nevertheless, an iterative methodology based on formal verification techniques can eventually provide a thorough analysis of all significant ones. At each iteration, if the design fails to satisfy the desired safety requirements, it is improved by adding new risk reduction measures. This methodology relies on a"human-in-the-loop" approach [6] and it does

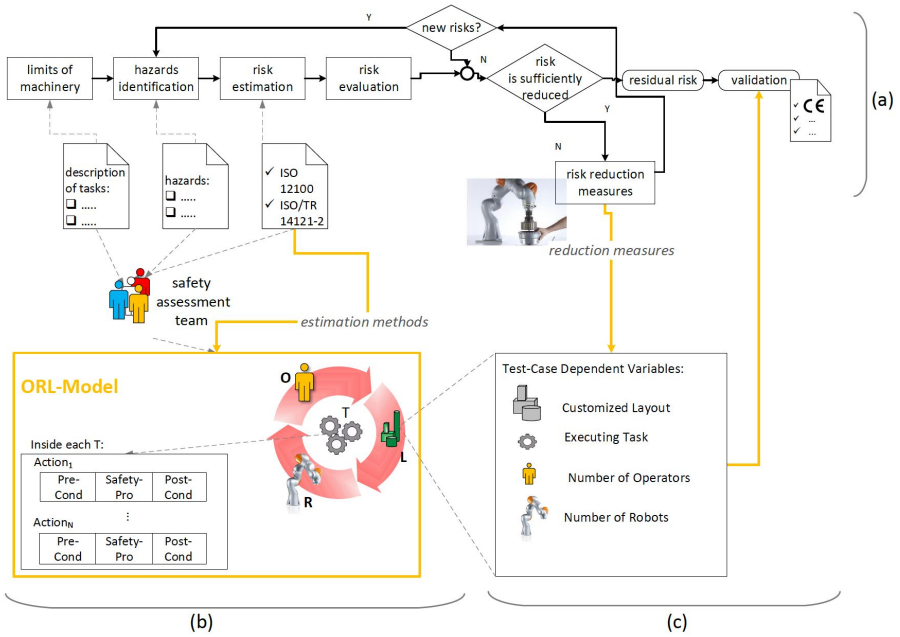

Fig. 1: Overview of the safety analysis methodology: (a) standard procedure; (b) principal model of SAFER-HRC; (c) scenario refinement.

not automatically select risk reduction measures. As shown in Figure 1(b), the safety strategy is designed and acknowledged by a pool of experts and users of the application under assessment (the safety assessment team). The essential aspect of the proposed methodology is the systematic validation of the constraints and their possible violations at all steps of the application. The thoroughness of the validation ensures that the selected safety strategy is failsafe. SAFER-HRC starts from informal, goal-oriented descriptions of collaborative tasks, and converts them into formal models built upon logical formulae, on which formal verification techniques are applied to check whether the safety requirements are satisfied or not. The model includes separate formalizations for operator and robot; hence, the verification phase also checks their interactions, taking into account how they are affected by the physical environment. After the principal model has been thoroughly analyzed, it can be modified and re-used to study different scenarios for the HRC application (e.g., combinations of different safety functions, uncommon actions by the operator).

The main contributions of the methodology presented in this paper are:

1. Applying formal methods to the safety assessment of HRC applications, where the presence of the operator negatively impacts on the predictability of the system behavior, but also imposes demanding safety standards that must be rigorously studied.

2. Providing a flexible approach that supports the safety assessment team in throughly exploring different design assumptions, thus complementing the human insight with the power of formal verification.

The paper is structured as follows: Section 2 discusses related works, and Section 3 gives a brief formal background. Section 4 introduces the essential aspects of the SAFER-HRC methodology. Section 5 illustrates our approach through an example of a collaborative assembling task. Section 6 concludes.

2 Related Works

Classic hazard identification approaches such as FTA and FMECA [4] are not well-suited for HRC applications, as they cannot deal with unpredictable human interactions with robots. We use formal verification methods as a means to improve hazard identification in robotic applications. These methods can be applied to complement informal techniques such as Hazop [10], which consists of a set of meetings and brainstorming sessions to identify and evaluate potential hazards concerning operators, equipment or efficiency. Its aim is to exploit as much information as possible from expert users and experienced safety engineers. Another informal hazard analysis technique is STPA [16], which builds a model of the control structure of the system to identify control-related flaws.

There are recent works tackling safety issues in robotic applications with human intervention using semi-formal solutions, or a combination of semi-formal and formal solutions. For example, in [9,19] State-charts are first used to describe the behavior of the robot, and then HAZOP is employed using UML models to identify potential hazards, their causes and their severity. In [17] hazards are identified by a combination of UML and HAZOP, then they are formalized in CTL (Computation Tree Logic). The same authors in a later work [18] compute a set of if-then-else safety constraints, and then add them to the logical model of the system to avoid predicted hazards. However, their application domain consists of assistive, rather than collaborative robots, and so the operator is a passive element whose on-the-fly decisions or errors are not considered as a determinative fact. A recent work [8] systematizes the pairing of HAZOP and UML, and presents results also for collaborative scenarios, excluding the formal point of view and focusing on an informal solution. As we aim at combining the two aspects, this approach could be used to define the first informal description of the system from which to derive the principal model discussed in Section 4.

In building our logical model, we use a contract-based approach similar to the one in [2]. Such an approach allows us to break down the overall task description into small components, to specify the requirements of each component separately, and to have a modular, clean formal description of collaborative tasks.

3 Preliminaries

Our approach is rooted in TRIO, a logical language which assumes an underlying linear temporal structure and features a quantitative notion of time [7].

Operator	Definition	Meaning
$\text{Futr}(\phi, d)$	$d > 0 \wedge \text{Dist}(\phi, d)$	ϕ occurs exactly at d time units in the future
$\text{Past}(\phi, d)$	$d > 0 \wedge \text{Dist}(\phi, -d)$	ϕ occurred exactly at d time units in the past
$\text{AlwF}(\phi)$	$\forall t(t > 0 \Rightarrow \text{Dist}(\phi, t))$	ϕ holds always in the future
$\text{Until}(\phi, \psi)$	$\exists t(\text{Futr}(\{\}\, \psi, t\} \wedge$ $\forall t'(0 < t' < t \Rightarrow \text{Dist}(\phi, t')))$	ψ will occur in the future and ϕ will hold until then
$\text{SomF}(\phi)$	$\exists t(t > 0 \wedge \text{Dist}(\phi, t))$	ϕ occurs sometimes in the future
$\text{SomP}(\phi)$	$\exists t(t > 0 \wedge \text{Dist}(\phi, -t))$	ϕ occurred sometimes in the past

Table 1: List of derived TRIO operators.

TRIO formulae are built out of the usual first-order connectives, operators, and quantifiers, as well as a single basic modal operator, called Dist, that relates the *current time*, which is left implicit in the formula, to another time instant: given a time-dependent formula ϕ (i.e., a term representing a mapping from the time domain to truth values) and a (arithmetic) term t indicating a time distance (either positive or negative), formula $\text{Dist}(\phi, t)$ specifies that ϕ holds at a time instant at a distance of exactly t time units from the current one.

While TRIO can exploit both discrete and dense sets as time domains, in this work we assume the standard model of the nonnegative integers $\mathbb{N}$ 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 1 defines some of the most significant ones, including those used in this work.

The satisfiability of TRIO formulae is in general undecidable. However, in this paper we consider a decidable subset of the language, that can be handled by automated tools, to build the system model and to express its properties. In particular, Zot [1] is a bounded satisfiability checker for TRIO formulae [20]. We use Zot in this work to check the model of the system against desired safety properties. In case the property is not satisfied, Zot provides a counterexample witnessing a system execution that violates the property.

4 Overview of the SAFER-HRC Methodology

This section introduces SAFER-HRC, a semi-automated verification methodology which benefits from formal verification techniques to extract the violation of safety requirements mentioned in ISO10218 [11] during the design of collaborative robotic systems. As depicted in Figure 1(b), at the core of SAFER-HRC lies a safety assessment team (SATeam). SATeam, which includes robotic and formal methods experts, studies the limitations of the machinery and the tasks of the target robot, and predicts possible human-robot interactions. They also determine which of the hazards listed in ISO 12100 can occur, and evaluate the risk level based techniques defined in ISO standard 14121 [14]. In SAFER-HRC, SATeam relies on a formal model of the HRC application to support and system-

atize these activities. More precisely, SATeam starts from the informal, textual definitions of the tasks, and then builds UML diagrams as a bridge towards the formal representation.

General O-R-L Model. The formal model captures the dynamics of the interactions occurring in the system in terms of the relationships among three main elements, O, R and L, which formally describe, respectively, operator, robot and layout through logic formulae. O is a formal model of the operator's body parts, each with critical safety requirements as described in standard ISO/TC 184/SC. R models the robot by describing the edges that have some degree of freedom in their movements; the nature of this model depends heavily on robot type and shape. O and R contain constraints to avoid considering unrealistic body shapes or robot structures (e.g., the head of the operator is in one corner of the workspace, while her hand is in the opposite one). L provides a representation of the layout of the system that allows us to describe the position of the physical features of O and R at any time instant. The O-R-L model contains some elements and constraints that are common to all HRC applications (e.g., the description of body parts); other parts of it (e.g., the features of the robot) are instead instantiated depending on the specific HRC application.

The O-R-L Model includes also a part related to the pool of tasks that the robot modeled in R is supposed to perform. Each task and its requirements and regulations are modeled in an element called T. The definition of each task T determines the type and frequency of interactions among O-R-L elements. The execution of a task involves a functional relationship between each pair of O-R-L elements. These relationships can be physical ones (e.g., contact between the robot arm and the operator, presence of the operator and robot in a common area in the layout) or informational (e.g., inputs given to the robot by the operator). For example, consider the following safety requirement: "operator's head should not be close to the robot end-effector while it is drilling". The model defines a value (L_{drill}) corresponding to the area in the layout where drilling is done, a variable (EF) capturing the position of the end-effector, and another one ($OpHead$) for the operator's head position; then, T contains the following constraint associated with the drilling task, stating that $OPHead$ cannot be in L_{drill} while drilling is executing: $Drilling_{\text{state}} = exe \Rightarrow \neg(OPHead = L_{drill}) \wedge (EF = L_{drill})$.

Usually the definition of a task has a goal-oriented view and contains multiple smaller units of execution. Breaking down a task into the smallest possible functional units, i.e., into *elementary actions*[5], enables SATeam to extract the previously mentioned relationships among O-R-L elements and also helps to identify the hazards that might otherwise be overlooked if one only stayed at a higher level of analysis. Another benefit of distinguishing single actions is that, in case it is possible to achieve the same goal with different sequences of actions, and the operator has the ability to decide on-the-fly what sequence to use, different sets of hazards caused by each sequence are identified. Further, the human body parts that are in contact with or close to the robot end-effector can differ for separate actions within a task, and this in turn can affect the possibility and criticality of hazards. SAFER-HRC characterizes each of the elementary

actions within model T of the corresponding task by three main features: its pre-conditions, post-conditions, and safety properties (Fig.1 (b)). These features are formalized as TRIO formulae that have to hold respectively before, after and during execution of each action. In addition, each action has a property called *priority*, which defines its execution preference over other actions. More precisely, if at a time instant the pre-conditions of both $action_i$ and $action_j$ are satisfied, the one with higher priority starts to execute. The current model considers that systems operate at their maximum level of parallelism; that is, all actions that have the highest priority among those that are enabled start executing in parallel. Each action can also have additional constraints and timing requirements that are included in its formalization. At each instant, an action is in one of the following states:

1. *ns (not started)*: pre-conditions are not yet satisfied.
2. *wait*: pre-conditions are satisfied, but there is another action with higher priority in execution or waiting mode.
3. *exe (executing)*: under execution (solo, or concurrently with other actions).
4. *pause (ps)*: at some point in the execution, safety properties are violated and execution is paused.
5. *dn(done)*: the execution is terminated.

Model Tailoring. When applying SAFER-HRC to perform the safety analysis for a system, SATeam first needs to tailor the O-R-L model to the target HRC application, by selecting the appropriate robot model and application parameters, which corresponds to carrying out the following activities (see Fig.1):

- Choose the tasks that the robot will be executing.
- Set the number of operators and robots. In case the application requires more than one element for each category, SAFER-HRC creates multiple, separate instances of elements R and O.
- Define the configuration of the layout, in terms of the number of regions, reachability of each region for robot(s) and operator(s), and specification of obstacles or other physical features.

At this point, SAFER-HRC checks through the Zot formal verification tool whether the model so tailored satisfies the desired safety requirements. If a safety property is violated, Zot produces a counterexample, signaling the presence of one or more hazards in the system. A violation can be due to: (i) the system still includes hazardous situations, or allows for operator errors or on-purpose misuses; or (ii) the system does not have proper reduction measures for identified hazards. Then, the designer should improve the system model: by adding proper risk reduction measures, which correspond to TRIO formulae that should avoid the violation; or by including new formulae to capture hazards that were undetected in the previous analysis. Next, a new validation is carried out on the improved model. The model is refined iteratively until no more violations occur.

The next section shows an example of using SAFER-HRC to design an assembly application for a KUKA Light Weight Robot (LWR).

5 Applying SAFER-HRC in practice

In this section we illustrate how SAFER-HRC works in practice. We used a test-case with a KUKA robot, performing an assembly task with one operator in the layout depicted in Fig.2. The scenario is the following:

> The operator fetches a workpiece from a bin and moves to the assembly position, where the robot screw-drives the workpiece to the pallet using N fixtures. Before the robot starts screw-driving each fixture, the operator must prepare it and put it in the right position. As soon as the screw-driving of all of the fixtures for the workpiece is finished, the operator can release the workpiece and leave the assembly position.

The execution of this task has a loop whose index spans the number of fixtures N to be screw-driven. It means that for example, if $N = 2$, then SAFER-HRC defines 2 instances of each action in the loop. For brevity, we provide only a simplified formalization of the test-case. Fig. 3 shows the complete list of actions of the task. As explained in Section 4, each action is formalized through its pre/post-conditions and safety properties according to a contract-based approach. We present as an example a snippet of the formalization of $action_9$ "screwing the workpiece".

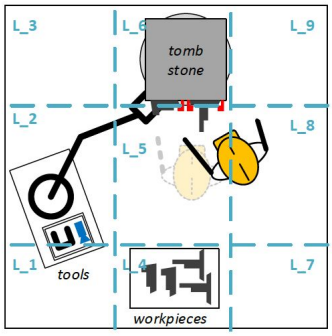

Fig. 2: Layout of the test system with nine areas. The assembly pallet board is in L_6 and the area blocked by the workpiece bin is L_4.

Pre-condition: There should be at least a prepared fixture. This property is captured by a Boolean predicate $prepared_{Fixture}$, which is set by data coming from visual sensors configured in the layout.

Safety property 1: Only the hands of the operator are allowed on the pallet. Since the O-R-L model includes in O an array to capture the position of the various body parts, and its seventh element refers to the hands, this corresponds to condition $Bodypart_7 = pallet$.

Safety property 2: The robot end-effector should be on the pallet. Model R captures the position of the end-effector through predicate EF, so this property simply corresponds to formula $EF = pallet$.

Safety property 3: The workpiece must be held. This simply corresponds to predicate $wpHeld$ being true.

Post-condition: end-effector and operator hands are still on the pallet: $Bodypart_7 = pallet \wedge EF = pallet$.

Other formulae in T are dedicated to formalizing different allowed sequences of actions to execute in order to achieve the goal of the task. One way to achieve

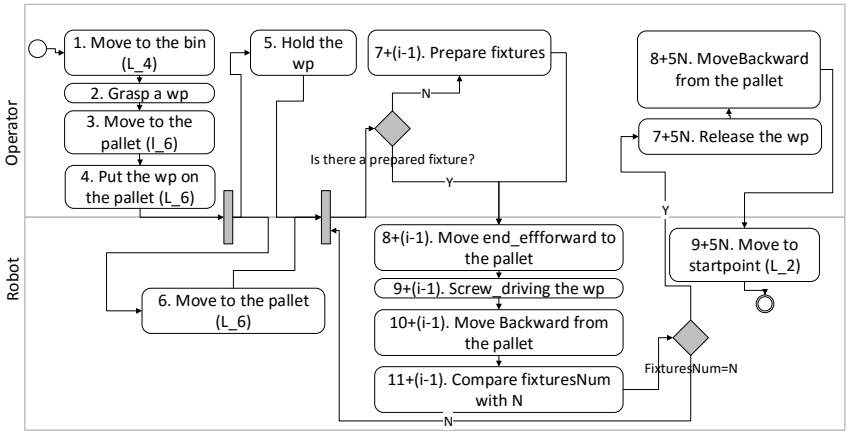

Fig. 3: Activity Diagram of the example task. The names of the actions in the loop are indexed by the current loop iteration i. There are 14 actions for $N = 1$.

this is by setting suitable values for the *priority* property of different actions. For example, $action_5$ "hold the workpiece" has higher priority than $action_7$ "prepare fixtures" in the definition of the task, since the operator must choose to give precedence to the former, even if he is ready to execute the latter. On the other hand, the robot must execute $action_6$ "move to the pallet" strictly before $action_8$ "move end-effector forward to the pallet", independent of the operator's choices; then, $action_6 = dn$ is defined as a pre-condition of $action_8$. Let us now provide some examples of formulae that are defined in T for this task. They are defined for each element $action_i$ of the set A_T of actions of the task.
(i) If an action has not started, it was never executing or done in the past:

$$action_{i,\text{state}} = \text{ns} \Rightarrow \neg\text{SomP}(action_{i,\text{state}} = \text{exe} \vee action_i = \text{dn})$$

(ii) If an action is waiting, it was never executing or done in the past, and it was in the "not started" state previously:

$$action_{i,\text{state}} = \text{wait} \Rightarrow \text{SomP}(action_{i,\text{state}} = \text{ns}) \wedge \neg\text{SomP}(action_{i,\text{state}} = \text{exe} \vee action_i = \text{dn})$$

(iii) If an action is executing (solo or concurrently with other actions), it has started in the past, it will never be starting or waiting again in the future, and it has not been done previously:

$$action_{i,\text{state}} = \text{exe} \Rightarrow \text{SomP}(action_{i,\text{state}} = \text{ns} \ \wedge action_i = \text{wait}) \ \wedge \neg\text{SomF}(action_{i,\text{state}} = \text{ns} \vee action_i = \text{wait}) \ \wedge \neg\text{SomP}(action_{i,\text{state}} = \text{dn})$$

(iv) If an action is paused, it was executing before that, and at some point it will restart its execution:

$$action_{i,\text{state}} = \text{ps} \Rightarrow \text{SomP}(action_{i,\text{state}} = \text{exe}) \wedge \text{SomF}(action_{i,\text{state}} = \text{exe}) \wedge \neg\text{SomF}(action_{i,\text{state}} = \text{ns} \wedge action_i = \text{wait})$$

(v) If an action is waiting, the next time unit it will start executing if there is no other waiting or executing action with higher priority:

$$action_{i,\text{state}} = \text{wait} \ \wedge \bigwedge_{j \in A_T, j \neq i} \begin{pmatrix} action_{j,\text{state}} = \text{wait} \vee action_{j,\text{state}} = \text{exe} \\ \Rightarrow action_{i,\text{priority}} \geq action_{j,\text{priority}} \end{pmatrix}$$
$$\Rightarrow \text{Futr}(action_{i,\text{state}} = \text{exe}, 1)$$

(vi) If multiple actions are waiting, those with higher priority will start to execute at the next time unit, whereas the others will remain waiting or go back to "not started" status (this can happen if their pre-condition stops holding):

$$action_{i,\text{state}} = \text{wait} \ \wedge \bigvee_{j \in A_T, j \neq i} \begin{pmatrix} action_{j,\text{state}} = \text{wait} \wedge \\ action_{i,\text{priority}} < action_{j,\text{priority}} \end{pmatrix}$$
$$\Rightarrow \text{Futr}(action_{i,\text{state}} = \text{ns} \vee action_i = \text{wait}, 1)$$

(vii) When execution of an action is done, it means that it was being executed in the past, and its state will not change in the future:

$$action_{i,\text{state}} = \text{dn} \Rightarrow \text{AlwF}(action_{i,\text{state}} = \text{dn}) \wedge \ \text{SomP}(action_{i,\text{state}} = \text{exe})$$

(viii) Each action must eventually terminate: $\bigwedge_{i \in A_T} \text{Som}(action_{i,\text{state}} = \text{dn})$

Model Tailoring. At this step, SATeam provides the details to instantiate the O-R-L model with the information specific to the target application, such as the actual layout of the common workspace. In the case study it is enough to introduce one instance each of O and R. Also, L is customized as follows. As Figure 3(a) shows, the layout of the cell is divided in nine regions. The positions of the pallet and of the workpiece bin are L_6 and L_4, respectively. Regions L_1 to L_6 are reachable by the operator, except for region L_4, where the bin is located. The adjacency of areas is defined through a matrix given by SATeam. After these configurations are carried out for the model, safety properties 2 and 3 mentioned above for $action_9$ become $Bodypart[7] = L_6$ and $EF = L_6$, respectively.[3]

Safety Analysis of the tailored O-R-L Model. In this step of the SAFER-HRC methodology, the SATeam carries out an iterative analysis, to find the operator errors that can cause serious problems (if they have not been taken into account in the initial model), or possible incompatibilities between layout and task execution. The analysis is done by formally verifying the O-R-L model described above against the safety properties of each action. The execution time for the verification activities is not a concern in this case study, since verification is completed in a few seconds using a modified plug-in [3] of the Zot bounded satisfiability checker. The verification bound (i.e., the maximum length of analyzed traces) was 100, which is over the completeness bound.

The following are examples of problems that SAFER-HRC found in the O-R-L model of the case study.

[3] The complete O-R-L Model can be found at github.com/Askarpour/ORL-Model.

(a) While $action_9$ is executing, the operator mistakenly gets close to the pallet with her face (for example, she might want to see the screw-driving action better) and when $action_{10}$ starts to execute the robot hits her face. This can cause serious injuries in the face and eye area, so we modified the safety property package of $action_9$. More precisely, we added the following formula, which states that no other body part other than the hand is allowed in the area close to the pallet, and in case the operator makes such mistake the execution is paused (in fact, whenever a safety property of an action is violated, the execution of that action is paused): $\bigwedge_{i \in BodyIndexes \wedge i \neq 7} \neg(Bodypart[i] = pallet)$.

(b) As mentioned above, the concurrency of actions depends on the values of their priorities. In some cases, the inconsistencies that might happen during the concurrent execution of actions have been avoided by design, through the definition of suitable pre/post-conditions. However, this issue has not been addressed in the initial model between $action_9$ and $action_5$. In fact, the safety property of $action_5$ is not satisfied, and according to the counterexample returned by Zot, there are system configurations where $action_9$ is executing, but the workpiece is not held by the operator. This highlights two issues: (i) the operator could make an error and release the workpiece before the screw-driving action terminates; (ii) $action_9$ should always execute concurrently with $action_5$. To circumvent this, the safety properties of $action_9$ are updated by adding formula $action_5 = \text{exe}$ to them. The modification is applied also to $action_8$, $action_{10}$, and $action_{11}$.

6 Conclusions

This paper introduced the SAFER-HRC methodology for the semi-automated safety analysis of HRC applications. The methodology is based on formal verification techniques to explore foreseeable wanted and unwanted interactions (errors and misuses) between operators and robots. We have applied the methodology to a realistic case study consisting of a KUKA robot performing an assembly task. Our approach allows a team of system safety experts to: (i) create formal models of HRC applications that can be flexibly modified to take into account different layout configurations and safety requirements; (ii) identify operational hazards caused by the relations and interactions among operators, robots, layouts and tasks; and (iii) introduce and validate suitable reduction measures to counter them. Unlike other approaches, our methodology emphasizes the effects of the presence of operators in the system and their choices in the execution order of the actions within a task.

As future work, we will include risk estimation techniques into the methodology to evaluate the level of risk associated with different possible execution orders of actions within a task. This will allow us to compare the criticality of each ordering and to help the operator to choose the one with the lowest risk value. We also aim to develop a framework based on the presented methodology to support safety engineers from the early design phases—e.g., semi-formal descriptions of tasks—to the introduction of risk reduction measures mitigating identified hazards.

References

1. The Zot bounded satisfiability checker. Available from github.com/fm-polimi/zot
2. Baracchi, L., Cimatti, A., Garcia, G., Mazzini, S., Puri, S., Tonetta, S.: Requirements refinement and component reuse: The FoReVer contract-based approach. Handbook of Research on Embedded Systems Design (2014)
3. Baresi, L., Kallehbasti, M.M.P., Rossi, M.: How bit-vector logic can help improve the verification of LTL specifications over infinite domains. In: Proc. of SAC. pp. 1666–1673 (2016)
4. Dhillon, B.S., Fashandi, A.R.M.: Safety and reliability assessment techniques in robotics. Robotica 15(6), 701–708 (1997)
5. Espiau, B., Kapellos, K., Jourdan, M.: Formal verification in robotics: Why and how? In: Robotics Research, pp. 225–236. Springer (1996)
6. Fung, P., Norgate, G., Dilts, T., Jones, A., Ravindran, R.: Human-in-the-loop machine control loop (1992), Patent nr. US 5116180 A
7. Furia, C.A., Mandrioli, D., Morzenti, A., Rossi, M.: Modeling Time in Computing. Monographs in Theoretical Computer Science. An EATCS Series, Springer (2012)
8. Guiochet, J.: Hazard analysis of human-robot interactions with HAZOP-UML. Safety Science 84, 225–237 (2016)
9. Guiochet, J., Do Hoang, Q.A., Kaaniche, M., Powell, D.: Model-based safety analysis of human-robot interactions: The MIRAS walking assistance robot. In: Proc. of the Int. Conf. on Rehabilitation Robotics (ICORR). pp. 1–7 (2013)
10. International Electrotechnical Commission: IEC 61882, Hazard and operability studies (HAZOP studies)-Application guide (2001)
11. International Standard Organisation: ISO10218-2:2011, Robots and robotic devices - Safety requirements for industrial robots - Part 2: Robot Systems and Integration
12. International Standard Organisation: ISO12100:2010, Safety of machinery - General principles for design - Risk assessment and risk reduction
13. International Standard Organisation: ISO13849-1:2015, Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design
14. International Standard Organisation: ISO14121-2:2007, Safety of machinery - Risk assessment - Part 2
15. International Standard Organisation: ISO/TS15066:2015, Robots and robotic devices – Collaborative robots
16. Leveson, N.: Engineering a safer world: Systems thinking applied to safety. MIT Press (2011)
17. Machin, M., Dufossé, F., Blanquart, J., Guiochet, J., Powell, D., Waeselynck, H.: Specifying safety monitors for autonomous systems using model-checking. In: Proc. of SAFECOMP. pp. 262–277 (2014)
18. Machin, M., Dufossé, F., Guiochet, J., Powell, D., Roy, M., Waeselynck, H.: Model-checking and game theory for synthesis of safety rules. In: Proc. of HASE (2015)
19. Martin-Guillerez, D., Guiochet, J., Powell, D., Zanon, C.: A UML-based method for risk analysis of human-robot interactions. In: Proc. of SERENE. pp. 32–41. ACM (2010)
20. Pradella, M., Morzenti, A., San Pietro, P.: Bounded satisfiability checking of metric temporal logic specifications. ACM TOSEM 22(3), 20:1–20:54 (2013)