

Testing Robot System Safety by creating Hazardous Human Worker Behavior in Simulation

Tom P. Huck, Christoph Ledermann, and Torsten Kröger

Abstract— We introduce a novel simulation-based approach to identify hazards that result from unexpected worker behavior in human-robot collaboration. Simulation-based safety testing must take into account the fact that human behavior is variable and that human error can occur. When only the *expected* worker behavior is simulated, critical hazards can remain undiscovered. On the other hand, simulating *all* possible worker behaviors is computationally infeasible. This raises the problem of how to find *interesting* (i.e., potentially hazardous) worker behaviors given a limited number of simulation runs. We frame this as a search problem in the space of possible worker behaviors. Because this search space can get quite complex, we introduce the following measures: (1) Search space restriction based on workflow-constraints, (2) prioritization of behaviors based on how far they deviate from the nominal behavior, and (3) the use of a risk metric to guide the search towards high-risk behaviors which are more likely to expose hazards. We demonstrate the approach in a collaborative workflow scenario that involves a human worker, a robot arm, and a mobile robot.

I. INTRODUCTION

Hazard analysis is a major challenge in human-robot collaboration (HRC). As HRC is increasingly deployed in industrial practice, effective methods are needed to ensure safety of human workers. While traditional robot systems maintain safety through spatial separation, HRC requires more sophisticated measures, like adaptive speed- and workspace limitations, area monitoring (e.g. by laser scanners and light curtains), collision detection, and more [1]. These measures must be configured appropriately with respect to workflow and robot properties like stopping time and collision potential. Furthermore, the measures must ensure safety not only for expected worker behaviors, but also for unforeseen or erroneous behaviors. To ensure that these requirements are met, a *hazard analysis* is performed before commissioning [2], [3]. Current hazard analyses are mostly based on human reasoning, expert knowledge, and experience. In recent years, new approaches including formal verification [4], [5] and rule-based expert systems [6], [7] have been proposed. While these approaches are viable, they typically operate on rather abstract models. Thus, their effectiveness is limited for systems that are complex or require fine-grained analysis. In such cases, *simulation-based testing* [8], [9] is useful, as it

This work was funded by the German Federal Ministry of Economics in the research project ‘FabOS’.

The authors are with the Intelligent Process Automation and Robotics Lab, Institute of Anthropomatics and Robotics (IAR-IPR), Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany.

Corresponding author: Tom P. Huck (tom.huck@kit.edu)

This is a preprint of a paper accepted for publication in the IEEE Robotics and Automation Letters (RA-L). The final published version may differ.

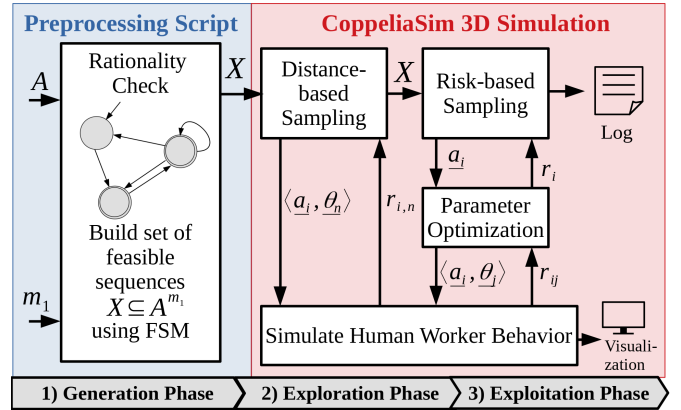


Fig. 1: Worker behavior is modeled with action sequences \underline{a} and parameters $\underline{\theta}$. Hazardous behaviors are found by: (1) Building a set X of feasible action sequences (Generation), (2) simulating sequences $\underline{a}_i \in X$ with nominal parameters $\underline{\theta}_n$ to get an initial risk estimate $r_{i,n}$ (Exploration), and (3) using optimization to find motion parameters that further increase risk (Exploitation).

can represent certain effects (e.g. collisions) more accurately. However, simulation of HRC requires simulating the human worker, which is challenging. Given the variability of human behavior and the possibility of human error, it is insufficient to simulate only the expected worker behavior. Instead, effective simulation-based tests should also cover deviating behaviors to see if they create unsafe states. In complex systems, it may not be immediately obvious what types of behaviors are potentially hazardous. Furthermore, as worker behavior is complex and the number of simulation runs is limited by the computational budget, exhaustive simulation of all possible behaviors is infeasible. This leads to the search problem of finding hazardous worker behaviors given a limited number of simulation runs. We address this problem as follows:

1) *Rationality checks*: We restrict the search space by rejecting action sequences that violate workflow-constraints.

2) *Prioritization*: If the remaining sequences are still too many for exhaustive simulation, we prioritize sequences closer to the nominal behavior based on a distance metric.

3) *Risk-guided search*: We use a risk metric to guide the search towards high-risk behavior, which increases the chances of uncovering hazards.

Note that this paper is not concerned with the development of digital human models. Instead, we assume a human model as given and focus on the aforementioned search problem.

Fig. 1 shows the general outline of our approach, which

is structured in three phases: In the generation phase, a preprocessing script builds a set of feasible sequences using workflow constraints that are modeled in a finite state machine (FSM). The sequences are fed into the 3D-simulator *CoppeliaSim*¹ to search for the occurrence of hazardous states. The search itself is structured in an exploration-phase, where initial risk estimates for the sequences are obtained, and an exploitation phase that attempts to find especially high-risk executions of the sequences (details in Sec. IV and V).

II. BACKGROUND AND RELATED WORK

A. Robot Safety and Hazard Analysis in Industrial Practice

Safety requirements for industrial robot systems are specified by ISO/TS 15066 [1] and ISO 10218 [2] (the former is intended specifically for HRC). Depending on the robot's capabilities and mode of operation, safety can be achieved through different measures like protective stops, speed and separation monitoring, collision force limitation, or combinations thereof. To implement these measures, collaborative robots (e.g., the *KUKA LBR iiwa* or the *Universal Robots UR10e*) typically provide a set of safety functions such as software-based workspace- and speed-limitations which are programmed and parameterized by users or system integrators according to their specific needs. Additionally, external safety sensors (e.g. light curtains, laser scanners, camera systems) are often used to monitor robot vicinity for approaching humans [10]. Choice and configuration of safety measures are highly use-case dependent and influenced by various system-specific factors (e.g., cell layout, reachability of hazardous areas, robot stopping time/distance etc.) [11]. To ensure that all relevant hazards are properly addressed, HRC systems are subjected to a hazard analysis² prior to commissioning [2]. This is a multi-step procedure to identify risks, estimate and evaluate potential hazards, and determine appropriate safety measures. Standards explicitly require that the hazard analysis does not only consider the nominal workflow, but also erroneous worker behavior [3]. In current practice, hazard analysis is largely based on human reasoning, expert knowledge, experience, and simple tools (e.g., checklists) [3], [12].

B. Novel Hazard Analysis Methods

Current hazard analysis methods are not well-suited for complex (HRC) systems. Thus, a variety of novel methods have been proposed in scientific literature. We shortly introduce the most prominent approaches. For a comprehensive review, we refer to [13].

1) *Semi-formal Methods*: These methods use semi-formal (often graphic) system representations such as control structure diagrams [14], UML diagrams [15] or task ontologies [16] to help the user analyze system behavior and identify potential hazards. Often, they also use a set of guide words to guide the user through the procedure [14], [17]. While

these methods provide a certain level of support, they have a limited potential to be automated, since core aspects of the hazard analysis are still based on human reasoning.

2) *Formal Verification*: Formal verification methods automatically check a system model in a formal language for possible violation of safety criteria, allowing for a certain extent of automation in the hazard analysis. In industrial HRC, this approach is mainly pursued by the *SAFER-HRC*-method, which is based on the formal language *TRIO* and the satisfiability checker *ZOT* [4]. The method has also been extended to cover erroneous worker behavior [18] and can be used in conjunction with a 3D simulator [19] for visualization. Besides industrial HRC, formal safety verification has also been applied to robot systems for personal assistance, medicine, and transport [20]–[22].

3) *Rule-based Expert Systems*: Rule-based expert systems also provide a certain degree of automation. They use a domain-specific model, such as the *Product Process Resource* model (PPR) [23], to describe safety-related workspace characteristics (e.g. layout, which tools are used, and which tasks are performed). Pre-defined rule-sets map these workspace characteristics onto a set of potential hazards. Additional rule-sets can assist the user with adjacent tasks such as risk estimation or decisions about appropriate safety measures [6], [7].

C. Simulation-based Safety Testing

The methods from Sec. II-B are based on relatively abstract system models (e.g., control structure diagrams, formal language descriptions, PPR models) that often require strong modeling simplifications and thus, their applicability is limited when it comes to detailed effects such as human motions or human-robot collisions. These effects can be explored better through *simulation-based testing*. But as simulation models are often complex and safety-critical states rare, creating interesting and critical test scenarios becomes a challenge [24]. Often, one uses algorithms for search, optimization, or machine-learning to find simulation conditions that expose hazards of failures [25]. Different heuristics (e.g. risk metrics or coverage criteria) are used to guide the algorithms [26]–[28]. So far, simulation-based safety testing has mainly been applied to autonomous vehicles [29]–[32] and aerospace systems [33], [34].

In robotics, simulation-based testing has been used mainly for verification and validation on the level of individual system components (e.g., controller- or code-validation) [8], [35], [36]. Recently, however, the idea of using simulation-based testing as a tool for hazard analysis has also gained some interest. In our earlier work, we proposed the use of Monte Carlo Tree Search (MCTS) to find hazardous states in HRC simulation [9], [37]. A similar concept was explored in [38], but with Deep Reinforcement Learning instead of MCTS. Other recent publications also consider simulation-based testing as a part of their hazard analysis strategies [39], [40]. Despite this recent interest, the potential of simulation-based testing for hazard analysis in HRC is still relatively unexplored, especially when compared to the wide-spread

¹<https://www.coppeliarobotics.com/>

²Note that there are also other terms for this, e.g. "hazard and risk analysis" or "risk assessment". Their meanings differ slightly. For simplicity we only use the term "hazard analysis".

use that simulation-based testing methods have found in other domains such as autonomous vehicles.

III. GOAL AND PROBLEM DEFINITION

A. Hazard Analysis as a Search Problem

This paper focuses on simulation-based safety testing for HRC systems and, more specifically, on the simulation of worker behavior. Our goal is to expose potential hazards by subjecting simulation models of HRC systems to critical worker behaviors that induce unsafe states. We frame this problem as a *search problem* with the goal of finding hazardous behaviors in the search space of possible worker behaviors. We assume that all non-human entities (e.g. robots, sensors, etc.) react deterministically for a given worker behavior³. Assuming a fixed initial state, the behavior of the worker can thus be considered as the only variable on which the resulting simulation states depend. Under this assumption, we can express the search problem as a 4-tuple:

$$\langle \mathcal{M}, s_0, B, U \rangle \quad (1)$$

where \mathcal{M} is the simulation model, s_0 the initial simulation state, B the set of possible worker behaviors, and U a (user-defined) set of unsafe states. The goal is now to *find hazardous behaviors* $\tilde{b} \in B$ for which a simulation starting in s_0 results in an unsafe state s_U :

$$s_U = \mathcal{M}(s_0, \tilde{b}), \quad s_U \in U \quad (2)$$

B. Modeling of Worker Behavior

Human behavior can be modeled on different levels from abstract action sequences to concrete motions [41]. In case of a human worker, unsafe states may result from deviations on a workflow-level (e.g., leaving out worksteps or switching their order), on a motion-level (e.g., walking unexpectedly fast or slow), or from combinations of both. We therefore model worker behavior as a tuple $b = \langle \underline{a}, \underline{\theta} \rangle$ consisting of an action sequence \underline{a} and a parameter vector $\underline{\theta}$:

$$\text{with } \underline{a} = (a_1, a_2, \dots, a_{m_1}) \quad a_i \in A \quad (3)$$

$$\text{and } \underline{\theta} = (\theta_1, \theta_2, \dots, \theta_{m_2}) \quad \theta_i \in \mathbb{R} \quad (4)$$

where the action sequence \underline{a} denotes the order of worksteps and the parameter vector $\underline{\theta}$ defines how these worksteps are executed on the motion level (i.e., aspects like walking speed, goals of reaching motions, etc.). Further, A denotes the worker's action space (i.e., the set of all possible worksteps), m_1 the length of the action sequence, and m_2 the number of parameters. We also assume that there is a known *nominal behavior*, consisting of a nominal action sequence \underline{a}_n and nominal parameters $\underline{\theta}_n$. The nominal behavior represents the behavior that is intended and expected in a normal, non-erroneous action sequence. With this worker model, our search problem has a mixed discrete-continuous search space of possible behaviors B :

$$B = A^{m_1} \times \mathbb{R}^{m_2} \quad (5)$$

³This assumption may seem very restrictive at first, but is justified since industrial robots are usually pre-programmed and robust to external influences. For significant non-deterministic effects (e.g. varying payloads and stopping distances), a conservative worst-case assumption can be made.

C. Example

To make the previous definitions more concrete, consider the example in Fig. 2. A worker, a stationary robot, and an automated guided vehicle (AGV) collaborate in a cyclic workflow. The worker's task is to deliver unmachined workpieces to the robot, which processes them and places them on the AGV that carries them away. This means that one cycle is as follows: Initially, the worker stands in front of the robot as a workpiece is processed. After processing, the robot loads the workpiece on the AGV, which takes it away. Meanwhile, the worker transitions to another station (Fig. 2-1) and picks up a new workpiece (Fig. 2-2). The worker waits until the AGV has cleared the area (Fig. 2-3), transitions back to the robot (Fig. 2-4) and puts down the new workpiece in front of the robot (Fig. 2-5). The robot senses the worker's arrival via a sensor mat (orange area in Fig. 2), moves towards the workpiece and starts the processing routine. Meanwhile, the AGV returns to its initial position. Action space and nominal action sequence are:

$$A = \{t, p, w, d\} \quad (6)$$

$$\underline{a}_n = (t, p, w, t, d, w) \quad (7)$$

with t, p, w and d denoting the actions (*transition*), (*pick up part*), (*wait*) and (*put down part*), respectively. Possible parameters include the worker's walking speed v and the workpiece placement coordinates x, y on the table⁴:

$$\underline{\theta} = (v, x, y) \quad (8)$$

Note that there are $|A|^n = 4^6 = 4,096$ potential action sequences. For each sequence, there is a continuous parameter space in \mathbb{R}^3 . Even with a relatively coarse parameter discretization into five steps, the search space would still consist of $4096 \cdot 5^3 = 512,000$ possible behaviors.

IV. PROPOSED SOLUTIONS

As the example shows, the search space (i.e., the space of possible worker behaviors) can be vast, even in simple models. Exhaustively simulating all possible behaviors is infeasible. Instead, we propose the following three measures:

A. Rationality Checks based on Workflow-Constraints:

It is usually not the case that any action can take any place in a workflow. Instead, there are certain constraints (e.g., a worker cannot put down a workpiece before picking it up). Using such constraints, we can reduce the search space by excluding infeasible sequences a priori (i.e., before running any simulations). We use a Finite State Machine (FSM) to define which actions are possible in which state of the workflow. The set of feasible action sequences $X \subseteq A^{m_1}$ is the set of all action sequences that are accepted by the FSM (here, "accepted" means that it must be possible to run the full action sequence in the FSM without ending up in a state where the required action is impossible).

⁴Depending on the desired level of detail, other parameters would be possible as well, e.g. waypoints to describe the worker's walking path. Here we only consider these three parameters for the sake of simplicity.

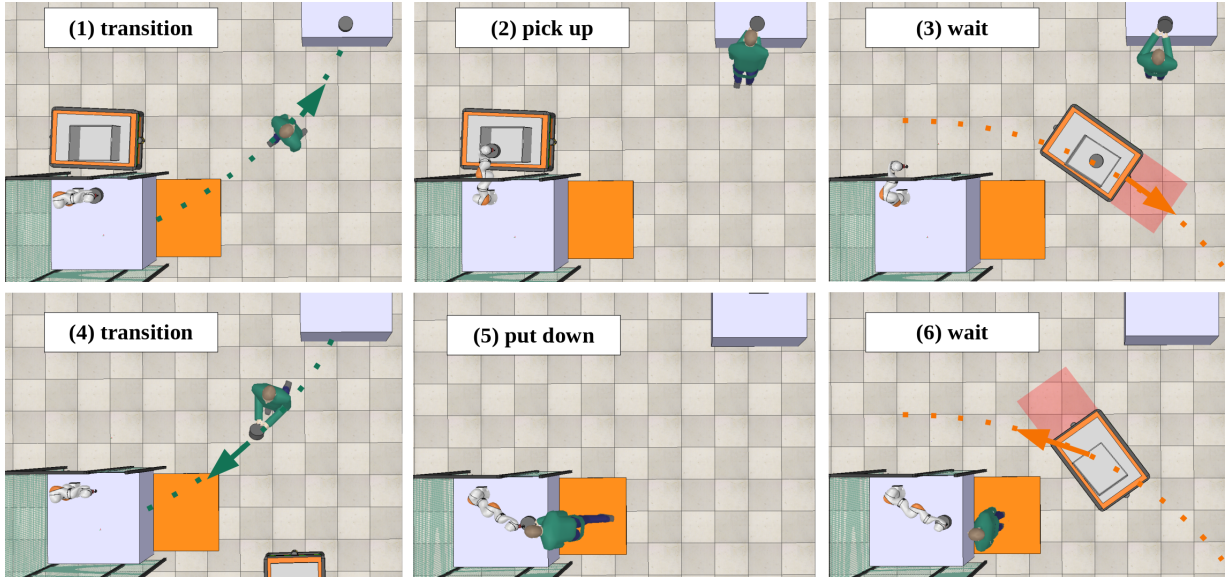


Fig. 2: A cycle of a simple example workflow: The worker transitions from the robot to the storage station (1), picks up a part (2), waits for the AGV to pass (3), transitions back (4), puts down the part in front (5) and waits for the AGV to return (6).

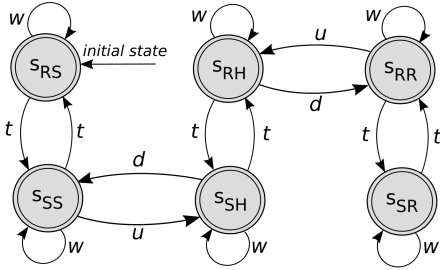


Fig. 3: The finite state machine determines the feasible action sequences. Note: The first index of the states refers to the worker's position (R: at robot station, S: at storage station) and the second index to the workpiece position (R: at robot station, S: at storage station, H: in worker's hands)

For an example, consider Fig. 3, which shows the FSM corresponding to the example workflow from Sec. III-C. The sequence (t, u, w, t, d, w) , for instance, is accepted, whereas the sequence (t, d, u, w, t, d) is infeasible, because the second transition (d : put down) is impossible: The worker cannot put down the workpiece in state S_{SS} because it has not been picked up previously.

B. Distance-based prioritization of action sequences:

The second measure is to prioritize sequences that are closer to the nominal sequence \underline{a}_n . In sequential workflows, erroneous behavior typically results from certain basic errors being superposed to \underline{a}_n [42]. Thus, the majority of erroneous sequences are likely to occur in a certain vicinity around \underline{a}_n , while highly erratic sequences can be regarded as unlikely. If the computational budget is not sufficient to simulate all feasible sequences, it is therefore reasonable to concentrate on exploring the vicinity of \underline{a}_n . To quantify how far a given sequence \underline{a} is from \underline{a}_n , we define a distance metric. Based on [42], we assume four basic error types, namely (i) *insertion*

of an action into the sequence, (ii) *omission* of an action, (iii) *substitution* of an action by another, and (iv) *reversing* the order of two adjacent actions⁵ (Note: Since we assume a fixed action sequence length m_1 , sequences subjected to an insertion are truncated to length m_1 by dropping the last action, and sequences subjected to an omission are extended to length m_1 by inserting an arbitrary action $a \in A$ at the end). We define the distance between \underline{a} and \underline{a}_n as the minimum number of basic errors required to convert \underline{a}_n into \underline{a} . We call this the *error distance* $d_e(\underline{a}, \underline{a}_n)$.

For an example, consider $\underline{a}_n = (t, u, w, t, d, w)$. The following sequence \underline{a}_1 results from switching the last two actions in \underline{a}_n , whereas the action sequence \underline{a}_2 results from three action substitutions:

$$\underline{a}_1 = (t, u, w, t, w, d), \quad \rightarrow d_e(\underline{a}_1, \underline{a}_n) = 1 \quad (9)$$

$$\underline{a}_2 = (t, w, w, u, w, w), \quad \rightarrow d_e(\underline{a}_2, \underline{a}_n) = 3 \quad (10)$$

Note that the definition of d_e is analogous to the Damerau-Levenshtein distance commonly used to measure the dissimilarity between character strings [43]. Thus, we can use an existing algorithm⁶ to efficiently calculate d_e .

C. Risk-guided search:

Thirdly, we use a *risk metric* to guide the search towards high-risk behaviors. To account for the mixed discrete-continuous search space (compare Eq. (5)), we use a two-stage search algorithm where the top level samples action sequences guided by the risk metric and the bottom level performs parameter optimization to find high-risk executions of a given action sequence (more in Sec. V). Since common risk metrics from safety standards (e.g. [3], [12]) are typically based on human judgement, they are difficult to calculate

⁵Note that [42] lists more than four error types. However, the additional types are special cases of these four.

⁶<http://www.ccpa.puc-rio.br/software/stringdistance/>

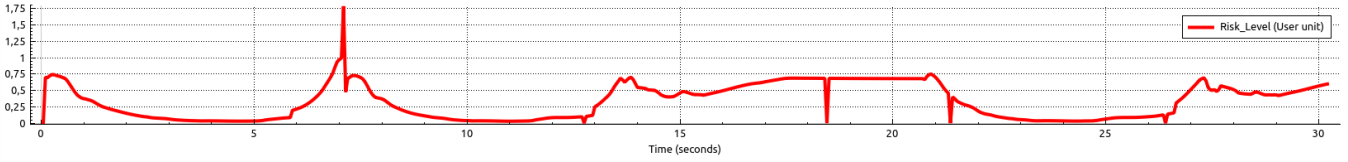


Fig. 4: Example of the risk level r over simulation time. The high peak corresponds to a collision, the low peaks to protective stops.

automatically. Thus, we define a simplified heuristic metric r to quantify the risk:

$$r = \begin{cases} 0 & \text{case (a): } v_R < v_{crit} \\ e^{-d_{HR}} & \text{case (b): } v_R \geq v_{crit}; d_{HR} > 0 \\ \frac{F_c}{F_{max}} + 1 & \text{case (c): } v_R \geq v_{crit}; d_{HR} = 0 \end{cases} \quad (11)$$

Here, d_{HR} denotes the human-robot distance, v_R denotes the robot speed⁷, v_{crit} is a user-defined speed threshold above which the robot is considered potentially dangerous (a typical choice is 250mm/s, see [2]). F_c denotes the estimated human-robot collision force, and F_{max} is a body-region specific collision force limit [1].

The metric differentiates between three types of situations:

- (a) The robot is slower than v_{crit} and thus poses a negligible danger to the worker; the risk is zero.
- (b) The robot is faster than v_{crit} , but the distance d_{HR} is greater than zero. Thus, there is no contact (yet), but the robot poses a potential danger to the worker. In this case, the risk is defined on the basis of d_{HR} .
- (c) The robot speed is faster than v_{crit} and the robot is in contact with the worker (i.e. $d_{HR} = 0$). In this case, the risk is defined as the ratio of the collision force F_c to the limit F_{max} . The addition of +1 ensures that case (c) always returns higher risk than case (b).

For a given behavior $b \in B$, the risk is obtained by simulating b , evaluating r in each simulation step (see Fig. 4), and returning the highest value that has occurred. It is assumed that exactly one human worker and one or multiple robots are present. For multiple robots, r is evaluated separately with respect to each robot and the maximum value is taken. Since safety constraints in HRC are typically defined in terms of distance, velocity, and contact forces [1], the definition of r allows us to express the set of unsafe states U via a risk threshold r_{th} :

$$U = \{s \mid r > r_{th}\} \quad (12)$$

For instance, with $r_{th} = 1$ all contact situations at critical robot velocity are deemed unsafe, whereas with $r_{th} = 2$ only collisions exceeding F_{max} are deemed unsafe.

V. SEARCH METHOD

A. Assumptions, Objective, and General Approach

This section describes a concrete implementation of this approach. We assume that A , \mathcal{M} , and s_0 are given and that the lengths of action sequences and parameter vectors are m_1 and m_2 , respectively. We also assume that the nominal

⁷for articulated robot arms, v_R is the cartesian speed of the fastest joint; for AGVs, v_R is the robot's translational ground speed

behavior $\langle \underline{a}_n, \underline{\theta}_n \rangle$, parameter limits $\underline{\theta}_{min}, \underline{\theta}_{max}$, and risk threshold r_{th} are given. The objective of the implemented search method is to find as many hazardous sequences as possible. According to Eq. (2), a sequence $\tilde{\underline{a}}$ is hazardous if parameters $\tilde{\underline{\theta}}$ exist so that the behavior $\tilde{\underline{b}} = \langle \tilde{\underline{a}}, \tilde{\underline{\theta}} \rangle$ results in an unsafe state $s_U \in U$ (i.e., a state where r_{th} is exceeded). As seen in Fig. 1, the general procedure is structured in three phases: In the *generation phase*, a preprocessing script builds a set X of feasible action sequences. These are then fed into the 3D Simulator CoppeliaSim. In the *exploration phase*, CoppeliaSim simulates the sequences with their nominal parameters to obtain an initial risk estimate. In the *exploitation phase*, CoppeliaSim deploys a parameter optimizer to find parameters that further increase the risk of the previously explored sequences. We assume that the computational budget is limited in terms of the maximum number of simulation runs N_{max} which are split between exploration and exploitation phase by a split factor $\alpha \in [0, 1]$, with $N_{explore} = \lfloor \alpha \cdot N_{max} \rfloor$ simulation runs for exploration and the remaining runs for exploitation.

B. Generation Phase

The preprocessing script generates a set X of feasible action sequences using the FSM introduced in Sec. IV-A. The script iterates through the cartesian product A^{m_1} of the action space and checks for each $\underline{a} \in A^{m_1}$ if it is accepted by the FSM. In Algorithm 1, this is represented by ll. 1-7 (Note: The function ISFEASIBLE in l. 3 represents the acceptance check).

C. Exploration Phase

In the exploration phase (ll. 8-17), the budget $N_{explore}$ (i.e., the available number of simulation runs) is allocated according to the split factor α . If the number of sequences in X is smaller than $N_{explore}$, the remaining budget is allocated to the exploitation phase. Then, the simulator calculates the distance to \underline{a}_n for the sequences in X and orders the sequences from low to high distance. (Note: For brevity, both the distance calculation and the sorting are represented by a single function in Algorithm 1, l. 14.) The sequences in X are simulated with nominal parameters $\underline{\theta}_n$ until the budget of $N_{explore}$ simulations is exhausted. Thereby, an initial risk estimate $r_{i,n}$ is obtained for each simulated sequence. Due to the ordering, sequences closer to \underline{a}_n are prioritized if $N_{explore}$ is insufficient to simulate *all* sequences in X .

D. Exploitation Phase

When $N_{explore}$ is reached, the simulator transitions into the exploitation phase (ll. 18-31) and deploys a parameter

optimizer to find parameters that further increase the risk of the explored sequences. Since $N_{exploit}$ is limited, usually not all sequences can be optimized. Instead, sequences are prioritized according to their initial risk estimates since high-risk sequences have a better chance of exceeding r_{th} . For the prioritization, two alternatives are implemented:

1) *Strict priority*: Sequences are ordered strictly by decreasing risk estimate (Algorithm 1, l. 18).

2) *Probabilistic priority*: Sequences are sampled with probabilities proportional to their risk estimate (this variant is not shown in Algorithm 1):

$$P(\underline{a}_i) \propto r_{i,n}, \quad a_i \in X \quad (13)$$

Note that the latter also includes sequences with a low $r_{i,n}$ which would potentially be left out by strict prioritization. Sequences whose initial risk estimate already exceeds r_{th} are removed from X (these are already identified as hazardous, so further simulations would be futile).

For optimization we use a Nelder-Mead Optimizer (NMO). NMO is chosen because it is a *direct search* method that does not require gradient information. This is necessary because the risk value is obtained from simulation, and not from an analytical function, so gradients are not available. Since NMO is well known, we omit a detailed description and refer to [44]. The optimization loop is shown in ll. 24-28. NMO.INIT(m_2) randomly initializes the NMO algorithm and returns an initial parameter vector $\underline{\theta}$. Then, the first sequence is simulated with these parameters, and the resulting risk is given to NMO.ITERATE(r), which performs one step of the NMO algorithm and returns a new $\underline{\theta}$. This is repeated iteratively until r_{th} is exceeded or a maximum number of steps N_{NMO} is reached, in which case the algorithm moves on to the next sequence. Penalty functions ensure that parameters remain in $[\underline{\theta}_{min}, \underline{\theta}_{max}]$. (Note: For brevity, this aspect is not shown in Algorithm 1.) The exploitation phase terminates when the budget $N_{explore}$ is exhausted.

VI. APPLICATION EXAMPLE

A. Scenario

We present an example using the workflow from Sec. III-C (a second example with a slightly more complex workflow is available on GitHub⁸, but is not discussed here for reasons of brevity). Action space A and parameters $\underline{\theta}$ are chosen according to Eq. (6) and Eq. (8), respectively. The nominal sequence \underline{a}_n is chosen according to Eq. (7). The workflow constraints are given by the FSM shown in Fig. 3 (see Sec. IV-A for details). Our safety criterion is that there must be no collisions where the contact force exceeds 70% of the collision force limit. Thus, we set the risk threshold to $r_{th} = 1.7$ (compare Eq. (11)). The set of unsafe states is:

$$U = \{s \mid r > 1.7\} \quad (14)$$

For testing purposes, deliberate safety flaws are designed into the system, so that both robot and AGV can cause an unsafe state if the worker behaves in certain ways that deviate from

Algorithm 1 Search Method Pseudocode

```

1:  $X \leftarrow \langle \rangle$ ,  $i \leftarrow 1$ 
2: for each  $\underline{a} \in A^{m_1}$  do # select feasible sequences
3:   if ISFEASIBLE( $\underline{a}$ ) then
4:      $X_i.actions \leftarrow \underline{a}$ 
5:      $i \leftarrow i + 1$ 
6:   end if
7: end for
8: if  $|X| \geq \lfloor \alpha \cdot N_{max} \rfloor$  then # alloc. exploration budget
9:    $N_{explore} \leftarrow \lfloor \alpha \cdot N_{max} \rfloor$ 
10: else
11:    $N_{explore} \leftarrow |X|$ 
12: end if
13:  $N_{exploit} \leftarrow N_{max} - N_{explore}$  #remaining budget
14:  $X \leftarrow \text{SORTBYINCREASINGDISTANCE}(X)$ 
15: for  $i = 1, \lfloor \alpha \cdot N_{max} \rfloor$  do # exploration
16:    $X_i.risk \leftarrow \text{SIMULATE}(X_i.actions, \underline{\theta}_n)$ 
17: end for
18:  $X \leftarrow \text{SORTBYDECREASINGRISK}(X)$ 
19:  $X \leftarrow X \setminus \{X_i \mid X_i.risk > r_{threshold}\}$  # remove sequences
   that already exceed  $r_{th}$ 
20:  $i \leftarrow 1, j \leftarrow 1$  # i: sim runs, j: sequences in  $X$ 
21: while  $i < N_{exploit}$  do
22:    $k \leftarrow 1$  # optimization step counter
23:    $\underline{\theta} \leftarrow \text{NMO.init}()$ 
24:   do
25:      $r \leftarrow \text{SIMULATE}(X_j.actions, \underline{\theta})$ 
26:      $\underline{\theta} \leftarrow \text{NMO.iterate}(r)$ 
27:      $k \leftarrow k + 1$ 
28:   while  $r < r_{threshold}$  and  $k < N_{NMO}$ 
29:    $j \leftarrow j + 1$  # go to next sequence
30:    $i \leftarrow i + k$  # k sim runs conducted in inner loop
31: end while

```

the nominal behavior (see Fig. 5). The generation phase yields 266 feasible action sequences from the initially 4096 potential combinations.

B. Test Runs and Results

The 266 remaining feasible sequences are searched for hazardous behavior in several different test runs.

1) *Comparison of search configurations*: Test runs with a constant split factor were performed to compare different search configurations: Search with strict prioritization (SP), search with probabilistic prioritization (PP), and, as a baseline, a random search (R) where sequences and parameters are sampled from uniform distributions over X and $[\underline{\theta}_{min}, \underline{\theta}_{max}]$ without any form of distance- or risk-prioritization. For each approach, ten test runs were conducted with $N_{max} = 300$ and $N_{max} = 500$, respectively. The results are reported in Table I. For $N_{max} = 500$, SP outperforms R in terms of the number of identified hazards, while the error distances of the identified hazards are similar. For $N_{max} = 300$, PP slightly underperforms R, but the identified hazards are closer to the nominal behavior than those found by R. Interestingly, PP underperforms both R and SP as it tends

⁸<https://github.com/Huck-KIT/Robot-Hazard-Analysis-Simulation>

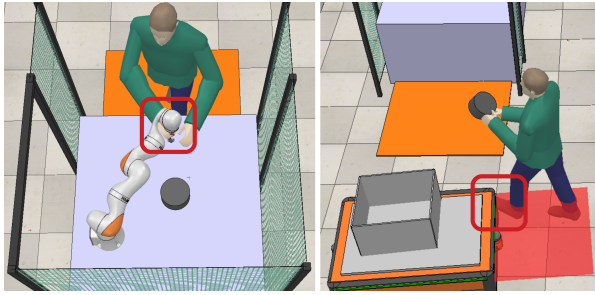


Fig. 5: Left: The worker’s hand collides with the robot after placing the workpiece on the table. Right: Worker and AGV collide.

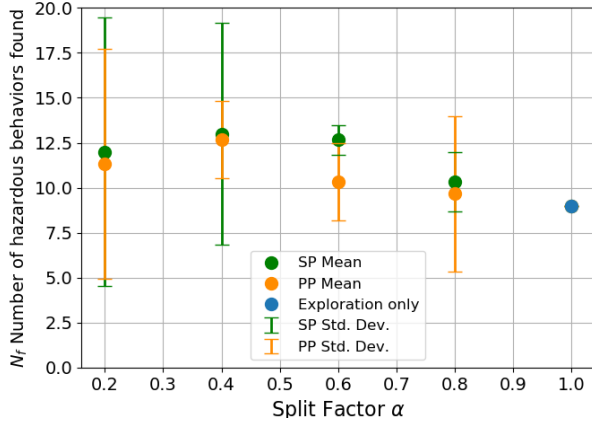


Fig. 6: Split factor sweep. Search configuration: SP, $N_{max} = 266$, $N_{NMO} = 8$. For each α , three test runs were conducted, except for $\alpha = 1$, where the search is fully deterministic (exploration only).

to waste simulation time with low-risk sequences that have little chance of exposing any hazard.

2) *Split factor sweep*: Additionally, tests with a fixed search configuration, but different split factors were conducted. For these tests, $N_{max} = 266$ was chosen (the same as the number of feasible sequences), so that an exploration-only search corresponds with $\alpha = 1$. Results are shown in Fig. 6. The best results were obtained with a medium split factor ($\alpha = 0.4$ and $\alpha = 0.6$), which is expected as a small α leads to insufficient exploration of the search space, whereas a large α leaves too few simulation runs for optimization. Yet, one should keep in mind that the influence of α is problem-dependent and might look very different in another scenario.

VII. DISCUSSION AND FUTURE WORK

The presented method enables users to consider effects of deviating worker behavior in their hazard analyses. The proposed simulation-based approach has several benefits: Simulation models are often already available in the development phase, so renewed system modeling is largely avoided (except for modeling the FSM to describe workflow constraints). Also, compared to PPR models or formal language descriptions, simulation captures physical effects (e.g. movements or collisions) more accurately. Finally, the simulation is treated similarly to a black-box model: Apart from action space, parameter ranges, and workflow constraints, no explicit knowledge about the systems’ internal workings is

#	Results for $N_{max} = 500$						Results for $N_{max} = 300$					
	R		SP		PP		R		SP		PP	
	N_f	d_e	N_f	d_e	N_f	d_e	N_f	d_e	N_f	d_e	N_f	d_e
1	27	3.1	36	3.2	24	3.1	20	3.2	19	2.6	12	2.9
2	27	3.2	38	3.2	21	3.1	17	3.2	19	2.8	12	2.8
3	24	2.5	37	3.3	22	2.6	9	3.2	15	2.9	12	2.6
4	27	3.1	32	3.2	26	3.3	20	2.6	19	2.8	10	2.8
5	24	2.8	37	3.2	25	2.8	24	2.6	15	2.4	13	2.8
6	25	3.3	34	3.2	27	2.8	16	3.1	17	3.0	12	2.7
7	29	3.0	32	3.2	23	2.7	12	3.4	18	2.2	10	2.8
8	28	3.0	33	3.2	23	3.0	15	2.9	14	2.5	16	2.6
9	27	3.1	31	3.2	23	3.2	18	2.8	16	2.6	10	2.8
10	29	3.4	30	3.2	23	2.9	21	2.8	17	2.5	12	2.5
Avg. 26.7 3.1 34 3.2 23.7 3.0							17.2 3.0 16.9 2.6 11.9 2.7					

TABLE I: Comparison of search configurations. R: Random Search, SP: Search with strict prioritization, PP: Search with probabilistic prioritization. N_f : Number of hazardous behaviors found, d_e : Avg. error distance of hazardous behaviors found. In all cases, $\alpha = 0.6$ and $N_{NMO} = 8$ was chosen.

required (e.g., what safety measures are used or how they are configured). This information is contained *implicitly* in the simulation, but there is no need to formalize it *explicitly* (as it would be the case e.g. in rule-based methods). This improves scalability to complex systems.

The main challenge is the large search space of possible worker behaviors. Our test example indicates that the proposed search techniques can effectively cope with this, although more extensive tests in different scenarios are needed to confirm the promising initial results. Note that our approach is limited in the sense that it can only *falsify* (i.e. find cases of safety constraints being violated) but cannot provide a safety guarantee. As with all simulation-based techniques, success depends on the model quality. With faulty or too simplistic models, the simulation might find hazards that do not exist in reality, or, more concerning, real-world hazards might not manifest themselves in simulation. For instance, our example uses a relatively simplistic worker model where certain potentially hazardous behaviors (e.g. deviations in the walking path) are not captured. Future work will therefore focus on the integration of more sophisticated digital human models and on the use of advanced search algorithms. It would also be interesting to test performance against an exhaustive search rather than a random search as a baseline. In the present paper this was not yet done due to the extensive computation time required.

Finally, we emphasize that the proposed method is an *addition* to existing ones and not a replacement. In the long term, simulation-based testing could be used in conjunction with formal verification to provide more detailed analysis of cases where formal verification indicates potential hazards.

ACKNOWLEDGMENT

We thank Tamim Asfour for his support and constructive discussions.

REFERENCES

- [1] “ISO TS 15066:2016 Robots and robotic devices - Collaborative robots,” International Organization for Standardization, 2016.

- [2] "ISO 10218: Robots and robotic devices - Safety requirements for industrial robots - Part 2: Robot systems and integration," International Organization for Standardization, 2011.
- [3] "ISO 12100:2010 Safety of machinery - General principles for design - Risk assessment and risk reduction," International Organization for Standardization, 2010.
- [4] M. Askarpour, D. Mandrioli, M. Rossi, and F. Vicentini, "SAFER-HRC: Safety analysis through formal verification in human-robot collaboration," in *35th International Conference SAFECOMP*, 2016.
- [5] F. Vicentini, M. Askarpour, M. G. Rossi, and D. Mandrioli, "Safety assessment of collaborative robotics through automated formal verification," *IEEE Transactions on Robotics*, vol. 36, no. 1, 2019.
- [6] R. Awad, M. Fechter, and J. van Heerden, "Integrated risk assessment and safety consideration during design of HRC workplaces," in *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2017.
- [7] Wigand, Krüger, Wrede, Stuke, and Edler, "Report on the Application of the COVR Toolkit and its Protocols for the Certification of Cooperative Robot Workstations," 2020.
- [8] D. Araiza-Illan, D. Western, A. G. Pipe, and K. Eder, "Systematic and realistic testing in simulation of control code for robots in collaborative human-robot interactions," in *Annual Conference Towards Autonomous Robotic Systems*. Springer, 2016.
- [9] T. Huck, C. Ledermann, and T. Kröger, "Simulation-based testing for early safety-validation of robot systems," in *2020 IEEE Symposium on Product Compliance Engineering*. IEEE, 2020.
- [10] L. Wang, S. Liu, H. Liu, and X. V. Wang, "Overview of human-robot collaboration in manufacturing," in *5th International Conference on the Industry 4.0 Model for Advanced Manufacturing (AMP) Belgrade, Serbia. 2020*. Springer, 2020.
- [11] "ISO 13855:2010 Safety of machinery - Positioning of safeguards with respect to the approach speeds of parts of the human body," International Organization for Standardization, 2010.
- [12] "ISO/TR 14121-2:2012 Safety of machinery — Risk assessment — Part 2: Practical guidance and examples of methods," International Organization for Standardization, 2012.
- [13] T. P. Huck, N. Münch, L. Hornung, C. Ledermann, and C. Wurl, "Risk assessment tools for industrial human-robot collaboration: Novel approaches and practical needs," *Safety Science*, vol. 141, 2021.
- [14] N. Leveson, *Engineering a safer world: Systems thinking applied to safety*. MIT Press, 2011.
- [15] J. Guiochet, Q. A. Do Hoang, M. Kaaniche, and D. Powell, "Model-based safety analysis of human-robot interactions: The MIRAS walking assistance robot," in *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*. IEEE, 2013.
- [16] J. A. Marvel, J. Falco, and I. Marstio, "Characterizing task-based human-robot collaboration safety in manufacturing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 2, 2014.
- [17] "IEC 61882:2016: Hazard and operability studies (HAZOP studies) - application guide," International Electrotechnical Commission, 2016.
- [18] M. Askarpour, D. Mandrioli, M. Rossi, and F. Vicentini, "Modeling operator behaviour in the safety analysis of collaborative robotic applications," in *36th International Conference SAFECOMP*, 2017.
- [19] M. Askarpour, M. Rossi, and O. Tiryakiler, "Co-simulation of human-robot collaboration: From temporal logic to 3D simulation," in *1st Workshop on Agents and Robots for Reliable Engineered Autonomy, AREA 2020*, vol. 319. Open Publishing Association, 2020, pp. 1–8.
- [20] M. Webster, C. Dixon, M. Fisher, M. Salem, J. Saunders, K. Koay, and K. Dautenhahn, "Formal verification of an autonomous personal robotic assistant," *Formal Verification and Modeling in Human-Machine Systems*, 2014.
- [21] B. J. Choi, J. Park, and C. H. Park, "Formal verification for human-robot interaction in medical environments," in *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, 2021, pp. 181–185.
- [22] M. Proetzsch, K. Berns, T. Schuele, and K. Schneider, "Formal verification of safety behaviours of the outdoor robot RAVON," in *ICINCO-RA (1)*, 2007.
- [23] B. R. Ferrer, B. Ahmad, A. Lobov, D. A. Vera, J. L. M. Lastra, and R. Harrison, "An approach for knowledge-driven product, process and resource mappings for assembly automation," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2015.
- [24] R. Lee, O. J. Mengshoel, A. Saksena, R. W. Gardner, D. Genin, J. Silbermann, M. Owen, and M. J. Kochenderfer, "Adaptive stress testing: Finding likely failure events with reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 69, 2020.
- [25] A. Corso, R. Moss, M. Koren, R. Lee, and M. Kochenderfer, "A survey of algorithms for black-box safety validation of cyber-physical systems," *Journal of Artificial Intelligence Research*, vol. 72, 2021.
- [26] R. Alexander, H. R. Hawkins, and A. J. Rae, "Situation coverage—a coverage criterion for testing autonomous robots," 2015. [Online]. Available: <https://eprints.whiterose.ac.uk/88736/>
- [27] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, "Adaptive Stress Testing with Reward Augmentation for Autonomous Vehicle Validation," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019.
- [28] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019.
- [29] A. A. Babikian, "Automated generation of test scenario models for the system-level safety assurance of autonomous vehicles," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 2020.
- [30] W. Ding, B. Chen, M. Xu, and D. Zhao, "Learning to collide: An adaptive safety-critical scenarios generating method," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.
- [31] D. Karunakaran, S. Worrall, and E. Nebot, "Efficient falsification approach for autonomous vehicle validation using a parameter optimisation technique based on reinforcement learning," *arXiv preprint arXiv:2011.07699*, 2020.
- [32] B. Groza, "Traffic models with adversarial vehicle behaviour," *arXiv preprint arXiv:1701.07666*, 2017.
- [33] R. Lee, M. J. Kochenderfer, O. J. Mengshoel, G. P. Brat, and M. P. Owen, "Adaptive stress testing of airborne collision avoidance systems," in *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*. IEEE, 2015.
- [34] R. Alexander and T. Kelly, "Hazard analysis through simulation for systems of systems," in *Proceedings of the 24th International Systems Safety Conference*, 2006.
- [35] G. Juez Uriagereka, E. Amparan, C. Martinez Martinez, J. Martinez, A. Ibanez, M. Morelli, A. Radermacher, and H. Espinoza, "Design-time safety assessment of robotic systems using fault injection simulation in a model-driven approach," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2019.
- [36] S. Ghosh, F. Berkenkamp, G. Ranade, S. Qadeer, and A. Kapoor, "Verifying controllers against adversarial examples with bayesian optimization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.
- [37] T. Huck, C. Ledermann, and T. Kröger, "Virtual adversarial humans finding hazards in robot workplaces," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.
- [38] H. Andersson and D. Kara, "Falsification of robot safety systems using deep reinforcement learning," Master's thesis, Chalmers University of Technology, 2021.
- [39] M. Camilli, M. Felderer, A. Giusti, D. T. Matt, A. Perini, B. Russo, and A. Susi, "Risk-Driven Compliance Assurance for Collaborative AI Systems: A Vision Paper," in *REFSQ*, 2021.
- [40] B. M. J.-R. Lesage and R. Alexander, "SASSI: Safety Analysis using Simulation-based Situation Coverage for Cobot Systems," in *Proceedings of SafeComp 2021*. York, 2021.
- [41] A. F. Bobick, "Movement, activity and action: the role of knowledge in the perception of motion," *Philosophical Transactions of the Royal Society of London (Biological Sciences)*, vol. 352, no. 1358, 1997.
- [42] E. Hollnagel, "The phenotype of erroneous actions," *International Journal of Man-Machine Studies*, vol. 39, no. 1, pp. 1–32, 1993.
- [43] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Communications of the ACM*, vol. 7, no. 3, 1964.
- [44] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.