



Deposited via The University of York.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/id/eprint/165173/>

Version: Accepted Version

Proceedings Paper:

Gleirscher, Mario and Calinescu, Radu (2020) Safety Controller Synthesis for Collaborative Robots. In: Proceedings of the 25th International Conference on Engineering of Complex Computer Systems (ICECCS). Engineering of Complex Computer Systems, 28-31 Oct 2020 , SGP.

Reuse

Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Safety Controller Synthesis for Collaborative Robots

Mario Gleirscher and Radu Calinescu

Department of Computer Science, University of York, York, UK
mario.gleirscher,radu.calinescu@york.ac.uk

Abstract—In human-robot collaboration (HRC), software-based automatic safety controllers (ASCs) are used in various forms (e.g. shutdown mechanisms, emergency brakes, interlocks) to improve operational safety. Complex robotic tasks and increasingly close human-robot interaction pose new challenges to ASC developers and certification authorities. Key among these challenges is the need to assure the correctness of ASCs under reasonably weak assumptions. To address this need, we introduce and evaluate a tool-supported ASC synthesis method for HRC in manufacturing. Our synthesis approach is informed by the manufacturing process, risk analysis, and regulations. A synthesised ASC is formally verified against correctness criteria and selected from a design space of feasible controllers according to a set of optimality criteria. Such an ASC can detect the occurrence of hazards, move the process into a safe state, and, in certain circumstances, return the process to an operational state from which it can resume its original task.

Index Terms—Controller synthesis, human-robot collaboration, software engineering, probabilistic model checking.

I. INTRODUCTION

An effective collaboration between industrial robot systems (IRSS) and humans [1], [2] can leverage their complementary skills, but is difficult to achieve because of uncontrolled hazards and unexploited sensing, tracking, and safety measures [3]. Such hazards have been studied since the 1970s, resulting in elaborate risk taxonomies based on workspaces, tasks, and human body regions [2], [4]–[10]. The majority are *impact hazards* (e.g. unexpected movement, reach beyond area, dangerous workpieces, hazardous manipulation), *trapping hazards* (e.g. operator in cage), and *failing equipment*.

Addressing these hazards involves the examination of each *mode of operation* (e.g. normal, maintenance) for its hazardous behaviour, and the use of automatic safety controllers (ASCs) to trigger mode-specific *safety measures* [2]. Malfunction diagnostics (e.g. fault detection, wear-out monitoring) can further inform the ASCs. Tab. I shows a variety of measures [3] to *prevent or mitigate* hazards by reducing the *probability of their occurrence* and the *severity of their consequences*.

The standardisation of safety requirements for IRSS [4] culminated in ANSI/RIA R15.06, ISO 10218 [11], 13482, and 15066. According to ISO 10218, an IRS comprises a *robot arm*, a *robot controller*, an *end-effector*, and a *workpiece* (see, e.g. Fig. 2a below). In *collaborative operation*, the operator and the IRS (called a *cobot* [12]) can occupy the *collaborative workspace* (i.e., a subset of the *safeguarded workspace*) simultaneously while the IRS is performing *tasks* [13]. Based on that, ISO 15066 recommends four *safety modes*, described and combined with work layouts in [8], [14]: *safety-rated monitored stop* (powered but no simultaneous activity of

robot and operator in shared workspace), *hand-guided operation* (zero-gravity control, guided by an operator, no actuation without operator input), *speed & separation monitoring* (speed continuously adapted to distance of robot and operator), and *power & force limiting* (reduced impact on the human body, a robot’s power and applied forces are limited).

Since the 1980s, tele-programming and simulation have led to a reduction of hazard exposure. However, guarding arrangements interfere with manufacturing processes and mobile robots. Complex tasks require continuous and close human-robot interaction (e.g. mutual take-over of tasks), mutual clarification of intent, and trading off risk [14], [15]. Robot movements need to be predictable and impacts on the human body need to be attenuated (e.g. speed & separation monitoring requires stereo vision and laser scanners to distinguish safety zones). Engineers need to consider a variety of complex failure modes. This situation implies *requirements and design spaces* for ASCs, so engineers want to answer questions such as:

- Which ASC design minimises the probability of incidents in presence of human and sensor errors?
- Which design minimises nuisance to the human, maximises productivity, etc. while maintaining safety?
- Does a selected controller correctly handle hazards when detected and return the system to a useful safe state?

Contributions: We introduce a tool-supported method for the synthesis of discrete-event ASCs that meet safety requirements and optimise process performance for human-robot *cooperation* (alternative use of shared workspace) and *collaboration* (simultaneous use of shared workspace, with close interaction) [8], [16]. We model the manufacturing process and its safety analysis as a Markov decision process (MDP) and select a correct-by-construction ASC from a set of MDP policies. We extend our notion of *risk structures* [17],

Table I: IRS safety measures by stage of causal chain

Stage	Type of Measure	Examples
Hazard prevention	1. Safeguard/barrier	Fence, interlock.
	2. IT safety	Verified safety controller.
	3. IT security	Security-verified (safety) controller.
Hazard mitigation & accident prevention	4. Reliability	Fault-tolerant scene interpretation.
	5. Workspace intrusion detection	Speed & separation monitoring, safety-rated monitored stop.
	6. Shift of control	hand-guided operation.
Accident mitigation (alleviation)	7. Power & force limitation	Low weight parts, flexible surfaces; variable impedance, touch-sensitive, & force-feedback control.
	8. System halt	Emergency stop, dead-man’s switch.

[18] and our tool YAP [19]. This simplifies the modelling of activities and actors, critical events (CEs, e.g. hazards), mitigations (e.g. safety mode changes) and reward structures for risk optimisation; and automates the translation of risk structures into MDPs. Our approach facilitates the verification of safety of the MDP and of *probabilistic reach-avoid* properties of a selected policy. A verified ASC detects hazards and controls their mitigation by (i) the execution of a safety function, (ii) a transition to a safer mode, or (iii) a transition to a safer activity.

Overview: Sec. II discusses related work, Sec. III introduces our case study as a running example, and Sec. IV provides the theoretical background. We describe and evaluate the ASC synthesis method in Sec. V and Sec. VI, respectively, and we conclude with a short summary in Sec. VII.

II. RELATED WORK

To the best of our knowledge, our method is the first end-to-end approach to synthesising ASCs for handling multiple risks in HRC for manufacturing processes.

Askarpour et al. [20] discuss a discrete-event formalisation of a work cell in the linear-time temporal language TRIO. Actions are specified as *pre/inv/post*-triples (with a safety *invariant*) for contract-based reasoning with the SAT solver Zot. In contrast, we use a probabilistic guarded command language (pGCL) [21], separating action modelling from property specification. Beyond counterexamples for model repair, our approach yields an executable policy. While their use of a priority parameter helps to abstract from unnecessary state variables, we propose guards to implement flexible individual action orderings. Moreover, violations of *inv* lead to pausing the cell whereas our approach can deal with multiple mitigation options offering a variety of safety responses.

For generic robot applications, Orlandini et al. [22] employ the action language PDDL and timed game automata for controller synthesis. Verification (i.e., finding winning strategies) of reach-avoid properties of type $\mathbf{A}(safe \ \mathbf{U} \ goal)$ is done by the model checker UPPAAL-TIGA. While game solving could enhance our verification, our method focuses on risk modelling for safely optimised HRC performance. Cesta et al. [23] present an approach to synthesise controllers (i.e., plans) for HRC applications using a timeline-based PDDL planner. Their focus is on task planning and scheduling rather than on risk modelling for verified synthesis of ASCs.

Heinzmann and Zelinsky [24] propose a power & force limiting mode always active during an HRC activity described as a discrete-event controller. Long et al. [25] propose a speed & separation monitoring scheme with nominal (max. velocity), reduced (speed limiting), and passive (hand-guided operation) safety modes. While these authors do not aim at synthesis or task modelling, their elaborate safety modes may serve as a target platform to our multi-risk synthesis approach.

III. RUNNING EXAMPLE: MANUFACTURING COBOTS

Fig. 1 shows an IRS *manufacturing cell* at a UK company (with the pictures anonymised for confidentiality reasons) and replicated in a testbed at the University of Sheffield (Fig. 1c).

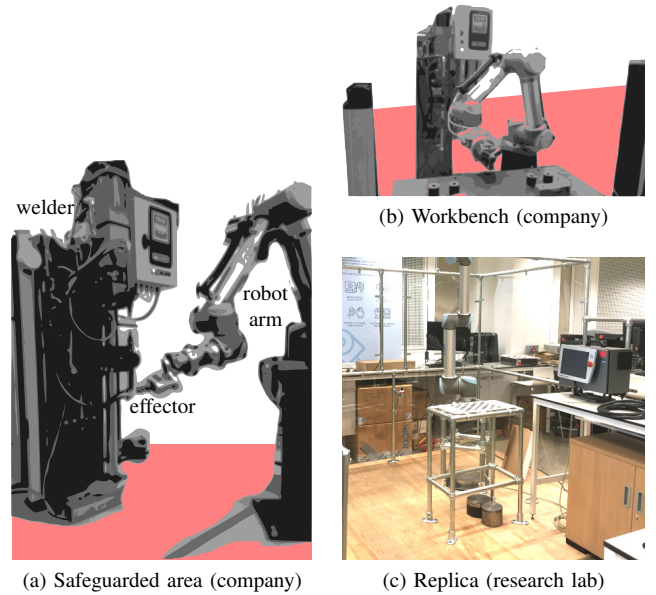


Figure 1: Actual (a, b) and replicated (c) cobot setting

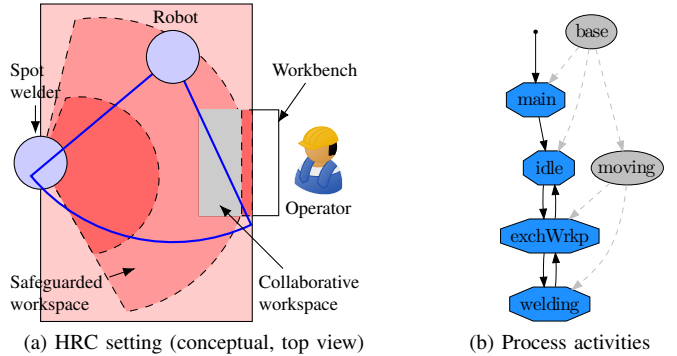


Figure 2: Conceptual setting (a) and activities in the manufacturing process (b) performed by the operator, the robot, and the welder (in blue), classified by the activity groups *moving* and *base* (in gray)

The corresponding process (call it \mathcal{P}) consists of activities (Fig. 2b) collaboratively repeated by an *operator*, a stationary *robotic arm*, and a *spot welder* (Fig. 2a). Previous safety analysis (i.e., hazard identification, risk assessment, requirements derivation) resulted in two sensors (i.e., a range finder in Fig. 1a and a light barrier in Fig. 1b, indicated in red) triggering an emergency stop if a person approaches the welder or enters the workbench while the robot or welder are active. Tab. II shows our partial safety analysis of the cell following the guidance in Sec. I. The right column specifies safety goals against each accident and controller requirements (e.g. mode-switch requirements) handling each latent cause in the left column, and indicating how the hazard is to be removed.

IV. PRELIMINARIES

Our method employs MDPs as a formal model of \mathcal{P} , and MDP policies as the design space for controller synthesis.

Definition 1. *Markov decision process (MDP).* Given all distributions $Dist(\alpha_{\mathcal{P}})$ over an action alphabet $\alpha_{\mathcal{P}}$ of a process \mathcal{P} , an MDP is a tuple $\mathcal{M} = (S, s_0, \alpha_{\mathcal{P}}, \delta_{\mathcal{P}}, L)$ with a

Table II: Our partial safety analysis of the manufacturing cell referring to the measures recommended in ISO 15066

Id	Critical Event (Risk Factor)	Safety Requirement
	Accident (to be <i>prevented</i> or <i>alleviated</i>)	Safety Goal
<i>RC</i>	<u>R</u> obot arm harshly <u>C</u> ollides with operator	The robot shall <i>avoid</i> harsh active collisions with the operator.
<i>WS</i>	<u>W</u> elding <u>S</u> park cause operator injuries	The welding process shall <i>reduce</i> sparks injuring the operator.
<i>RT</i>	<u>R</u> obot arm <u>T</u> ouches the operator	The robot shall <i>avoid</i> active contact with the operator.
	Latent Cause (to be <i>mitigated</i> timely) [†]	Controller Requirement
<i>HRW</i>	<u>H</u> uman operator and <u>R</u> obot use <u>W</u> orkbench at the same time	(m) The robot shall perform a <i>safety-rated monitored stop</i> and (r) resume <i>normal operation</i> after the <i>operator</i> has left the <i>shared workbench</i> .
<i>HW</i>	<u>H</u> uman operator is entering the <u>W</u> orkbench while the robot is away from the bench	(m) If the robot moves a workpiece to the bench then it shall switch to <i>power & force limiting mode</i> and (r) resume <i>normal operation</i> after the <i>operator</i> has left the <i>workbench</i> .
<i>HS</i>	<u>H</u> uman operator has entered the <u>S</u> afeguarded area while robot moving or welder active	(m) The <i>welder</i> shall be <i>switched off</i> , the <i>robot</i> to <i>speed & separation monitoring</i> . (r) Both shall resume normal mode after the operator has left and acknowledged the notification.
<i>HC</i>	<u>H</u> uman operator is <u>C</u> lose to the welding spot while robot working and welder active	(m) The <i>welder</i> shall be <i>switched off</i> , the <i>robot</i> to <i>safety-rated monitored stop</i> . (r) Both shall resume <i>normal or idle mode with a reset procedure</i> after the operator has left.

[†] m...mitigation requirement, r...resumption requirement

set S of states, an initial state $s_0 \in S$, a probabilistic transition function $\delta_{\mathcal{P}}: S \times \alpha_{\mathcal{P}} \rightarrow \text{Dist}(\alpha_{\mathcal{P}})$, and a map $L: S \rightarrow 2^{AP}$ labelling S with atomic propositions AP [21].

Given a map $A: S \rightarrow 2^{\alpha_{\mathcal{P}}}$, $|A(s)| > 1$ signifies non-deterministic choice in s . Its resolution for S forms a *policy*.

Definition 2. Memoryless Policy. A memoryless policy is a map $\pi: S \rightarrow \text{Dist}(\alpha_{\mathcal{P}})$ s.t. $\pi(s)(a) > 0 \Rightarrow a \in A(s)$. π is deterministic if $\forall s \in S \exists a \in A(s): \pi(s)(a) = 1 \wedge \forall a' \in \alpha_{\mathcal{P}} \setminus \{a\}: \pi(s)(a') = 0$.

The following discussion is restricted to deterministic memoryless policies. Let $\Pi_{\mathcal{M}}$ be the set of all such policies for \mathcal{M} . Then, *action rewards* defined by a map $r_{action}^q: S \times \alpha_{\mathcal{P}} \rightarrow \mathbb{R}_{\geq 0}$ allow the assessment of $\Pi_{\mathcal{M}}$ based on a quantity q .

Verification of \mathcal{M} is based on probabilistic computation tree logic (PCTL) whose properties over AP are formed by

$$\phi ::= \top \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \mathbf{E}\phi \mid \mathbf{A}\phi \quad \text{and} \quad \varphi ::= \mathbf{X}\phi \mid \phi \mathbf{U}\phi$$

with $a \in AP$; an optional bound $b \in \mathbb{N}_+$ for $\mathbf{U}^{\sim b}$ with $\sim \in \{<, \leq, =, \geq\}$; the quantification operators $\mathbf{P}_{\sim b=?} \varphi$ to verify (or with $=?$, to quantify) probabilities, $\mathbf{S}_{\sim b=?}[a]$ to determine long-run probabilities. The PCTL extension $\mathbf{R}_{\sim b[\min]\max=?}^q[\mathbf{F}\phi \mid \mathbf{C}^{\sim b}]$ calculates reachability and accumulative action rewards. We use the abbreviations $\mathbf{F}\phi \equiv \top \mathbf{U}\phi$, $\mathbf{G}\phi \equiv \neg \mathbf{F}\neg\phi$, and $\phi \mathbf{W}\psi \equiv \phi \mathbf{U}\psi \vee \mathbf{G}\phi$. For sake of brevity, consider the treatment of PCTL in [21], [26].

The concise construction of $\delta_{\mathcal{P}}$, the behaviour of \mathcal{P} , can be facilitated by using a flavour of pGCL (e.g. as implemented in PRISM [21], [27]). Guarded commands are of the form $[\alpha] \gamma \longrightarrow v$ where α is an event label and v a probabilistic update applicable to $s \in S$ only if $s \models \gamma$, where γ is an expression in the propositional fragment of PCTL.¹ Generally, $v ::= \pi_1: v_1 + \dots + \pi_n: v_n$ with $\sum_{i \in 1..n} \pi_i = 1$ and assignments v_i to state variables of type \mathbb{B} , \mathbb{N} , or \mathbb{R} . Let $[\phi]_S$ denote the largest subset of S fulfilling the state predicate ϕ .

For *safety analysis*, we view the cell in Sec. III as a *process* \mathcal{P} , monitored and influenced by an ASC to mitigate *hazards* and prevent *accidents*. An *accident* $a \in S$ is an undesired

¹We use \longrightarrow to separate guard and update expressions and \rightarrow both for logical implication and the definition of mappings.

consequence reachable from a set $\Xi \subset S$ forming the *causes* of a . The fraction of a cause $c \in \Xi$ not related to the operator is called a *hazard* h [28], [29]. We call c *latent*² if there are sufficient resources (e.g. time for removing h by transition to $s \notin \Xi$) to prevent the accident. h also refers to states in $S \setminus \Xi$ being critical because certain events (e.g. an operator action) cause a transition to Ξ , and possibly a , if h stays active, further conditions hold, and no safety measures are put in place timely.

Risk modelling can be facilitated by specifying risk factors and combining them into *risk structures* [18]. A *risk factor* f is a labelled transition system (LTS) modelling the life cycle of a critical event (i.e., hazard, cause, accident). f has the basic phases *inactive* (0^f), *active* (f), and *mitigated* (\bar{f}). Transitions between these phases signify *endangerment* events (e) and *mitigation* (m) and *resumption* (m_r) actions. A risk structure from a factor set F (e.g. column **Id** in Tab. II) operates over a *risk (state) space* $R(F) = \times_{f \in F} Ph_f$ with $Ph_f = \{0^f, f, \bar{f}\}$.

V. APPROACH: SAFETY CONTROLLER SYNTHESIS

Fig. 3 shows steps and artefacts of the proposed method. We use pGCL to create an action model 1 of the cobot. We transcribe the results of safety analysis 2 and mitigation design 3 into factor LTSs. These LTSs are used by YAP [19] to extend the action model by another set of commands representing the ASC design space. We develop the factor notion in Sec. V-B and Fig. 4 as part of our contribution. Factors can include probabilistic commands but may focus on risk possibility without quantifying uncertainty. Both command sets, cobot and ASC, are combined. The MDP semantics of this combination allows a model checker (e.g. PRISM) to verify PCTL properties and synthesise policies 4. The ability of MDPs to express nondeterminism supports the modelling of alternative mitigations available to the ASC. Optimal policy synthesis produces a discrete-time Markov chain (DTMC) containing the discrete-event ASC. These steps are detailed in the following and illustrated with a running example.

A. 1 Modelling the (Manufacturing) Process

Activities in \mathcal{P} (Fig. 2b) are structured by sets of guarded commands. We distinguish *actions* of controllable actors (e.g.

²As opposed to *immediate* causes with limited or no risk handling controls.

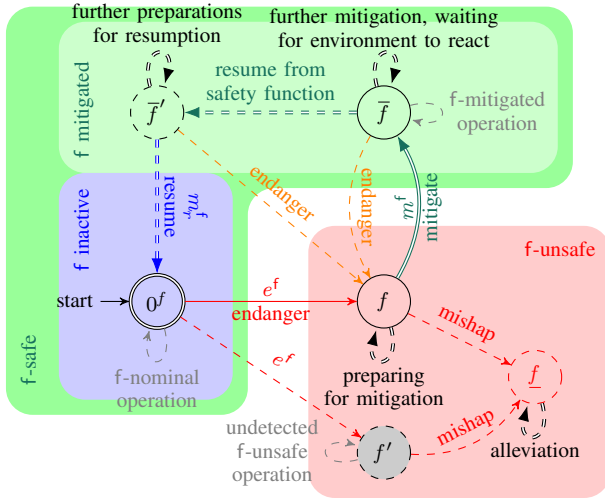


Figure 4: Phases and actions of a risk factor f . \Rightarrow ... multiple optional actions considered, --- ... minimum amount of information to be provided for a risk factor, - - ... optional modelling aspects.

Next, the HazardModel section lists critical events relevant to welding, the two mishaps RC and RT and the latent cause HC (cf. Tab. II). One can hypothesise high-level relationships between critical events using constraints. E.g. RC requiresNOF (2|HRW, HS, HC|2) expresses the assumption that exactly two of the listed events have to have occurred before RC can occur. Such relationships are typically identified during preliminary hazard operability studies (HazOp), system FMEA, or system FTA.

Furthermore, HC is specified by (a) an informal description, (b) a guard describing its activation HC , (c) *mis* (i.e., an action, e.g. of the operator, with the mishap \underline{HC} as a bad outcome if HC is undetected or not mitigated timely), (d) *prob* (i.e., the probability of \underline{HC} under these conditions), and (e) *sev*, quantifying the severity of the best, average, or worst expected consequences from \underline{HC} .

Probabilistic choice in \mathcal{M} can be used to model several uncertainties. Informed by FTA and FMEA, one can consider sensor and actuator faults. In our example, the range finder as the detector of e^{HC} fails by 5% when the operator enters the cell. Informed by HazOp, human errors can be modelled similarly. In our example, with a 10% chance, the operator enters the cell, knowing that robotArm and welder are active. Moreover, one can model the probability of occurrence of a mishap under the condition of an active hazard. In our example, with a 20% chance, \underline{HC} may follow HC' (i.e., HC remains undetected because of the aforementioned sensor fault) or HC (i.e., the ASC is not reacting timely). Such probabilities should, if possible, be inferred from observations, experiments, accident statistics, and product data sheets.

C. 3 Designing Mitigation and Resumption Options

The capabilities of actors in \mathcal{P} determine the controllability of critical events. We found three techniques useful in designing mitigations and resumptions: *action filters* (i.e., safety modes, cf. Sec. I), *activity changes* (e.g. change from welding to off), and *safety functions* (e.g. operator notification). Recall that mitigations and resumptions are actions (i.e., transition labels in a factor LTS). Accordingly, Listing 4 exem-

Listing 4: YAP action specifications for the risk factor HC

```

mode HCdet desc "range finder"
  guard "hACT_WELDING & rngDet=close"
  embodiedBy rngDet;
mode HCmit desc "protective emergency stop of robotic system"
  event stop
  update "(notif=leaveArea)"
  target (act=off, safmod=stopped)
  disruption=10
  nuisance=2
  effort=1;
mode HCres desc "resumption from emergency stop"
  event resume
  guard "notif=leaveArea & !hST_HOinSGA"
  update "(notif=ok)"
  target (act=exchWrkp, safmod=normal);

```

Listing 5: Monitor predicates for HC generated for the PRISM model

```

// HC:monitor "(H)uman (C)lose to active welder and robot working"
formula RCE_HC = hACT_WELDING & hloc=atWeldSpot;
formula CE_HC = hACT_WELDING & rngDet=close;

```

plifies the actions referred to in Listing 3. Here, the following parameters drive the design space of an ASC: (a) a detectedBy reference (i.e., associating the guard with a sensor predicate), (b) a mitigatedBy reference to one or more mitigation options, and (c) a resumedBy reference to one or more resumption options. For this approach, we extended YAP's input language to develop these actions into guarded commands.

Example 4. As an example for (b), in Listing 4, the action HCmit of type SHUTDOWN (i) synchronises with the robotArm and welder on the event stop, (ii) update models a safety function, issuing a notification to the operator to leave the safeguarded area, and (iii) target switches the manufacturing cell to the activity off and to the safety mode stopped, all triggered by the range finder.

Indicated in Fig. 4, HCmit models one option for m^{HC} . One can distinguish several such options by quantities such as *disruption* of the manufacturing process, *nuisance* of the operator, and *effort* to be spent by the machines. In combination with processing *time* and *value* for each nominal action of \mathcal{P} , these quantities enable the evaluation and selection of optimal policies as we shall see below.

This part of the YAP model can be translated into pGCL. Endangerments are translated into commands of the form

$$[e^f] \phi_a \wedge \chi \longrightarrow f' \quad \text{and} \quad [e^f] \phi_a \wedge \zeta \longrightarrow (1-p): f + p: f'$$

with guards including a hazard condition χ and a corresponding monitoring (or sensor) predicate ζ . Constraints, such as requiresNOF in Example 3, are then used to derive part of ζ .

Example 5. Listing 5 indicates the transcription of guard and detectedBy from Listing 3 into a pair of predicates, RCE_HC describing actual states, and CE_HC signifying states monitored by the range finder, where p can denote the sensor fault probability.

Mitigations are translated into commands of the form

$$[m_t^f] \phi_t \wedge f \longrightarrow v_t' \quad \text{and} \quad [m^f] \phi_{sm',a',sf'} \wedge f \longrightarrow \bar{f}$$

with $t \in \{sm, a, sf\}$ in ϕ_t for checking permission in the current safety mode, activity, and state of safety functions, and in $v_{t'}$ for hazard removal by switching into a safer activity a' , a safer mode sm' , and by applying the safety function sf . These updates are checked by $\phi_{sm',a',sf'}$ to be able to proceed to \bar{f} . *Resumptions* are translated into commands of the form

$$[m_{r,t}^f] \phi_t \wedge \rho_{\bar{f}} \longrightarrow v_{t'} \quad \text{and} \quad [m_r^f] \phi_{sm',a',sf'} \wedge \rho_{\bar{f}} \longrightarrow 0^f$$

where ϕ_t guards the resumption based on the safety mode and function in place, $\rho_{\bar{f}}$ restricts permission to risk states $\llbracket \rho_{\bar{f}} \rrbracket_R \subset R(F)$ (Sec. IV, Fig. 4) with f mitigated; and $v_{t'}$ inverts the safety function (sf^{-1}), relaxes to the safety mode sm' , and returns to an, ideally more productive, activity a' of \mathcal{P} .

D. 4 Verified Controller Synthesis

Our approach follows a two-staged search through the ASC design space: The first stage is carried through by YAP when generating the guarded commands. The second stage is performed by a model checker (e.g. PRISM [27]) when synthesising MDP policies. For search space reduction, YAP employs *risk gradients* between safety modes and activities in the first stage. For the second stage, YAP generates reward structures for some of the quantities introduced in Sec. V-C.

1) *Guarded Command Generation*: The generation of $v_{t'}$ for mitigations and resumptions requires the choice of a safety mode and activity to switch to, depending on the current mode and activity. Given activities S_a and modes S_{sm} , two skew-diagonal *risk gradient matrices* $\mathfrak{G}^a \in \mathbb{R}^{|S_a| \times |S_a|}$ and $\mathfrak{G}^{sm} \in \mathbb{R}^{|S_{sm}| \times |S_{sm}|}$, e.g. manually crafted from safety analysis, can resolve this choice based on the following justification.

Assume $a_1, a_2 \in S_a$ vary in physical movement, force, and speed. If a_1 means more or wider movement, higher force application, or higher speed than a_2 , then a change from a_1 to a_2 will likely reduce risk. Hence, a positive gradient is assigned to $\mathfrak{G}_{a_1 a_2}^a$. Similarly, assume $m_1, m_2 \in S_{sm}$ vary \mathcal{P} 's capabilities by relaxing or restricting the range and shape of permitted actions. If m_1 relaxes capabilities more than m_2 , then a change from m_1 to m_2 will likely reduce risk. Again, we assign a positive gradient to $\mathfrak{G}_{m_1 m_2}^{sm}$. The diagonality of \mathfrak{G} provides the dual for resumptions where a negative gradient of the same amount from m_2 to m_1 is assigned to $\mathfrak{G}_{m_2 m_1}^{sm}$.

Consider an active safety mode c and a mitigation m^f with target mode t . m^f (m_r^f) changes to t only if the gradient from c to t is ≥ 0 (≤ 0). If $\mathfrak{G}_{ct}^{sm} \geq 0$, then a switch to t is included in $v_{sm'}$, otherwise $v_{sm'}$ leaves \mathcal{M} in c . We implemented this scheme for activities analogously.

Example 6. Listing 6 shows the result of applying this scheme in the generation of an ASC for the risk factor HC based on Listing 4.

Essentially, \mathfrak{G} approximates the change of risk in case of a change from one activity or safety mode to another. Using \mathfrak{G} , the majority of an ASC can be described in YAP script.

2) *MDP Verification*: This step requires establishing $\mathcal{M} \models \phi_{wf} \wedge \phi_c$ with properties expressed in PCTL (Sec. IV). ϕ_{wf} is a *well-formedness* property including the verification of, e.g. hazard occurrence and freedom of pre-final deadlocks, and

Listing 6: PRISM model fragment generated for the risk factor HC

```
// Endangerments (monitor)
[si_HCact] wact=welding & ract=exchWrkp & CE_HC & !(Hcp=act | Hcp=mis)
  -> (Hcp=act);
...
// Escalation to mishap if not mitigated (for analysis only)
[h_exitPlant] true ->
  ((! Hcp=mis & (CE_HC | RCE_HC))?0.2:0):(Hcp=mis)
  +(! Hcp=mis & (CE_HC | RCE_HC))?0.8:1):true;
// Mitigation with synchronous events
[s_HCstop] wact=welding & ract=exchWrkp & Hcp=act -> true;
...
// Change of safety modes
[si_HCmitsafmod] safmod=normal & Hcp=act -> (safmod=stopped);
...
// Execution of safety functions
[si_HCmitfun] Hcp=act & !(notif=leaveArea) -> (notif=leaveArea);
// For entering the mitigated phase
[si_HCmit] Hcp=act & (notif=leaveArea) & !CE_HC -> (Hcp=mit);
// Switching off safety functions
[si_HCresfun] Hcp=mit & !CE_HC & notif=leaveArea & !hST_HOinSGA -> (
  notif=ok);
// Resuming to a less restrictive safety mode
[si_HCressafmod] safmod=normal & Hcp=mit & Hcp=mit & (notif=ok)
  -> (safmod=normal)&(Hcp=sfd);
...
// Resuming actor's activities (via synchronisation)
[s_HCresume] Hcp=sfd & !CE_HC & (notif=ok) -> (Hcp=inact);
```

the falsification, e.g. that final states must not be initial states. ϕ_{wf} helps to simplify model debugging, decrease model size, guarantee progress, and reduce vacuity. ϕ_c specifies *safety-carrying correctness* including, e.g. ASC progress (and across cycles, liveness), particularly, that $\Pi_{\mathcal{M}}$ (i.e., the ASC design space) contains complete mitigation paths from CEs. Tab. III lists examples of ϕ_{wf} and ϕ_c to be verified of \mathcal{M} .

3) *Policy Synthesis*: The pGCL action system consisting of the two fragments, cobot and ASC, and translated into an MDP by a probabilistic model checker, represents the ASC design space ($\Pi_{\mathcal{M}}$). Choice in $\Pi_{\mathcal{M}}$ stems from commands (e.g. mitigations, resumptions) simultaneously enabled in $s \in S$, yielding multiple policies for s and from commands enabled in multiple states, giving rise to a policy for each ordering in which these commands can be chosen.

An *optimal policy* π^* , including the ASC decisions, can be selected from $\Pi_{\mathcal{M}}$ based on multiple criteria (e.g. minimum risk and nuisance, maximum productivity). For that, \mathcal{M} uses action rewards to quantify (i) *productivity*, up- and down-time of \mathcal{P} ; (ii) *factor-, mode-, and activity-based risk*; risk reduction *potential*; disruptiveness and *nuisance*; resource consumption; and *effective time* of the ASC.

Example 7. Listing 7 shows reward structure fragments for nuisance and risk generated by YAP.

Based on the action rewards, the model checker investigates all choice resolutions for this MDP that fulfil well-formedness (e.g. a reach-avoid property; see Sec. V-D2) and further PCTL constraints (e.g. a bounded reward property). The checker then identifies resolutions that are (Pareto-)optimal w.r.t. to the aforementioned criteria formulated as PCTL reward formulae. See the second property category

Listing 7: PRISM rewards for risk from HC and nuisance of HCmit

```

// Risk of occurrence of HC
rewards "risk_HC"
[r_moveToWelder] (RCE_HC | CE_HC) & !CYCLEEND : 5;
[rw_weldStep] (RCE_HC | CE_HC) & !CYCLEEND : 10;
[h_approachWeldSpot] (RCE_HC | CE_HC) & !CYCLEEND : 7;
...
[h_exitPlant] notif=leaveArea & (RCE_HC | CE_HC) & !CYCLEEND : 2;
endrewards

// Nuisance (e.g. to the human operator; per mitigation option)
rewards "nuisance"
// HC-mitigation: HCmit
[si_HCmitsafmod] (HCP=act | HCP=mit) : 2.0;
[s_HCstop] (HCP=act | HCP=mit) : 2.0;
[si_HCmitfun] (HCP=act | HCP=mit) : 2.0;
endrewards

```

in Tab. III. The existence of an optimal strategy depends on the existence of a strategy in $\Pi_{\mathcal{M}}$ that fulfils the PCTL constraints. Note that, by Definition 2, all policies considered for a particular MDP are of the same size but may vary in their distribution of choice among the involved actors.

4) *DTMC Verification*: Due to some restrictions in combining multi-objective queries and constraints in the available version of PRISM, part of the ASC verification is postponed to the synthesised policy, that is, the DTMC. This step requires establishing $\pi^* \models \phi_s$ where ϕ_s can include liveness, safety, and reliability properties (e.g. “reach-avoid” of type $\mathbf{A} \mathbf{G} \mathbf{F} \psi \wedge \mathbf{A} \mathbf{G} \neg \phi$; the probability of failure on demand of the ASC; the probability of a mishap from any hazard is below a threshold). The third and fourth categories in Tab. III list examples of properties to be verified of π^* .

Fig. 5 visualises π^* as a graph with nodes for all states reachable from s_0 , and edges for all transitions generated by $\delta_{\mathcal{P}}$ as derived from the guarded commands of \mathcal{P} . The edges form executions of \mathcal{M} from s_0 under π^* .

Example 8. Fig. 5 (left) provides a bird’s eye view of a synthesised policy. States coloured in green form the set HC-safe, i.e., states in which HC is inactive or mitigated.

Fig. 5 (right) highlights a policy fragment. Note the 5% chance of a sensor failure from state 90 leading to state 92 (i.e., the operator has approached the robotArm and welder) where HC will remain undetected and not handled. Otherwise, in state 93, HC will be mitigated after the next work step of the robotArm and welder leading to state 202. From there, the ASC mitigates to a protective stop (state 176), proceeds to state 167 (i.e., where sensors confirm that the operator left the safeguarded area), and resumes to state 178 from where the current work cycle can be finished (state 33).

VI. EVALUATION

In this section, we discuss the adequacy and efficacy of the proposed method from several viewpoints.

A. Research Questions and Evaluation Methodology

Based on the questions raised in Sec. I, we investigate the (i) scalability and performance of the approach and (ii) the effectiveness of the ASCs synthesised by it, asking:

Table III: Examples of checked properties and queried objectives

Property [†]	Description
<i>Sec. V-D2: Well-formedness ϕ_{wf} of \mathcal{M}</i>	
$v: \mathbf{E} \mathbf{F} (f \wedge \neg final)$	Can the hazard f occur during a cycle of \mathcal{P} ?
$f: \mathbf{E} \mathbf{F} (deadlock \wedge \neg final)$	Are all deadlock states final? Does \mathcal{P} deadlock early?
$f: \mathbf{A} \mathbf{F} f$	Is f inevitable?
$f: \neg \exists s \in S: final \wedge init$	Are there initial states that are also final states?
$v: \mathbf{E} \mathbf{F} final$	Can \mathcal{P} finish a production cycle?
<i>Sec. V-D3: Querying for a (Pareto-)optimal ASC π^*</i>	
$\mathbf{R}_{max=?}^{pot} [C] \wedge \mathbf{R}_{max=?}^{eff} [C]$	Assuming an adversarial environment, select π that maximally utilises the ASC.
$\mathbf{R}_{max=?}^{prod} [C] \wedge \mathbf{R}_{\leq s}^{sev} [C] \wedge \mathbf{R}_{\leq r}^{risk} [C]$	Select ASC that maximises productivity constrained by risk level r and expected severity s .
$\mathbf{R}_{max=?}^{prod} [C] \wedge \mathbf{R}_{\leq s}^{sev} [C]$	Select ASC that maximises productivity constrained by exposure p to severe injuries.
<i>Sec. V-D4: Cycle-bounded correctn. ϕ_c of policy π (or pol. space $\Pi_{\mathcal{M}}$)</i>	
$v: \mathbf{A} \mathbf{F} (\zeta \rightarrow \mathbf{A} \mathbf{X} f)$	Does the ASC on all paths immediately detect the hazard χ ?
$v: \mathbf{A} \mathbf{F} (f \rightarrow (\mathbf{A} \mathbf{F} \bar{f} \rightarrow (\mathbf{A} \mathbf{F} 0^f)))$	Does the ASC lively handle hazard f in all situations?
$v: \mathbf{E} \mathbf{F} (f \wedge \mathbf{F} final)$	Does the ASC resume \mathcal{P} so it can finish its cycle after f has occurred?
$v: \mathbf{P}_{>p} [\mathbf{G} \neg mishap]$	Is the probability of mishap freedom greater than p ?
<i>Sec. V-D4: Reliability ϕ_r of a selected ASC π^*</i>	
$v: \mathbf{S}_{<p} mishap$	Is the steady-state (long-run) probability of any mishap f below p ?

[†] *deadlock* ... state with no commands enabled, *final* ... end of manufacturing cycle, *init* ... initial state of a manufacturing cycle, *mishap* ... mishap state, p ... probability bound, v ... to be verified, f ... to be falsified, *prod* ... productivity, *sev* ... severity, *eff* ... ASC effectiveness, *risk* ... risk level, *pot* ... risk reduction potential

RQ1 How well can the approach deal with multiple hazards and mitigation and resumption options? What are the resulting model sizes and analysis times?

RQ2 What is the likelihood of incident/accident-free operation under the control of the synthesised ASCs?

RQ3 Which process overheads are to be expected of an ASC implementation?

For **RQ1**, which assesses the scalability of the approach, we consider as inputs and parameters a YAP risk model and an MDP model (given in PRISM’s flavour of pGCL) of the cell (with YAP template placeholders), a single initial state of these models where all actors are in the activity `off` and no hazard is active. Accordingly, we prepare and analyse multiple increments of the risk model, each adding one critical event, mitigation options, and constraints to the model.

For **RQ2**, let $\Xi \subset S$ be the set of non-accident F -unsafe states, i.e., states labelled with at least one CE, describing the abstract state where any critical event has at least been sensed by the ASC (e.g. CE_HC with its handling not yet started, i.e., 0^{HC}). For MDPs, we evaluate accident freedom with

$$\mathbf{P}_{\neg A} \equiv f_{s \in \Xi} \mathbf{P}_{\min=?}^s [-mishap \mathbf{W} safe] \quad (1)$$

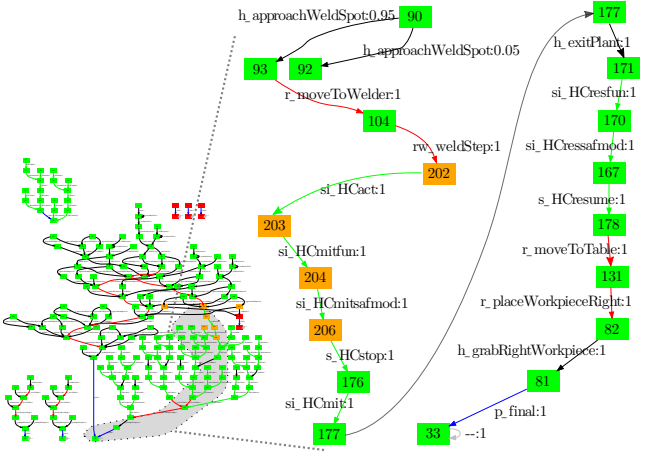


Figure 5: The left shows a bird’s-eye view of the policy synthesised for the query $\mathbf{R}_{\max=?}^{pot}[\mathbf{C}] \wedge \mathbf{R}_{\max=?}^{eff}[\mathbf{C}]$. The gray fragment is magnified on the right and split at state 177 for layout efficiency. Nodes are the states reachable in \mathcal{M} from s_0 , including HC-safe states (green), HC-unsafe states (orange), and mishap states (red). Edges indicate *robotArm* and *welder* actions (red), actions of the operator (black), the ASC (green), and cycle termination (blue).

where $f \in \{\min, \text{mean}, \max\}$. For Ξ , Formula (1) requires the ASC to minimise the probability of mishaps until an F -safe state (i.e., $S \setminus \Xi$) is reached. In Tab. IV, $[\mu]$ denotes the triple comprising min, the arithmetic mean μ , and max. \mathbf{P}_{-A} aggregates these three probabilities over Ξ .

Next, we synthesise policies for each of the MDP increments for the three optimisation queries

$$\mathbf{R}_{\max=?}^{pot}[\mathbf{C}] \wedge \mathbf{P}_{\max=?}[\mathbf{F} \text{ final}_t], \quad (\text{a})$$

$$\mathbf{R}_{\max=?}^{prod}[\mathbf{C}] \wedge \mathbf{P}_{\max=?}[\mathbf{F} \text{ final}_t], \text{ and} \quad (\text{b})$$

$$\mathbf{R}_{\max=?}^{eff}[\mathbf{C}] \wedge \mathbf{R}_{\max=?}^{nuis}[\mathbf{C}]. \quad (\text{c})$$

where $\text{final}_t = \{s \in S \mid s \in \text{final} \wedge \text{all tasks finished}\}$. In the spirit of negative testing, Formula (a) aims at maximising the use of the ASC (i.e., approximating worst-case behaviour of the operator and other actors) while maximising the probability of finishing two tasks, i.e., finishing a workpiece and carrying through cell maintenance. This query does not take into account further optimisation parameters defined for mitigations and resumptions. As opposed to that, Formula (b) fosters the maximisation of *productivity*, any combination of decisions allowing the finalisation of tasks is preferred, hence, transitions leading to accidents or the use of the ASC are equally neglected. While Formula (c) also forces the environment to trigger the ASC, these policies represent the best ASC usage in terms of *nuisance* and *effort*. Because of constraints in the use of \mathbf{R}_{\min} for MDPs, we maximise costs interpreting positive values as negative (e.g. the higher the nuisance the better).

We investigate the Pareto curves of the policies synthesised from the Formulas (a) to (c). For policies with less than 1000 states, we inspect the corresponding policy graphs (e.g. whether there is a path from *initial* to *final* or whether paths from unsafe states reachable from *initial* avoid deadlocks). Fi-

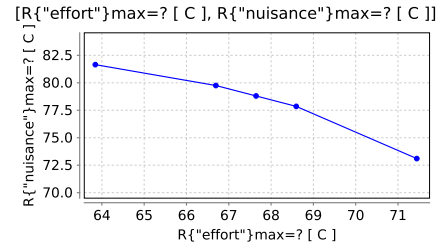


Figure 6: Pareto curve with five policies for Formula (c) for model 7

nally, we evaluate accident freedom according to Formula (1), except that we use $\mathbf{P}_{=?}$ for DTMCs instead of $\mathbf{P}_{\min=?}$.³

B. Results

For the experiment, we used YAP 0.5.1 and PRISM 4.5, on GNU/Linux 5.4.19 (x86, 64bit), and an Intel® Core i7-8665U with up to 8 CPUs of up to 4.8 MHz, and 16 GiB RAM.

Tab. IV shows the data collected from seven models created for **RQ1** and **RQ2**. The result $[\mu] = [1, 1, 1]$ for a policy denotes 100% conditional accident freedom. This desirable result is most often achieved with Formula (c) due to the fact that simultaneity of decisions of the environment and the ASC in the same state is avoided by focusing on rewards only specified for ASC actions. Such rewards model the fact that an ASC is usually much faster than an operator. Formulas (a) and (b) show poorer accident freedom because *productivity* rewards given to the environment compete with rewards given to the ASC to exploit its risk reduction *potential*.

For demonstration of YAP’s capabilities, the incident RT and the accident RC are included in the risk model without handler commands. However, these factors add further constraints on $R(F)$ to be dealt with by the ASC. Hence, *mr* stays at 15 actions and *c* rises to 15 constraints. In model 7 (last line of Tab. IV), $R(F)$ (122 risk states) and the Ξ -region of S (12079 states) differ by two orders of magnitude. Risk states offer a higher level of abstraction for state assessment. The derivation of properties that focus on relevant regions of the MDP state space from a risk structure can ease the state explosion problem in explicit model checking. For example, the constraints HRW prevents HC and HC prevents HRW express that the combined occurrence of HC and HRW is considered infeasible or irrelevant by the safety engineer. Hence, checking properties of the corresponding region of the MDP state space can be abandoned.

For **RQ3**, we can at the current stage of this project only provide a ballpark figure for the *detection and handling overheads*. Let $t: \alpha_{\mathcal{P}} \rightarrow \mathbb{R}$ be the processing time required for an action, e.g. for the calculation of the detection of HC in e^{HC} . If implemented as part of a sequential cell controller, the ASC requires a time slot of length $\sum_{f \in F} t(e^f)$ in each control cycle. If monitored simultaneously in dedicated ASC hardware, the slowest detection rate for F is $1/\max_{f \in F} t(e^f)$. The overhead for handling f can be estimated from Fig. 4 and may range from $t(m^{\text{HC}})$ to $\sum_{k \in \{sm, a, sf\}} (t(m_k^f) + t(m_{r,k}^f))$.

³To keep manual workload under control, if PRISM lists several adversaries, we apply the experiment procedure only to the first listed.

Table IV: Results of the experiment for **RQ1** (scalability) and **RQ2** (accident-free operation)

Risk Model [†]		MDP [†]			(a) max-ASC [†]			(b) max-prod			(c) opt-ASC				
F	mr/c	$ R(F) $	t_Y	$\mathbf{P}_{\neg A}$	Ξ	sta/tra	$\mathbf{P}_{\neg A}$	Ξ	t_P	$\mathbf{P}_{\neg A}$	Ξ	t_P	$\mathbf{P}_{\neg A}$	Ξ	t_P
			[ms]	$[\mu]$			$[\mu]$		[s]	$[\mu]$		[s]	$[\mu]$		[s]
HC	5/0	3	40	[.9,.9,.9]	14	322/1031	[1,1,1]	3	.02	[1,1,1]	1	.02	[1,1,1]	6	.15
+HS	9/2	5	52	[.92,.96,.98]	256	930/3483	[.07,.66,1]	11	.77	[0,.88,1]	8	.82	[.95,.98,1]	18	.9
+WS	11/3	8	44	[.93,.97,1]	288	1088/3865	[0,.29,1]	17	2.1	[0,.8,1]	5	2	[1,1,1]	24	1.5
+HRW	13/7	16	65	[.93,.97,1]	981	7675/33322	[1,1,1]	17	9.7	[1,1,1]	11	9.4	[1,1,1]	15	13.3
+HW	15/8	36	76	[.93,.97,1]	2296	21281/98694	[1,1,1]	15	42.9	[0,.71,1]	7	41.4	[1,1,1]	15	46.6
+RT	15/9	50	87	[.93,.97,1]	2864	21965/100133	[1,1,1]	13	48.2	[1,1,1]	9	46.4	[1,1,1]	15	53.8
+RC	15/15	122	162	[.93,.99,1]	12079	21670/102263	[0,.94,1]	35	38	[0,.72,1]	22	36.6	[1,1,1]	36	51.1

[†] F ...critical event set; mr/c ...number of mitigations+resumptions/constraints; $|R(F)|$...cardinality of the *relevant subset* of $R(F)$ defined in Sec. IV; t_Y ...YAP’s processing time; $\mathbf{P}_{\neg A}$...probability of conditional accident freedom; Ξ ...set of F -unsafe states; sta/tra ...number of states/transitions of the MDP (sta equals the size of the policies); Formulas (a) to (c)...optimisation queries; t_P ...PRISM’s processing time

C. Discussion

Relative Safety of a Policy: To simplify game-theoretic reasoning about \mathcal{M} , we reduce non-deterministic choice for the environment (i.e., operator, robot, welder). The more deterministic such choice, the closer the gap between policy space $\Pi_{\mathcal{M}}$ and ASC design space. Any decisions left to the environment will make a verified policy π *safe* relative to π^* ’s environmental decisions. These decisions form the *assumption* of the ASC’s *safety guarantee*. Occupational health and safety assumes trained operators not to act maliciously, suggesting “friendly environments” with realistic human errors. To increase priority of the ASC, we can express such an assumption, for example, by minimising *risk* and maximising *pot*.

Sensing Assumptions: In our example, the ASC relies on the detection of an *operator* (e.g. extremities, body) and a *robot* (e.g. arm, effector) entering a location, the *cell state* (e.g. grabber occupied, workbench support filled), and the *work-piece location* (e.g. in grabber, in support). For \mathcal{M} , we assume the tracking system (i.e., range finder and light barrier in the industrial setting, MS Kinect in the lab replica) to map the *location* of the operator and robot to the areas “at table”, “at workbench”, “in cell”, and “at welding spot”. In Fig. 1b, the range finder signals “at welding spot” if the closest detected object is nearer than the close range, and “in cell” if the closest object is nearer than the wide range. Tracking extensions, not discussed here, could include object silhouettes and minimum distances, operator intent, or joint velocities and forces.

Sensor Faults: pGCL, as we used it, requires care with the modelling of real-time behaviour, particularly, when actions from several concurrent modules are enabled. To model real-time ASC behaviour, we synchronise operator actions with sensor events and force the priority of ASC reactions in π^* by maximising the risk reduction potential (cf. *pot* in Tab. III). While *synchronisation* restricts global variable use, increasing \mathcal{M} ’s state space, we found it to be the best solution.

Model Debugging and Tool Restrictions: To reduce the state space, we strongly discretise *location*. To simplify debugging, we use probabilistic choice in synchronous updates only in one of the participating commands. To support synchronisation with complex updates, we avoid global variables.

State rewards allow a natural modelling of, e.g. risk exposure. However, in PRISM 4.5, one needs to use action rewards for multi-objective queries of MDPs. Risk gradient matrices

help to overcome a minor restriction in PRISM’s definition of action rewards.⁴ Alternatively, we could have introduced extra states at the cost of increasing \mathcal{M} ’s state space, undesirable for synthesis. Rewards require the elimination of non-zero end components (i.e., deadlocks or components with cycles that allow infinite paths and, hence, infinite reward accumulation). PRISM provides useful facilities to identify such components, however, their elimination is non-trivial and laborious in larger models and can require intricate model revisions.

VII. CONCLUSION

We introduced a tool-supported method for the verified synthesis of automatic safety controllers from Markov decision processes, focusing on human-robot collaboration settings. These controllers implement regulatory safety goals for such settings. We describe steps for streamlining the modelling of MDPs and demonstrate our method using two tools, YAP [19] for structured risk modelling and MDP generation and PRISM [27] for probabilistic model checking and MDP policy synthesis. We show that our approach can be used to incrementally build up multi-hazard models including alternative mitigation and resumption options. We also discuss how our methodology can reduce effort in explicit model checking when dealing with large state spaces. Our approach improves the state of the art of ASC synthesis for HRC settings, particularly, when dealing with multiple risks, mitigation options, and safety modes. Verification results from using our method can contribute evidence to an ASC *assurance case* [30]–[32].

Future Work: Our approach limits the inference of high ASC effectiveness from high conditional accident freedom of the associated policy. Our setting can require the comparison of the extents to which decisions of the ASC and the environment contribute to the accident freedom. We plan to explore game-theoretic settings to remove this limitation.

The evaluation of the verified controller in the manufacturing cell (e.g. overhead in resource usage, influence on nominal operation) is out of scope of this paper. Such an evaluation requires the translation of the controller into an executable form. Our next steps will include the conversion of the synthesised DTMC into a program for the digital twin

⁴Currently, rewards cannot be associated with particular updates, i.e., with incoming transitions rather than only states.

simulator and the replica of the cell. Note that this translation has to be verified to match the executable form with the verified properties. Additionally, we plan to derive tests for this program from the facilities provided by the simulator.

For optimal synthesis, the proposed method uses parameters such as upper risk and severity bounds in constraints. We plan to introduce parameters for the probabilities into the MDP, supported by tools such as EVOCHECKER [33], and to use parametric risk gradient matrices by extending YAP. We intend to explore the use of EVOCHECKER to avoid the split of the verification procedure into two stages (cf. Secs. V-D2 and V-D4). We also plan to explore online policy synthesis [34] to allow more variety in environmental decisions (e.g. malicious operators). This corresponds to weakening the assumptions under which the ASC can guarantee safety.

Unable to collect data (cf. Sec. V-B) from an industrial application, we had to make best guesses of probabilities. However, the frequency of undesired intrusion of operators into the safeguarded area and accident likelihood can be transferred into our example. This example can be extended by randomised control decisions with fixed probabilities (e.g. workload), by adding uncertain action outcomes (e.g. welding errors), and by time-dependent randomised choice of mitigation options. To use time in guarded commands, we want to explore clock-based models as far as synthesis capabilities allow this, rather than only using reward structures.

Acknowledgements: This research was funded by the Lloyd's Register Foundation under the Assuring Autonomy International Programme grant CSI:Cobot. We are grateful for many insights into manufacturing robot control from our project partners at the University of Sheffield and our industrial collaborator. We also thank David Parker for his advice in the use of PRISM's policy synthesis facility.

REFERENCES

- [1] P. Nicolaisen, "Occupational safety and industrial robots," in *Robot Safety*, Bonney and Yong, Eds. IFS, 1985, pp. 33–48.
- [2] R. H. Jones, "A study of safety and production problems and safety strategies associated with industrial robot systems," Ph.D. dissertation, Imperial College, 1986.
- [3] A. D. Santis, B. Siciliano, A. D. Luca, and A. Bicchi, "An atlas of physical human–robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, 2008.
- [4] N. Sugimoto, "Safety engineering on industrial robots and their draft standards for safety requirements," in *7th Int. Symposium on Industrial Robots*, 1977, pp. 461–470.
- [5] R. Alami, A. Albu-Schaeffer, A. Bicchi, R. Bischoff, R. Chatila *et al.*, "Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2006.
- [6] S. Haddadin, A. Albu-Schaeffer, and G. Hirzinger, "Requirements for safe robots: Measurements, analysis and new insights," *The Int. Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1507–1527, 2009.
- [7] X. V. Wang, Z. Kemény, J. Váncza, and L. Wang, "Human-robot collaborative assembly in cyber-physical production: Classification framework and implementation," *CIRP Annals*, vol. 66, no. 1, pp. 5–8, 2017.
- [8] L. Kaiser, A. Schlotzhauer, and M. Brandsttner, "Safety-related risks and opportunities of key design-aspects for industrial human-robot collaboration," in *LNCS*. Springer, 2018, pp. 95–104.
- [9] B. Matthias, S. Kock, H. Jerregard, M. Kallman, and I. Lundberg, "Safety of collaborative industrial robots: Certification possibilities for a collaborative assembly robot concept," in *IEEE Int. Symposium on Assembly and Manufacturing (ISAM)*, 2011.
- [10] J. A. Marvel, J. Falco, and I. Marstio, "Characterizing task-based human-robot collaboration safety in manufacturing," *IEEE Tran. on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 2, pp. 260–275, 2015.
- [11] ISO 10218, "Robots and robotic devices – safety requirements for industrial robots," Robotic Industries Association (RIA), Standard, 2011. [Online]. Available: <https://www.iso.org/standard/51330.html>
- [12] R. B. Gillespie, J. E. Colgate, and M. A. Peshkin, "A general framework for cobot control," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 391–401, 2001.
- [13] ISO/TS 15066, "Robots and robotic devices – collaborative robots," Robotic Industries Association (RIA), Standard, 2016. [Online]. Available: <https://www.iso.org/standard/62996.html>
- [14] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.
- [15] B. Hayes and B. Scassellati, "Challenges in shared-environment human-robot collaboration," in *Collab. Manipulation Workshop at HRI*, 2013.
- [16] E. Helms, R. D. Schraft, and M. Hagele, "rob@work: Robot assistant in industrial environments," in *11th IEEE Int. Workshop on Robot and Human Interactive Communication*, 2002.
- [17] M. Gleirscher, "Run-time risk mitigation in automated vehicles: A model for studying preparatory steps," in *1st iFM Workshop Formal Verification of Autonomous Vehicles (FVAV)*, ser. EPTCS, no. 257.8, 2017.
- [18] M. Gleirscher, R. Calinescu, and J. Woodcock, "Risk structures: A design algebra for risk-aware machines," Department of Computer Science, University of York, Working paper, 2020, arXiv:1904.10386.
- [19] M. Gleirscher, *YAP Against Perils: User's Manual*, Technical University of Munich and University of York, 2020. [Online]. Available: <http://gleirscher.de/yap/yap-manual.pdf>
- [20] M. Askarpour, D. Mandrioli, M. Rossi, and F. Vicentini, "SAFER-HRC: Safety analysis through formal vERification in human-robot collaboration," in *LNCS*. Springer, 2016, pp. 283–295.
- [21] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," in *Formal Methods for the Design of Comp., Comm. and Soft. Sys.: Performance Evaluation (SFM)*, ser. LNCS, M. Bernardo and J. Hillston, Eds. Springer, 2007, vol. 4486, pp. 220–70.
- [22] A. Orlandini, M. Suriano, A. Cesta, and A. Finzi, "Controller synthesis for safety critical planning," in *IEEE 25th Int. Conference on Tools with Artificial Intelligence*. IEEE, nov 2013.
- [23] A. Cesta, A. Orlandini, G. Bernardi, and A. Umbrico, "Towards a planning-based framework for symbiotic human-robot collaboration," in *IEEE 21st Int. Conference on Emerging Technologies and Factory Automation (ETFA)*, sep 2016.
- [24] J. Heinzmann and A. Zelinsky, "Quantitative safety guarantees for physical human-robot interaction," *The Int. Journal of Robotics Research*, vol. 22, no. 7-8, pp. 479–504, 2003.
- [25] P. Long, C. Chevallereau, D. Chablat, and A. Girin, "An industrial security system for human-robot coexistence," *Industrial Robot: An Int. Journal*, vol. 45, no. 2, pp. 220–226, 2018.
- [26] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT P., 2008.
- [27] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *23rd CAV*, ser. LNCS, vol. 6806. Springer, 2011, pp. 585–591.
- [28] N. G. Leveson, *Safeware: System Safety and Computers*. Addison-Wesley, 1995.
- [29] —, *Engineering a Safer World: Systems Thinking Applied to Safety*, ser. Engineering Systems. MIT P., 2012.
- [30] M. Gleirscher, S. Foster, and Y. Nemouchi, "Evolution of formal model-based assurance cases for autonomous robots," in *17th Int. Conf. SEFM*, ser. LNCS, vol. 11724. Springer, 2019.
- [31] S. Foster, M. Gleirscher, and R. Calinescu, "Towards deductive verification of control algorithms for autonomous marine vehicles," in *25th Int. Conf. ICECCS, Singapore*, 2020, arXiv:2006.09233.
- [32] R. Calinescu, D. Weyns, S. Gerasimou, M. U. Iftikhar, I. Habli, and T. Kelly, "Engineering trustworthy self-adaptive software with dynamic assurance cases," *IEEE Transactions on Software Engineering*, vol. 44, no. 11, pp. 1039–1069, Nov. 2018.
- [33] S. Gerasimou, R. Calinescu, and G. Tamburrelli, "Synthesis of probabilistic models for quality-of-service software engineering," *Automated Software Engineering*, vol. 25, no. 4, pp. 785–831, 2018.
- [34] R. Calinescu, M. Autili, J. Cámara, A. Di Marco, S. Gerasimou, P. Inverardi, A. Perucci, N. Jansen, J.-P. Katoen, M. Kwiatkowska *et al.*, "Synthesis and verification of self-aware computing systems," in *Self-Aware Computing Systems*. Springer, 2017, pp. 337–373.