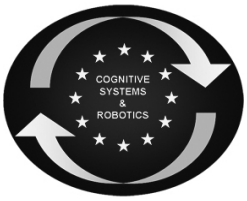
SAPHARI

SAFE AND AUTONOMOUS PHYSICAL HUMAN-AWARE ROBOT INTERACTION



Project funded by the European Community's 7th Framework Programme (FP7-ICT-2011-7)
Grant Agreement ICT–287513

Deliverable D1.3.1

Report on safety monitoring framework and safe control strategies

Deliverable due date: November 2014	**Actual submission date:** 16th December 2014
Start date of project: 1 November 2011	**Duration:** 48 months
Lead beneficiary: CNRS-LAAS	**Revision:** Final

Nature: R	**Dissemination level:** PU
R = Report P = Prototype D = Demonstrator O = Other	PU = Public PP = Restricted to other programme participants (including the Commission Services) RE = Restricted to a group specified by the consortium (including the Commission Services) CO = Confidential, only for members of the consortium (including the Commission Services)

Contributors

Mathilde Machin - LAAS-CNRS, France
Jérémie Guiochet - LAAS-CNRS, France
Tim Guhl - KUKA, Germany
Steffen Walther - KUKA, Germany
Vito Magnanimo - KUKA, Germany

Executive Summary

Due to its complexity, a controller of an autonomous system cannot be completely tested or verified, and so, some faults can remain. Moreover, operational conditions can differ from those intended at design time, in particular during interactions. As the safety of human beings may be endangered by autonomous systems, a faulty controller or an unexpected environment condition should not result in catastrophic consequences.

One fault tolerance mechanism to address this issue is a safety device that is only responsible to ensure safety. Such a safety monitor is intended to be simple enough to be verifiable and independent from the main controller. The safety monitor has not only observation means, but also intervention means (engage the brakes, block a joint, emergency stop, etc.), and should be able to check online safety rules. Monitoring and triggering interventions is a classical approach to ensure safety. With respect to classical systems, a collaborative autonomous robot faces many different hazards in a more changing environment. Hence, such safety rules may be complex to specify, and some inconsistencies between safety actions may occur.

We address this issue by proposing a method and a tool for the automatic generation of safety rules. This method starts from a risk analysis process based on UML and HAZOP, and then uses a tool we developed (based on a model checker) to synthesize the safety rules. Potentially excessive limitation of system functionality due to presence of the safety monitor is addressed through the notion of permissiveness.

A first validation of the approach has been done during a collaboration between LAAS-CNRS and KUKA. After a risk analysis of the KUKA use case, several safety invariants has been carried out through a collaboration by LAAS and KUKA. These safety invariants have been formalized and they resulted in safety rules. Three safety invariants have been successfully implemented. A next step, is the implementation of safety rules at different levels of the robot controller architecture.

The method and the tool presented in this report are detailed in two publications that have been accepted in peer-reviewed international conferences:

- M. Machin, F. Dufosse, J.-P. Blanquart, J. Guiochet, D. Powell, and H. Waeselynck, “Specifying safety monitors for autonomous systems,” in SAFECOMP. Springer, 2014.
- M. Machin, F. Dufosse, J. Guiochet, D. Powell, M. Roy, and H. Waeselynck, “Model-checking and Game Theory for Synthesis of Safety Rules,” in HASE. IEEE, 2015.

Table of contents

1 Introduction

The autonomous robotic systems of interest to us offer a wide range of features and operate in a diverse unstructured environment. They can thus be complex, which makes them difficult to verify. Moreover, diversity of the environment implies that testing cannot significantly cover the situations that the system will face. Here, we choose a classical fault tolerance approach by considering online safety measures implemented in a device called a *safety monitor*, that is simple and independent from the main control channel, and thus easier to verify. The monitor is solely responsible for safe system behavior. To this end, the monitor is equipped with means for context observation (i.e., sensors) and is able to trigger safety interventions.

The monitor behavior is specified declaratively by a set of *safety rules*, each defining one intervention to apply in certain observation conditions. However, safety interventions may also prevent the system from fulfilling its functions. For instance, a vehicle whose emergency brakes are permanently engaged is useless. We require the monitor to be *permissive* with respect to the possibility for the system to perform useful tasks.

We propose a process based on hazard analysis to specify safety monitors and extend it by means of formal methods. Once a hazard is identified, it is necessary to specify what the monitor has to do to avoid it, i.e., the safety rules. We aim to explore solutions very early in the autonomous system design process. Thus, many observations and interventions can be considered in a first design iteration, whereas only the most appropriate ones are actually developed and implemented. We propose to use model-checking to explore and check the specifications.

2 Baseline and concepts

We introduce here the underlying concepts of our work, based on definitions adapted from [5], and then present the overall process.

2.1 Concepts

Taking inspiration from the IEC 61508 standard [1], we define a *safety monitor* as a device responsible for safety, in opposition to the main control channel which is responsible for all other functional and non-functional requirements of the system. The monitor is equipped with means for context observation (i.e., sensors) and able to trigger safety interventions. The safety monitor is independent from the main control channel, as regards its means of observation, computation and intervention. It is required to protect against all faults that adversely affect safety, including interaction faults. The whole safety channel is assumed fault-free (for example, we consider that the sensors available to the monitor are perfect, without uncertainty.) In practice, this must be achieved through classical redundancy and verification techniques. We focus in our work on the upstream task of obtaining a correct high-level specification with respect to safety and permissiveness.

A safety invariant (SI) is a necessary and sufficient condition to avoid a hazardous situation. If a safety invariant is violated, we assume that damage is immediate and irreversible, with no possible recovery. We refer to any state violating the safety invariant as a *catastrophic state*.

Example: "the robot speed shall not exceed $3\ m/s$*" (where* $3\ m/s$ *is the speed beyond which harm is considered to be inevitable).*

A safety intervention is an activity carried out explicitly to prevent the system from violating a safety invariant by constraining the system behavior. An intervention is only applicable in states satisfying its associated *precondition*. We distinguish two types of interventions: inhibitions and actions.

A safety inhibition prevents a change in system state. When triggered, an inhibition is assumed to be immediately effective.

Example : "lock the wheels" (with "robot stationary" as precondition).

A safety action triggers a change in system state (and implicitly prevents other state changes).

Example : "apply emergency brake".

A safety trigger condition defines when the intervention is needed. The trigger condition is chosen such that it becomes true before the safety invariant is violated.

Example: "the robot speed is greater than $2\ m/s$ *(i.e., less than the safety invariant threshold of* $3m/s$*)".*

A safety rule defines a way of behaving in response to a hazardous situation. A safety rule can be operationalized as an if-then rule:

Safety rule $\triangleq$ *if* [safety trigger condition] *then* [safety intervention].

Example: "if the robot speed is greater than $2\ m/s$ *then apply emergency brake."*

As illustrated in Figure 1, the safety invariant defines the partition between catastrophic states and non-catastrophic states of the monitored system. Interventions have to be applied before the catastrophe, i.e., in non-catastrophic states. Now, interventions add constraints to the system behavior. So the set of non-catastrophic states is partitioned into warning states, where interventions are applied, and safe states, in which the system operates without constraint. The warning states are defined such that every path from a safe state (e.g., $\mathbf{x}_s$ on Figure 1) to a catastrophic state, e.g., $\mathbf{x}_c$, passes through a warning state, e.g., $\mathbf{x}_w$. The warning state enables triggering of an intervention to abort the path to the catastrophic state.

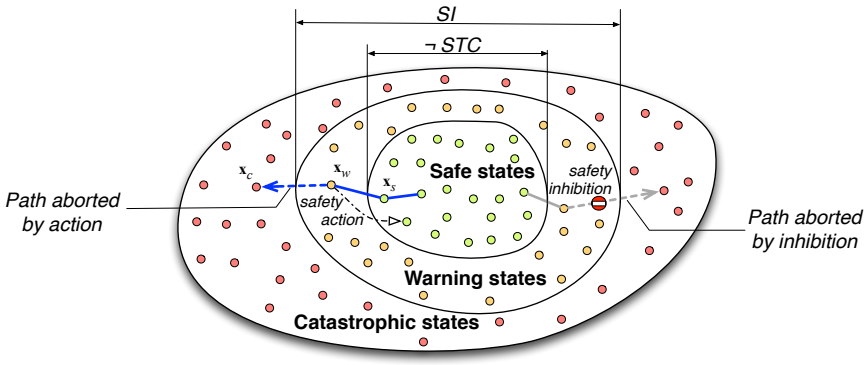


Figure 1: Partition of system states in catastrophic, warning and safe states

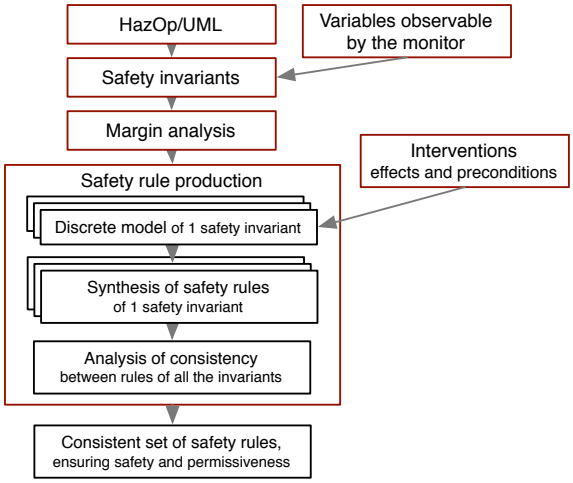


Figure 2: Overview of the process

We assess the monitor and its safety rule set according to the following three properties:
Safety is the ability to ensure that the safety invariants are never violated, i.e., that catastrophic states are unreachable.
Permissiveness is the ability to allow the system to perform its tasks.
Validity specifies that no intervention is applied while its precondition is false.

Safety and permissiveness are antagonistic. We take this antagonism into account by designing the monitor to be *maximally permissive with respect to safety*, i.e., to restrict functionality only to the extent necessary to ensure safety.

2.2 Process overview

Figure 2 presents the overall process. We base our process on a HAZOP-UML hazard analysis, which outputs safety invariants expressed in natural language. We consider as a running example a mobile robot with a manipulator arm and the informal safety invariant *The arm must not be extended beyond the base when the speed is greater than* V_0.

The safety invariant is then expressed formally with predicates on variables that are observable by the monitor. We focus for now only on predicates involving a variable compared to a fixed threshold. This type of safety

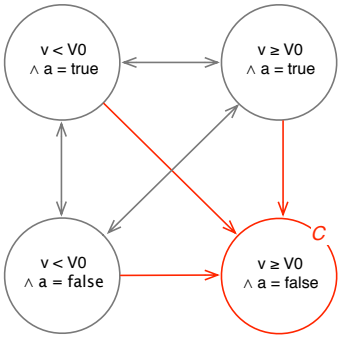


Figure 3: The example model without margin

threshold is amenable to formal verification and is used in many real systems. Considering the two monitor observations: the absolute speed v, and a Boolean observation of the arm position a ($true$ when the arm is above the base, $false$, when the arm is extended), the example safety invariant is formalized as $v < V_0 \vee a = true$. This can be represented througha graph as in Figure 3.

The margin analysis partitions non-catastrophic states into safe states and warning states by splitting variable value intervals or sets. This is done one variable after another. For example, the speed interval $[0, V_0[$ from the safety invariant is partitionable according to a margin m in two intervals $[0, V_0 - m[$ and $[V_0 - m, V_0[$. In the case of arm position, the observation is Boolean. The singleton value set $\{true\}$ cannot be partitioned, hence no margin exists. Formal conditions for the existence of a margin are studied in [5].

From the margin analysis, we can discretize variables involved in the safety invariant in order to synthesize safety rules. We call this the discrete model analysis, which is detailed in Section 3. It is composed of three main steps: creation of a discrete model, rule synthesis, and rule consistency checking. In order to keep models simple enough to be validated, each safety invariant is modeled separately. The state variables of the model are the observable variables discretized by intervals according to the thresholds of the safety invariant and the existing margins. The discrete model (e.g., Figure 4) is the Cartesian product of the variable partitions. A catastrophic state is one that violates the safety invariant (there is one catastrophic state on Figure 4, labeled C). The warning states (W) are those that lead the system to the catastrophe in one step. Interventions are modeled using the same discretized variables. In the example the monitor is able to brake (action) and to prevent the arm from extending (inhibition).

The monitor is responsible for neutralizing every transition leading to a catastrophic state. For instance, Figure 5 illustrates a satisfying safety rule set, which applies braking in $s3$ and arm inhibition in $s1$ and $s2$. Additionally to the transitions leading directly to the catastrophic state, several other transitions are deleted. The safety rule set respects the safety properties, as the system cannot enter the catastrophic state. All non-catastrophic states are reachable. Nevertheless, there is some loss of permissiveness as the system cannot stay in $s3$. We consider this to be acceptable. In Section 3.4, we propose a method to find systematically such safety rule sets.

As safety invariants are processed separately, the final step is to check the consistency between the safety rule sets from different safety invariants. This is addressed in Section 3.5.

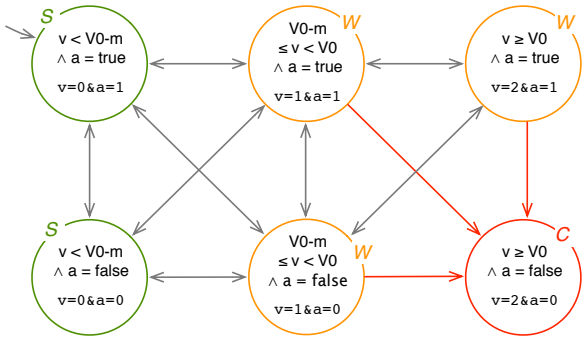


Figure 4: The example discrete model from the partitions $\{true, false\}$ for arm position, and $\{[0, V_0 - m[,\ [V_0 - m, V_0[,\ [V_0, V_{max}[\}$ for speed

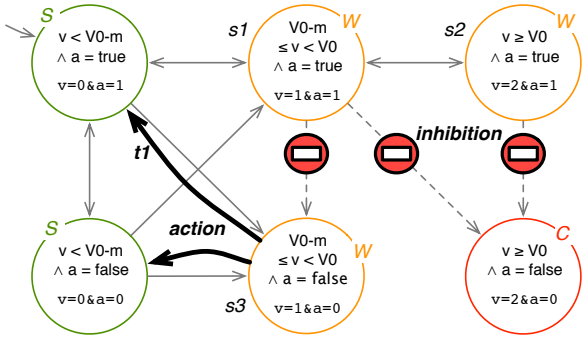


Figure 5: The example model with a safety rule set

3 Safety rule production

Given a safety invariant, several safety rules are usually needed to avoid violation of the safety invariant. We call a *safety strategy* a set of rules applied with respect to a single safety invariant. In this section, we aim to synthesize a safe, permissive and valid strategy based on the discrete model.

We have two approaches to synthesize strategies. The *automatic synthesis* finds strategies fast, given permissiveness requirements, by exploring automatically the various combinations of safety rules. The *interactive synthesis* is more manual and enable the user to build or modify a strategy rule by rule. As it is less convenient than the automatic synthesis, it is omitted here for brievity. It is fully presented in [3].

3.1 Tools

We use the modeling language SMV and the model-checker NuSMV2 [2]. SMV enables the declaration of integer variables and constraints on their behavior. NuSMV builds transparently the Cartesian product of the ranges of all variables. When no constraint is declared, all the combinations of variable values (i.e., states) are possible and all transitions between each pair of states are implicitly declared. Constraints are then added to delete undesired states and transitions. As for variables, time is discrete. It is modeled by the operator `next()`. NuSMV is well-adapted to our variable-oriented modeling approach. Moreover, the implicit transition declaration is convenient for modeling the whole physically possible behavior.

In the following, SMV code and output of NuSMV are given in typewriter font. `!`, `&`, `|`, `->` are the classical logical operators NOT, AND, OR, $IMPLIES$. We have developed a template file to facilitate the modeling and

to allow the process to be automated.

3.2 System and intervention modeling

The domain of each variable of the safety invariant is partitioned according to the thresholds of the safety invariant and the margin (if it exists), and the resulting elements are numbered. For instance $\{[0, V_0-m[, [V_0-m, V_0[, [V_0, V_{max}[\}$ is encoded as $\{0,1,2\}$ (see Figure 4). Continuity of variables, i.e., contiguity of partition elements, is modeled as the constraint: `next(x) = x | x+1 | x-1`, i.e., a variable `x` can stay in the same interval or move to an adjacent interval, but it cannot jump from one interval to another that has no common boundary.

We then model possible dependencies between variables. Nevertheless, some dependencies cannot be modeled in a discrete way or with a given partition. If a dependency is not modeled, the discrete model has less constraints than it should, or from another point of view, it has too many transitions. If this "super-graph" is safe, so is the "true" model. On the contrary, the permissiveness results of the super-graph are not trustworthy. The resulting strategies are always safe; but their level of permissiveness depends on the dependency modeling effort.

Interventions are always effective (when their preconditions are true), provided some environmental and dimensioning assumptions. A safety braking action requires to consider for example a maximum slope rate, a maximum torque from the motors. Safety interventions are then modeled as constraints that may be applied or not. Consider a discretized speed `v`. The braking action and the acceleration inhibition can be modeled by:

```
braking ->  ( (v!=0 -> next(v)=v-1) & (v=0 -> next(v)=0) )
acc_inhibition -> next(v)!=v+1
```

As these examples show, an intervention usually adds a constraint on only one variable, and leaves the others free. For example, at the same time step: speed can be decreased by braking, and the arm can fold (as in transition $t1$ in Figure 5).

We make no restrictive assumption about the behavior of the main control channel. The system model represents what is physically possible in the system without a monitor. Therefore, safety interventions only remove transitions, i.e., possible behaviors, and cannot add transitions, i.e., add physically impossible behaviors.

3.3 Safety, permissiveness and validity modeling

Monitor properties are expressed in CTL (*Computation Tree Logic*), which is entirely supported by NuSMV without any syntax change. Time along paths is modeled by three operators: X for a property to hold in the next state, G to hold on the entire path, F to hold eventually. The branching aspect is modeled by A, all the branches, and E, there exists a branch. A CTL operator is composed of one branching operator and one time operator. It is applied on states, or more generally on statements on the system state.

To model safety, we use the atomic property $cata$ to denote the catastrophic states. $cata$ is the negation of the safety invariant, e.g., `cata := speed=2 & arm_pos=0`. Safety is modeled as the unreachability of the catastrophic states, i.e., in CTL, $AG \neg cata$. The expression of $cata$ is the only user task in the initial property modeling. Permissiveness and validity properties are generated automatically. During the synthesis, the user is supposed to remove some permissiveness properties according to the accepted permissiveness loss choices.

Permissiveness is translated by two liveness properties applied to each non-catastrophic state s_{nc}:

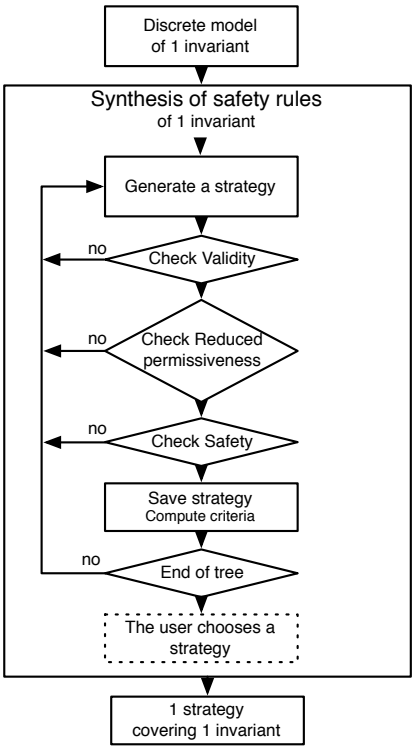


Figure 6: The synthesis algorithm

- SIMPLE REACHABILITY *EF* s_{nc}
  The state is reachable from the initial state.
- UNIVERSAL REACHABILITY *AG EF* s_{nc}
  The state is reachable from any reachable state.

The automaton without any safety rule is usually permissive because it is only a structure without specified behavior. Variables can change freely their values. Similarly, it is unsafe, as catastrophic states are reachable.

Validity specifies that interventions are not applied in states that violate their preconditions. We express this as:

$$AG \bigwedge_{i \in Interventions} i \rightarrow precondition_i$$

where $Interventions$ is the set of the candidate interventions and $precondition_i$ is the precondition associated to intervention i.

Once the safety invariant and the interventions have been defined, and the properties have been generated, we can synthesize a strategy using the synthesis algorithm.

3.4 Synthesis algorithm

The synthesis algorithm (Figure 6) runs from the discrete model. It outputs all safe and valid strategies that satisfy the permissiveness requirements (if any such strategies exist). The synthesis algorithm is based on the enumeration of the strategies through a branch-and-cut algorithm and the verification of properties by

Table 1: Performance experiments

	Number of strategies	Examined nodes		Number of solutions	Costs	
		Number	%		Time	Memory
2var2val_l	4096	6	0.15%	0	0.08s	11.8M
2var2val_a	262144	8	0.003%	0	0.16s	11.9M
2var3val_l	4096	74	1.8%	36	1.3s	12.2M
2var3val_a	262144	109	0.04%	36	3.5s	12.3M
3var2val_l	$4.39 * 10^{12}$	764	<0.001%	108	15s	12.3M
3var2val_a	$9.22 * 10^{18}$	1191	<0.001%	108	35s	12.5M
3var3val_l	$4.39 * 10^{12}$	94723	<0.001%	48000	1h7m	14M
3var3val_a	$9.22 * 10^{18}$	159145	<0.001%	48000	2h6m	14.3M

NuSMV. All the possible strategies are structured as the set of leaves of a tree. Some pruning criteria are applied to faster the tree exploration. A comprehensive description of the used algorithm can be found in [4]. The synthesis is implemented using NuSMV scripts and a C program.

To assess the tool efficiency, we test it on generic models. The models are generated as follows. All variables have the same number of values. For example, in Table 1, system `2var3val` has two variables and each variable has 3 values. The initial state has all variables at value 0, and the only catastrophic state has variables at their maximum values. There are two possible models for interventions. In the first one, (suffix `_a`), for any variable, the monitor can increase the value of the variable, decrease it, or inhibit any changes of value. In the second case (suffix `_l`), the monitor can only decrease and inhibit the variable values, and in addition, the variables cannot be inhibited when they take their maximum value. These models allow us to consider search spaces ranging for a few thousands strategies to more than 10^{18}. Experiments have been done on one core of an Intel Core I7-4770 processor running at 3.4GHz. The results are in Table 1.

The first columns gives the size of the search space, i.e. the number of complete strategies. It would be the number of steps of brute-force search. The second column compares our algorithm to brute-force search. For example, in the first row, our algorithm visits 6 nodes, while brute-force search would visit 4096 nodes ($6/4096 = 0.15$%). The gain is very high for all models. Finally, the third column gives the number of solutions found by the search. These solutions are a subset of the examined strategies and it is interesting to see how many extra nodes are visited to find them. In the largest model (`3var3val_a`), the total number of visited nodes is only 3.3 times the number of solutions. The pruning criteria are very efficient.

The memory consumption of our tool is reasonable and increases slowly.. The increase of the execution time is also kept reasonable, if one considers that there are 15 orders of magnitude in size between the smallest and largest search spaces, and that we do not stop the algorithm after a first solution is found. The pruning criteria allow us to limit the number of steps to explore the tree of strategies.

Nevertheless, it may be surprising that a model with only three variables requires a 2-hours synthesis. One might wonder whether the approach is useful in realistic cases. Firstly, the number of variables is not unrealistic. A safety invariant models only one safety-relevant aspect of a system. In the real system studied by [5], each invariant had no more than two variables. Secondly, the artificial models we used are generic, i.e., they have many interventions and no variable dependencies. It follows that there are numerous solutions to find, much more than in real cases. A preliminary version of the safety invariant “Tilting the box” from the KUKA use case has 4 variables (1 boolean variable and 3 variables with three values) and 3 interventions are available. Due to some dependency, the synthesis problem has only 4 minimal solutions. They are found in 3s by our tool.

Figure 7: The robot of the KUKA use case

3.5 Consistency between strategies

Different strategies may apply interventions simultaneously, which may be incompatible, e.g., braking and acceleration. To check strategy consistency, the previous models (with their strategies) are merged into a single model. When observable variables are common to several models but with different domain partitions, a new domain partition is defined by taking the union of the thresholds from the different models.

A concurrent application of the interventions i_1 and i_2 is checked by the CTL formula:

$$\neg EF(i_1 \wedge i_2)$$

4 Case Study

LAAS-CNRS and KUKA collaborate to apply the presented method on the KUKA use case, from the hazard analysis to the implementation of the safety monitor. The robot is composed of a Omnirob mobile base and an LWR arm (see Figure 7). It is an industrial co-worker in a manufacturing setting. It takes and places part boxes on shelves, work stations, or on the robot base in order to convey them. It operates in the human workspace. A complete description of this case study is given in [6]. Safety aspects of the use case are described in [7].

4.1 HAZOP-UML hazard analysis

From the UML description given in [7], we applied the HAZOP/UML method to find hazards. As a reminder the HAZOP-UML approach (Figure 8) is based on UML diagrams (sequence and use case), and uses some guide-words to perform a systematic analysis of deviations of the scenarios of use. In this application, it results in more than 100 HAZOP lines with a non-zero severity.

4.2 Safety invariant formalisation

From those HAZOP lines, 16 safety invariants have been formulated in natural language (see Table 2). In close collaboration, some invariants have then been formalized using only variables available to the safey monitor (see Appendix 5).

As the invariant 15 “The handguiding is not followed”, some invariants have not been formalized because the relevant variables were not observable by the monitor.

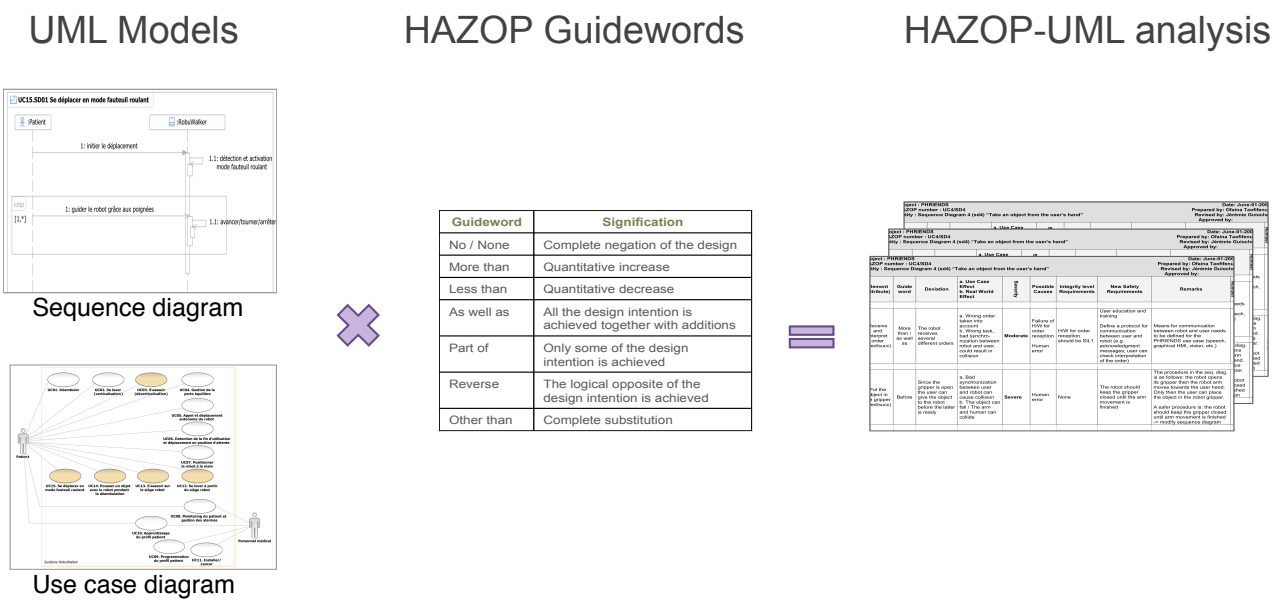


Figure 8: HAZOP-UML process

Table 2: List of Safety Invariants

Number	*Name*	*Description*
SI1	Tilt box	A box is in the gripper. If the box tilts too much, parts fall out.
SI2	Collision with platform	The robot platform collides with a human.
SI3	Arm and platform velocities	The platform moves. The arm moves and collides with a human. The collision velocity is the sum of the two velocities.
SI4	Gripper clamps human hand	The gripper fingers close on human part.
SI5	Workspace sharing	Collision in the particular case of workspace sharing in workstation B.
SI6	Collision	The robot arm collides something with an high torque.
SI7	Collision with extended arm	Collision between a human and the robot arm that is extended beyond the platform footprint while the platform is moving.
SI8	Robot in restricted area	The robot must not be in restricted area.
SI9	Platform motion overturn boxes on table	The gripper takes or places a box on a table. The platform moves, so the gripper moves horizontally at box height. The gripper overturn boxes and sharp parts scatter on table.
SI10	Excessive arm velocity	In general, the arm velocity must be limited.
SI11	Drop box	A box is in the gripper and is dropped anywhere but on table, robot storage, shelf or during handover.
SI12	Excessive platform velocity	In general, the platform velocity must be limited.
SI13	Excessive arm velocity during interaction	During physical interaction, the arm velocity must be limited.
SI14	Excessive platform velocity during guiding	The user moves the robot out of his way by guiding it (applying a force on robot arm). The platform velocity must be limited during this step.
SI15	The handguiding is not followed	In the quality control workstation, the user manipulates a box in the robot gripper by handguiding the robot arm. The robot arm does not follow the handguiding.
SI16	Clamping	The end-effector clamps a human part on table or robot storage. This hazard is effective only when the end-effector (empty gripper or box) is close to table.

4.3 Safety rule production

Safety invariants have been modeled within our framework, in collaboration with KUKA. Corresponding strategies have been synthesized using our tool. 3 invariant models are given in Appendix 5.

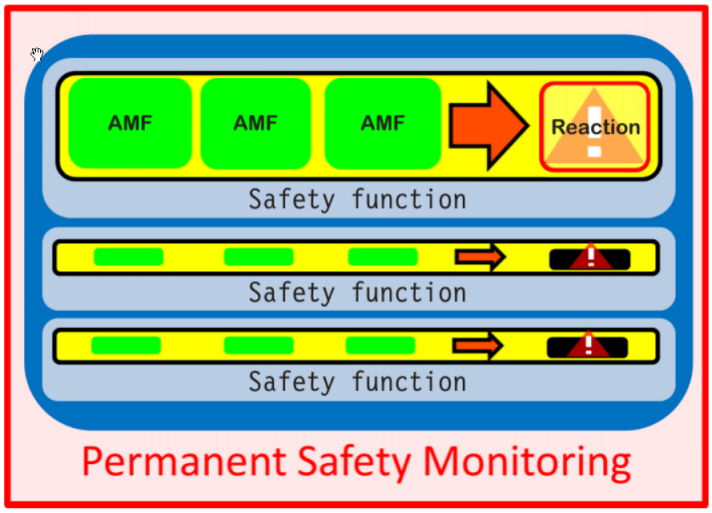


Figure 9: Overview of the safety monitoring of KUKA. Each AMF (*Atomic Monitoring Function*) is a predicate on an observable variable. If each AMF is assessed to true (AND operator) then the reaction is triggered. Safety functions runs concurrently (OR operator).

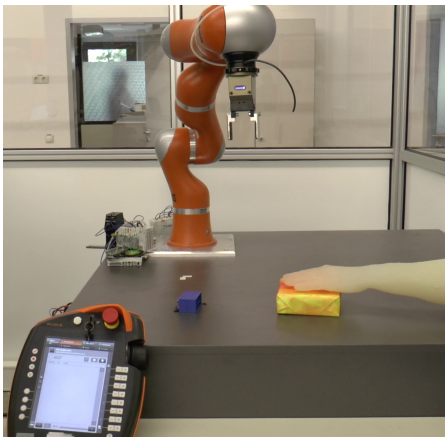


Figure 10: The experimental setup

4.4 Implementation

The safety channel used by KUKA is close to the assumptions presented in this report regarding the monitor. The safety channel of KUKA is executed redundantly on two devoted cores, which is a compliant design of the monitor regarding the independence and correctness requirements. Consequently, a restricted set of observable variables is available to the monitor. The safety channel behavior is declared through a table (see Figure 9).

In the context of our study, the KUKA architecture has two possible interfaces. The first is the safety channel, which is close to the assumptions of monitor correctness. The second is the usual Java interface, which enables the programmer to access to any observation and actuation. This interface is normally devoted to the functional robot programming.

The experimental setup (see Figure 10) was a LWR robot arm. The safety channel was simulated by the Java

interface. 3 strategies, presented in Appendix 5 were implemented, covering the hazards of clamping, collision and tilting in a “pick and place” task.

The experiments show that some assumptions of the method must not be neglected. The method requires that the control latency, the intervention efficiency and the observation accuracy are bounded and assessed, which was not possible in the preliminary implementation because of physical restrictions. In particular, the use of some observation requires to add their own safety rules. For example to be able to detect fast enough an increase of clamping force, the velocity of the arm must be limited. This limit must be, and has been, included in the safety monitor. The systematic identification of this kind of problems and of the numerical values of thresholds to avoid them is a challenge and a prerequisite to apply successfully our method.

Some limitations of the method have been identified. For example, the very fast changes of forces in a contact situation, can appear as a discontinuity in the sampled observations, violating a basic assumption of the method. In general, there is discussion to determine whether some variable observations, e.g., localization, are reliable enough to be used by the safety monitor, and in what conditions.

An extension of the current method to a double-layer safety monitor has been discussed.

5 Conclusions

We have described a method for obtaining a high-level safety monitor specification, taking into account the specific features of autonomous systems. We base it on hazard analysis, which is non-formal. Thanks to formal methods, we ensure that the derivation from formal safety invariants to safety rules is correct, provided the modeling of safety invariants is valid. Safety invariants are modeled separately in order to maintain model validability and to ensure scalability.

Our method justifies the modeling effort in that it does not only check the specification but also guides the user in building it. Compared with related work, both actions and inhibitions are allowed, resulting in a more generic method. Another strong point is the explicit modeling of permissiveness. The user has no permissiveness requirement to provide and can choose precisely the permissiveness trade-off (provided variable dependency is modeled). By using the template, the modeling approach is scalable to many variables and interventions.

We plan to apply the method on more safety invariants from the KUKA use case. Validation of the method will be done by implementing and testing strategies on the KUKA robot.

References

[1] "ISO/IEC 61508-7: Functional safety of electrical / electronic / programmable electronic safety-related systems - part 7: Overview of techniques and measures," 2010.

[2] A. Cimatti, E. Clarke, A. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani and A. Tacchella, "Nusmv 2: An opensource tool for symbolic model checking," Computer Aided Verification, Springer, 2002.

[3] M. Machin, F. Dufossé, J.-P. Blanquart, J. Guiochet, D. Powell, and H. Waeselynck, "Specifying safety monitors for autonomous systems," in SAFECOMP. LNCS, 2014.

[4] M. Machin, F. Dufossé, J. Guiochet, D. Powell, M. Roy, and H. Waeselynck, "Model-checking and Game Theory for Synthesis of Safety Rules," in HASE. IEEE, 2014.

[5] A. Mekki-Mokhtar, J.P. Blanquart, J. Guiochet, D. Powell and M. Roy, "Safety trigger conditions for critical autonomous systems," 18th. Pacific Rim Int. Symp. on Dependable Computing (PRDC), IEEE, 2012.

[6] KUKA, "Milestone 7: Use Case requirements communicated to other WPs", T8.2-4, SAPHARI, 2012.

[7] LAAS-CNRS, "Milestone 8: Risk analysis of target use cases and safety monitoring completed", T8.2-4, SAPHARI, 2013.

Appendix A: Implemented strategies

Experimental setup

The experimental setup consists in a arm fixed on the table equipped with a gripper (see Figure 11). The monitor is simulated through the Java interface for faster implementation and more flexibility than available in the restricted safety layer. Among all the observation available to the Java interface, we use only the ones that would be available to the safety channel. The only intervention available is the pause, that is a fast stop of the robot because it is the only reaction of the robot that can be guaranteed in terms of true machine safety (e.g., in case the power is cut). In this setup, no values are available as timing bounds for computation and stop, so as to observation accuracy. As a consequence values of margin are determined empirically and not computed.

The table height, the positions of (blue) part and (yellow) box are known.

The task is to pick the blue part and to place it on the yellow box.

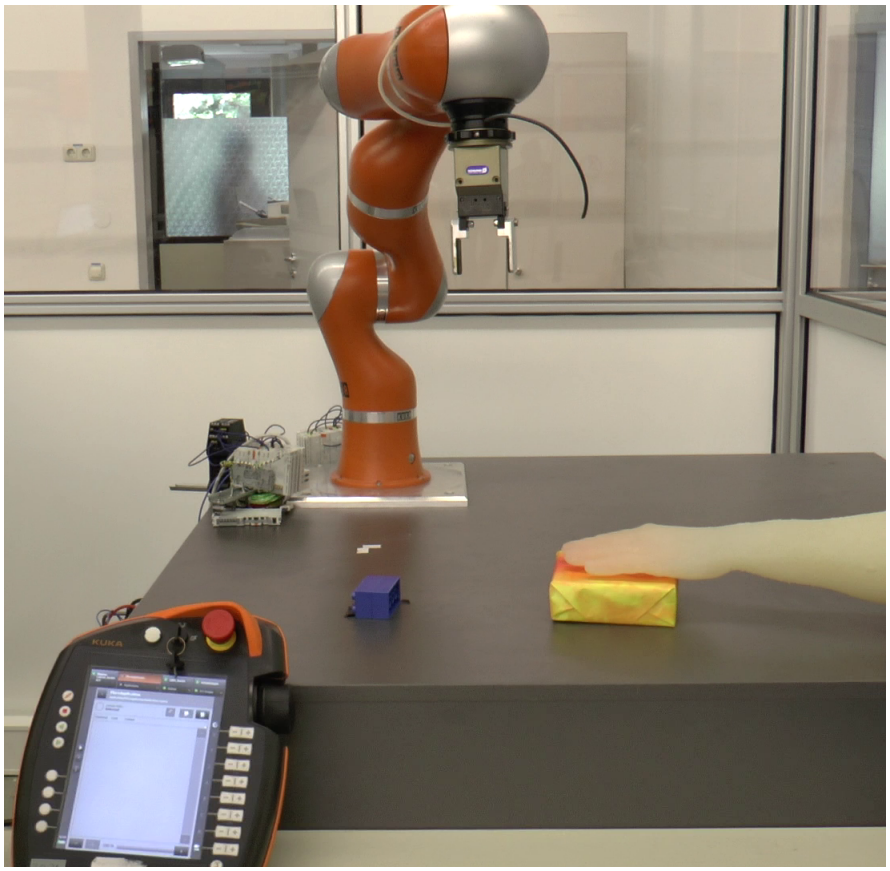


Figure 11: The experimental setup

We implemented three strategies, covering the hazards of collision, clamping and tilting the box.

Collision in free space (SI6)

A collision with high forces is, of course, hazardous. It is observable through the external torque measurements of the LWR. Nevertheless, when the part is placed, an external torque is expected. We consider that an external torque occurring when the gripper is very close to the table is not a collision. This is due to the assumption

that if the gripper is already very close to the table, there can be no hand in between. The discrete model, expressed in the SMV language, is as follow :

```
MODULE Collision
VAR
-- external torque
ext_torque : Continuity(0,2,0); -- 2: normal torque, 1: margin, 0: too high
-- absolute module of velocity
vel : Continuity(0,3,0);          -- 0: standstill, 1: normal, 2:margin, 3:too high
-- z gripper position
z : Continuity(0,2,2); -- 0: on table, 1: margin, 2: high from table

--Dependence: no position change with velocity=0
TRANS vel.v=0 & next(vel.v)=0 -> z.v=next(z.v)

DEFINE cata:= ((ext_torque.v=2 & vel.v!=0) | vel.v=3 ) & z.v=2 ;
```

The only intervention available is the stop. Nevertheless, stop is modeled by several intervention modules in the model, as it has effect on position, velocity and torque at the same time.

The hazard is limited to the zone high from table. It happens when an external torque occurs with a non-zero velocity. From a determined velocity, torque can exceed its limit too fast to be measured in its margin. The role of the sampling is here to be underlined. Consequently, a collision cannot be mitigated at high velocity. We consider high velocity as hazardous.

The synthesis returns one strategy:

```
flag_stop =
  ext_torque.v=1&arm_vel.v=0&z.v=1       -- a first state
| ext_torque.v=2&arm_vel.v=0&z.v=1       -- a second state
| ext_torque.v=1&arm_vel.v=1&z.v=1       -- a third state
| ext_torque.v=2&arm_vel.v=1&z.v=1
| ext_torque.v=0&arm_vel.v=2&z.v=1
| ext_torque.v=1&arm_vel.v=2&z.v=1
| ext_torque.v=2&arm_vel.v=2&z.v=1
| ext_torque.v=0&arm_vel.v=3&z.v=1
| ext_torque.v=1&arm_vel.v=3&z.v=1
| ext_torque.v=2&arm_vel.v=3&z.v=1
| ext_torque.v=1&arm_vel.v=0&z.v=2
| ext_torque.v=2&arm_vel.v=0&z.v=2
| ext_torque.v=1&arm_vel.v=1&z.v=2
| ext_torque.v=0&arm_vel.v=2&z.v=2
| ext_torque.v=1&arm_vel.v=2&z.v=2
```

Discrete models have been introduced as finite state machines (see Figure 4). Each line is a state of the machine ; each state is defined by a combination of values of variables. The states mentioned here are the warning states, i.e., the states from which one step is enough to reach a catastrophic states. The first state corresponds to: the torque close to the safety limit (in the margin interval modeled by the value 1), the velocity is zero (standstill), the gripper is close to the table (in the margin interval modeled by the value 1). The trigger flag of the intervention `stop` is an OR of every line, i.e., it is set to true for all in every warning states. This strategy (apply `stop` in the mentioned states) is obtained fully automatically from the discrete model.

To test this strategy, a collision is done by an external user (see Figure 12). The robot stops satisfyingly in case of collision. The monitored external torque does not exceed the safety limit. As the setup does not include an external sensor, the measure cannot be validated by an independent measurement. As expected, collisions when the gripper is close to the table are not covered.

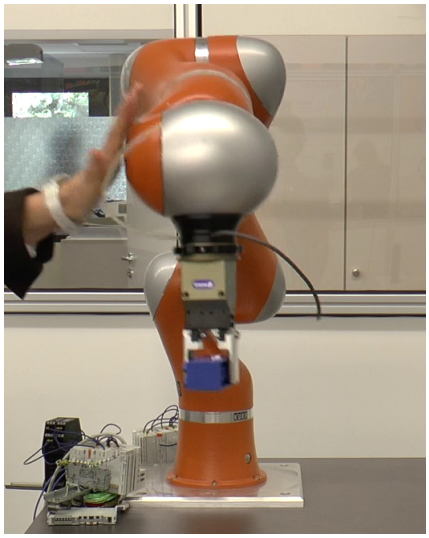


Figure 12: Collision experiment

Clamping (SI16)

To avoid clamping, the z-axis force at the tool point must not exceed a limit when the tool is close to the table. We model the hazard as following:

```
-- Model
MODULE Clamping

VAR
tcp : Continuity(0,2,0);
vel : Continuity(0,2,0);
z : Continuity(0,2,2);

DEFINE cata:= z.v=0 & (tcp.v=2 | vel.v=2);
```

The state is hazardous when the gripper is close to the table (z=0 with z the discrete height from the table) ; and when the force (tcp) is high. In case the velocity is high, the force measure increases too fast to enable the monitor to react. Then, we add a high velocity as a hazardous case. We make here the assumption that the whole table surface is as soft as the box, enabling to measure the torque increasing slowly enough to be detected before it reach the safety limit. The synthesis returns one strategy:

```
flag_stop =
  tcp.v=1&vel.v=0&z.v=0
| tcp.v=0&vel.v=1&z.v=0
| tcp.v=1&vel.v=1&z.v=0
| tcp.v=1&vel.v=0&z.v=1
| tcp.v=2&vel.v=0&z.v=1
| tcp.v=0&vel.v=1&z.v=1
| tcp.v=1&vel.v=1&z.v=1
| tcp.v=2&vel.v=1&z.v=1
| tcp.v=0&vel.v=2&z.v=1
| tcp.v=1&vel.v=2&z.v=1
| tcp.v=2&vel.v=2&z.v=1
```

The strategy applies stop in every listed state.

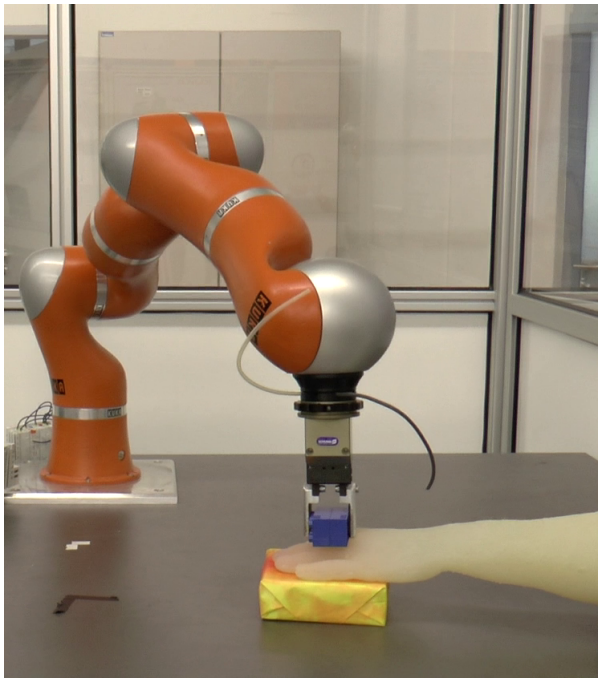


Figure 13: Clamping experiment

To test this strategy, a fake hand is put on the box. When placing the part on the box, the robot begins to apply a force on the arm. The monitor triggers a stop (see Figure 13), and so the force limit is not exceeded. It must be underlined that this strategy does not prevent the robot to place the part on box when there is no hand, i.e., when the external force occurs in the expected placing position. As a consequence the strategy is permissive.

Tilt box (SI1)

The robot is supposed to manipulate boxes, filled with parts. A hazard arises if some parts fall from the box during manipulation. In particular it will happen, if the robot tilts the box too much. Once more the hazard is not considered very close to the table, as the box could simply not be tilted.

```
VAR
alpha : Continuity(0,2,0);
d_grip : Continuity(0,1,0);      -- 0: open, 1:is not open
z : Continuity(0,2,0);

VAR
box: 0..1;
ASSIGN init(box):=0;
next(box):= case
        d_grip.v=0 & next(d_grip.v)=1 & z.v=0: 1;
        next(d_grip.v)=0 : 0;
        TRUE : box;
        esac;

--!! Specify cata with state variables
DEFINE cata:= alpha.v=2 & box=1 & z.v=2;
```

The hazardous state is: a box is present in the gripper, the gripper position is such that the angle (in x and y axis w.r.t. the table) is high, and the gripper is not close to the table. Nevertheless, the box presence is not

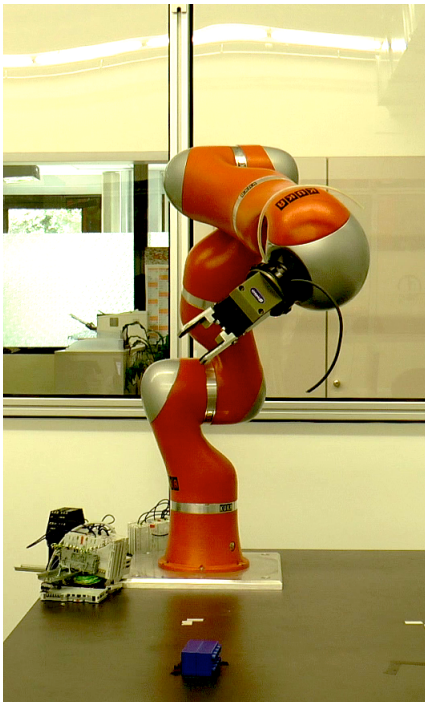
directly observable. The load observation could be used. However the monitor is only able to recognize some predetermined load values. As a box could be more or less filled, the load observation is not useful. We build a box observer, based on the hypothesis that a box can only be picked on the table, and is released when the gripper opens.

The synthesized strategy is:

```
flag_stop =
  alpha.v=1&z.v=1&d_grip.v=1&box=1
| alpha.v=2&z.v=1&d_grip.v=1&box=1
| alpha.v=1&z.v=2&d_grip.v=1&box=1
```

The strategy applies stop in every listed state.

(a) There is a box in the gripper, the tilt is the maximum safe tilt. The robot is stopped by the monitor. The strategy ensures safety.



(b) In absence of box, the tilt is not constrained, i.e., the robot is not stopped. The strategy is permissive.

Figure 14: Test of the tilt strategy

The strategy is tested as presented in Figure 14.

General remarks

In the Java layer, the implementation of the synthesized strategy is done quite easily. The strategy is used directly. Some observation functions realize the discretization in values 0, 1, ... , from the real thresholds.

The strategies are presented separately. Nevertheless, they run all in parallel, such as the implemented monitor covers the three hazards as the same time.

Appendix B: Preliminary list of safety invariants

Introduction

This appendix presents the safety invariants found by hazard analysis for the KUKA robot (OmniRob platform and LBR arm). The safety invariants are the negation (logical complement) of the hazards. We choose here to express hazards instead of safety invariants as hazards are often more understandable.

This document focuses on hazards, i.e., problems. The solutions to avoid the hazard (and ensure the safety invariant) are the safety rules which are not presented in this document. They are synthesized from the hazards described here in a formal model (.smv files). The safety rules are available in .res files (or in the header of .smv).

Each safety invariant is presented in this document as follows:

- Number. Name
- Description
- The use cases (UC from UML model, presented in [7]) where the hazard is relevant and in which the rules are applied
- Formal expression of the hazard
- Description of the variables and parameters of the formal expression
- Remarks
- Hypotheses or assumptions about the environment, the system, the monitor resources.
- Questions on issues that require clarifications

Variables are denoted in lower case, fixed parameters and thresholds are denoted in upper case. Velocities thresholds are indexed by the object (arm or platform) and the number of the hazard.

Unless otherwise stated, variables are available as AMF (Atomic Monitoring Function, observable variable in KUKA terminology).

Hazards

1. Tilt box

Description A box is in the gripper. If the box tilts too much, parts fall out.

Applied to all UC

Formal expression of hazard $\theta > \theta_{lim} \wedge box \wedge z \in Zint$

with

- θ is the minimum value of cartesian angles B and C (axis x and y) of the end-effector
- z is the z Cartesian position of the end effector. $Zint$ defines an interval just above the table where cannot be tilted because of the table.

- box is an observation of box presence based on sequential observation on z and gripper state (boolean value open or closed)

Remarks May be duplicated to cover two angles (in x and y axis).

2. Collision with platform

Description The robot platform collides with a human.

Applied to all UC

Formal expression of hazard $d = 0 \wedge v_{pf} = 0$

with

- d distance to the closest obstacle (given by laser scanner)

Remarks Taking into account inertia and the fact that stopping takes time, the hazard is modified to express that the collision is unavoidable.

Formal expression of hazard $v_{pf} > \sqrt{2.a.d}$

with

- d distance to the closest obstacle (given by laser scanner)
- A (parameter) maximum deceleration. The value should take into account maximum braking and maximum acceleration from the functional channel and due to environment.

Remarks For technical implementation, square root may be difficult. It is equivalent to use $v_{pf}^2 > 2.A.d$.

Hypotheses

- Every obstacle is sensed, i.e., visible at scanner height.
- d is continuous. This requires that obstacles are sensable as soon as they are in laser range (they can be hidden by sensed obstacle). This hypothesis forbids that obstacles arrive from the ceiling or from the floor.
- Obstacles are fixed, or only the contribution from the robot to the relative velocity is taken into account to define a collision.

3. Arm and platform velocities

Description The platform moves. The arm moves and collides with a human. The collision velocity is the sum of the two velocities.

Applied to all UC

First formal expression of hazard $v_{pf} + v_{arm} > V_3$

with

- v_{arm} maximum velocity among the velocities of the seven axes of the robot arm

- v_{pf} absolute platform velocity

Remarks Such a hazard is modelable by considering $v_{pf} + v_{arm}$ as one variable. In such a model, interventions (brake arm and brake platform) are difficult to model. Furthermore deciding which intervention has to be triggered is difficult too. For these reasons, we have chosen to modify the formal expression of the hazard.

Formal expression of hazard $v_{pf} > V_{p3} \vee v_{arm} > V_{a3}$

with

- V_{p3} and V_{a3} are chosen such as $V_{p3} + V_{a3} < V_3$
- v_{arm} maximum velocity among the velocities of the seven axes of the robot arm
- v_{pf} absolute platform velocity

Remarks The second expression is conservative with respect to the first expression. Every hazardous state in the first version is considered as hazardous in the second version. On the opposite some safe state are not longer permitted.

Hypotheses

- States that satisfy $v_{pf} > V_{p3} \wedge v_{arm} > V_{a3} \wedge v_{arm} + v_{pf} < V_3$ are no longer permitted (even though they are safe). The permissiveness loss is accepted.

Remark May be covered by SI7 and SI3. Depends on modification if SI7.

4. Gripper clamps human hand

Description The gripper fingers close on human part.

Applied to all UC

Formal expression of hazard None

Remarks The gripper is not able to clamp a human part by design. On the one hand, the distance between gripper fingers is clearly larger than a human hand in closed position. On the other hand the distance in opened position is barely larger than the box size such that no human part can be inserted between the box and the finger.

Hypotheses No object in the environment is such that the gripper can clamps a human hand.

5. Workspace sharing

Description Collision in the particular case of workspace sharing in workstation B.

Applied to UC7 (Place box on a table), UC8 (Take box from a table)

Formal expression of hazard $v_{arm} > V_{a5} \wedge human \wedge loc =$ workstation

with

- v_{arm} maximum velocity among the velocities of the seven axes of the robot arm

- loc localization of the robot platform
- $human$ human presence in the workstation. Need sensing in the environment.

Remarks If the human presence is not observable, the condition may be released. It require to accept a permisseveness loss.

Questions Extend the invariant to UC5 (Give box to user), UC6 (Take box from user), UC15 (Manipulate a part in robot gripper)?

6. Collision

Description The robot arm collides something with an high torque.

Permissiveness Exception We exclude from the hazard the region very close to the table to be able to place a box on the table. SI16 covers the region close to the table.

Applied to all UC ? or only collaborative UC ?

Formal expression of hazard $((\sigma_{ext} > \Sigma_6 \wedge v_{arm} \neq 0) \vee v_{arm} > V_6) \wedge z \in freespace$

with

- z cartesian position of the end-effector. 0 value is the table height.
- v_{arm} maximum velocity among the velocities of the seven axes of the robot arm.
- Σ_6 (parameter) threshold of the collision detection.
- σ_{ext} external torque.

Remark The torque comparison is embedded in the collision detection AMF. The external torques are compared with threshold values to detect collisions. External torques are computed from the dynamic model in combination with the joint torque sensor measurements. The collision detection may give false detection in presence of inertial forces due to the platform acceleration/deceleration/turning. Torque cannot be defaultly considered continuous or controllable.

7. Collision with extended arm

Description Collision between a human and the robot arm that is extended beyond the platform footprint while the platform is moving.

Permissiveness Exception (To be added) When the platform is close to the table, platform motion may be needed to take a box.

Applied to UC3 (Go to location), UC4 (Approach user)

Formal expression of hazard $v_{pf} \neq 0 \wedge p_{arm} =$ beyond the platform

with

- v_{pf} velocity of the robot platform. Here the observed variable can be the standstill monitoring.

- p_{arm} is the arm position. The predicate stands for at least one point of the monitoring spheres beyond the platform (use of monitoring spheres and Cartesian workspace)
- loc localization of the robot platform

Remarks The predicate $v_{pf} \neq 0$ is both part of the hazard definition and the expression of the use cases where the invariant is applied.

Hypotheses
- The tool is configured in monitoring spheres

8. Robot in restricted area

Description The robot must not be in restricted area

Applied to all UC

Formal expression of hazard $loc =$ restricted area

with
- loc localization of the robot platform

9. Platform motion overturn boxes on table

Description The gripper takes or places a box on a table. The platform moves, so the gripper moves horizontally at box height. The gripper overturn boxes and sharp parts scatter on table.

Permissiveness Exception When the platform is close to the table, platform motion may be needed to take a box.

Applied to UC7 (Place box on a table), UC8 (Take box from a table)

Formal expression of hazard $v_{pf} \neq 0 \wedge loc =$ workstation $\wedge\, p_{ee} =$ above table

with
- p_e position of the end-effector
- loc localization of the robot platform
- v_{pf} velocity of the robot platform.

Remarks This hazard is a restricted version of hazard #7. It will be covered by the rules of #7.

Questions Extend the invariant to UC1 (Take box from a shelf), UC2 (Place box on a shelf)?

10. Excessive arm velocity

Description In general, the arm velocity must be limited.

Applied to all UC

Formal expression of hazard $v_{arm} < V_{a10}$

with

- v_{arm} maximum velocity among the velocities of the seven axes of the robot arm

11. Drop box

Description A box is in the gripper and is dropped anywhere but on table, robot storage, shelf or during handover.

Applied to all UC

Formal expression of hazard

$$load = \text{box} \wedge next(load) = \text{nobox} \wedge \left(p_{ee} \neq \text{robot storage} \wedge \left(p_{ee} \neq \text{shelf} \vee loc = \neg\text{storage area}\right) \wedge \neg(?\text{handover}?)\right)$$

with

- p_{ee} position of end-effector. Drop zone is defined by a protected workspace on end-effector position (and not on monitoring spheres).
- loc localization of the robot platform
- $load$ is from the load determination

Remarks This hazard is a transition rather than a state.

Hypotheses

- The gripping is done by form (and not by force).

Questions

- How can the handover situation be observed?
- How can the position of shelf and table with respect to the platform be observed?

12. Excessive platform velocity

Description In general, the platform velocity must be limited.

Applied to all UC

Formal expression of hazard $v_{pf} > V_{p12}$

with

- v_{pf} absolute platform velocity

13. Excessive arm velocity during interaction

Description During physical interaction, the arm velocity must be limited.

Applied to UC5 (Give box to user), UC6 (Take box from user), UC15 (Manipulate a part in robot gripper)

Formal expression of hazard $v_{arm} < V_{a13} \wedge loc =$ **quality control area**

with

- loc localization of the robot
- v_{arm} maximum velocity among the velocities of the seven axes of the robot arm

Hypotheses

- Interaction is done only in the quality control area.
- In quality control area, no task requires $v_{arm} > V_{a13}$.

14. Excessive platform velocity during guiding

Description The user moves the robot out of his way by guiding it (applying a force on robot arm). The platform velocity must be limited during this step.

Applied to UC9 (Move the robot out of the way)

Formal expression of hazard $v_{pf} > V_{p14} \wedge$ (uc=9)

with

- v_{pf} absolute platform velocity

Remarks How to detect UC9 (Move the robot out of the way)?

Questions How can UC9 (Move the robot out of the way)be detected ? The guidance position of the arm is not satisfying from a safety point of view. What is the real hazard of this case ?

15. The handguiding is not followed

Description In the quality control workstation, the user manipulates a box in the robot gripper by handguiding the robot arm. The robot arm does not follow the handguiding.

Applied to UC15 (Manipulate a part in robot gripper)

Formal expression of hazard To Be Determined

Hint Using external force as an image of it the robot guided or not. Problem: external torque applied by the human go through 0 values. Taking delayed aver age value. Apply a stop if no more handguiding.

Questions Here the monitor lacks both observation and intervention. It is neither able to observe the handguiding, nor to force the arm to move.

16. Clamping

Description The end-effector clamps a human part on table or robot storage. This hazard is effective only when the end-effector (empty gripper or box) is close to table.

Applied to UC7 (Place box on a table), UC8 (Take box from a table), UC10 (Place box on robot storage), UC11 (Take box from robot storage)

Formal expression of hazard $TCP_z > F_{16} \wedge p_{ee} \in W_{16}$

with

- W_{16} is a workspace close to table, robot storage... W_{16} may not touch the clamping surface.
- TCP_z Cartesian external force at the end-effector along z axis. The external force is computed from the dynamic model in combination with the measured joint torques. It can be considered continuous if the velocity is strongly limited.

Remark To be able to distinguish between clamping force (in z-axis) and placement force (in x and y axes), the path should be purely vertical.

Remaining hazards

One hazard is not currently completely covered by the invariants. When the arm is taking or placing a box on table, the gripper can overturn boxes due to a horizontal movement (if the movement is due to the platform, it is covered by #9). There other similar situations: instead of placing the gripper fingers on both sides of a box, it can place fingers in boxes.

The monitor may not be able to entirely cover this hazard due to the lack of observability (it does not know where boxes are) and lack of intervention (forbidding a horizontal movement and allowing a vertical movement is not possible). This hazard is only of moderate severity.

General environment hypotheses

- The robot is always on a flat floor. There is no slope in the allowed area of the robot.