THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Towards the safe-IOCOS relation

Adnan Khan

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2021

Towards the safe-IOCOS relation

Adnan Khan

ISBN 978-91-7905-467-0
Series Number. 4934
in the series Doktorsavhandlingar vid Chalmers tekniska högskola. Ny serie (ISSN 0346-718X)
This thesis has been prepared using LaTeX.

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000
www.chalmers.se

Printed by Chalmers Reproservice
Gothenburg, Sweden, March 2021

To my parents.

Abstract

The technological advancement that has occurred at a blistering pace in the past decades has enabled manufacturing companies to conceive innovative products. However, to meet growing demands of consumers, manufacturing companies are expected to maintain a steady production rate without compromising product quality. To meet these requirements, the industrial sector is increasingly using robots and other automated machinery.

Automated machines are predominantly controlled via programmable logic controllers (PLCs) to carry out the nominal tasks. For safety critical tasks, though, special devices in conjunction with safety PLCs are used to prevent material damage and accidents leading to human injuries.

Before physical commissioning of a manufacturing system, the nominal PLC code is tested to uncover faults. This can be done either by running tests on the physical system or using a simulation model via *virtual commissioning*. However, the safety code is usually tested during the *factory acceptance test* phase on the actual physical system. The faults found in the safety code are corrected manually, which is time consuming and error prone.

The formal methods community has developed testing relations and approaches that can be used to automatically test and amend faults in the implementation. The work presented in this thesis is based on such a testing relation and formal approach.

The *safe input-output conformance simulation relation* (safe-IOCOS) is a testing relation that requires equality for traces composed of safety behaviors. However, in practical settings, many safety behaviors in a production system are implemented for each nominal operation. And these behaviors get tested multiple times during testing, which increases the testing time unnecessarily. To counter this problem, an approach to minimize testing time is proposed.

Furthermore, an approach to automatically amend a faulty implementation to ensure safety properties with respect to a safety specification is presented. This approach uses the procedure of *synthesis*, from the framework of supervisory control theory, based on the *infimial controllable superlanguage*, which not only removes the faults from the implementation but also guarantees to make it safe-IOCOS.

Keywords: Safety, PLC, Supervisory control theory, Model-based testing, Input-output conformance simulation relation.

List of Publications

This thesis is based on the following publications:

[A] **Adnan Khan**, Petter Falkman, Martin Fabian, "Testing and Validation of Safety Logic in the Virtual Environment". CIRP Journal of Manufacturing Science and Technology, Accepted.

[B] **Adnan Khan**, Sahar Mohajerani, Martin Fabian, "On test case reduction for testing safety properties of manufacturing systems". Journal of Manufacturing Systems, Submitted.

[C] **Adnan Khan**, Martin Fabian, "On testing and automatic mending of safety PLC code". CIRP Journal of Manufacturing Science and Technology, Invited for resubmission after revision.

Other publications by the author, not included in this thesis, are:

[D] **Adnan Khan**, Petter Falkman, and Martin Fabian, "Virtual engineering for automatic generation of control logic including safety". *13th IEEE Conference on Automation Science and Engineering (CASE)* Xian, China, August. 2017.

[E] **Adnan Khan**, Martin Dahl, Petter Falkman, and Martin Fabian , "Digital Twin for Legacy Systems: Simulation Model Testing and Validation". *14th IEEE Conference on Automation Science and Engineering (CASE)*, Munich, Germany, August. 2018.

[F] **Adnan Khan**, and Martin Fabian, "On the Equivalence of Controllability and the Input Output Conformance Testing Relation". *Report*, August. 2018.

[G] **Adnan Khan**, David Thönnessen, and Martin Fabian, "On-the-fly conformance testing of safety PLC code using QuickCheck". *IEEE International Conference on Industrial Informatics(INDIN)* Helsinki-Espoo, Finland, August. 2019.

[H] **Adnan Khan**, and Martin Fabian, "On the Safe IOCOS relation for Testing Safety PLC Code". *24th IEEE Conference on Emerging Technologies and Factory Automation(ETFA)* Zaragoza, Spain, September. 2019.

Acknowledgments

First of all, I would like to thank none other than Martin Fabian for his guidance, valuable feedback, and support during the whole journey. I have learnt many things from you and I am forever thankful for the kindness you have shown towards me. Whenever I felt down, you were there to motivate me and I believe you have a lot to offer as a teacher and as a supervisor. Thanks for everything.

I also would like to thank Petter Falkman for sharing his thoughts, knowledge, and experience during the initial years of my PhD. Although I changed my research direction after my licentiate, I still feel that your constructive feedback in the initial years of my PhD kept me motivated throughout.

Special thanks to Sahar, you joined at a later stage of my PhD journey. However, your ideas and suggestions has helped me to further my research.

Thanks to all other group members in the Automation research group. Especially, Ashfaq, Fredrick, and Martin Dahl, your friendly aura really made me comfortable at Chalmers and I feel blessed to have such good office mates.

Sarmad, I m not sure if I have told you this before that I am really impressed by your soft-skills. I think you have all the ingredients to be a great teacher and I wish you all the best in your future endeavours.

Finally, a big thanks to everyone in the administration. Especially, Christine and Natasha, you guys have been really helpful and kind throughout my time at Chalmers.

Adnan Khan
Göteborg, March 2021

This work has been carried out at the Wingquist Laboratory VINN Excellence Centre within the Production Area of Advance at Chalmers. It has been supported by VR SyTeC (ref 2016-06204). Department of Electrical Engineering, Chalmers University of Technology, Göteborg, Sweden

Acronyms

BSC:	Broadcast synchronous composition
FAT:	Factory acceptance testing
FSC:	Full synchronous composition
IOCO:	Input-output conformance
IOCOS:	Input-output conformance simulation
MBT:	Model-based testing
PLC:	Programmable logic controller
PSC:	Prioritized synchronous composition
Safe-IOCOS:	Safe input-output conformance simulation
Safety PLC:	Safety programmable logic controller
SCT:	Supervisory control theory
VC:	Virtual commissioning

Contents

Part I

Overview

CHAPTER 1

Introductory chapters

1.1 Introduction

Life of modern man compared to our ancestors is more luxurious, for which we should thank science and technology. All the inventions of this modern age e.g. cars, blenders, internet, cell phones etc., became a reality due to evolving technologies. However, we rarely think about how these products are initially conceived, manufactured, and eventually produced.

The process of manufacturing involves interaction of various machines in an orderly manner under the command of *programmable logic controllers*, PLCs. These PLCs control the nominal tasks of machines, like moving a part from one place to another. However, when it comes to safety, ordinary PLCs are rather limited due to the absence of redundancies etc. [1]. If safety is compromised either due to improper implementation of the PLC program or absence of safety measures, it can result in scrapped material, wrecked machines, or in the worst case human injuries or even loss of lives.

To safeguard the whole manufacturing system and prevent human accidents, safety devices are installed at various locations in the production facility. Some examples of safety devices are floor scanners, pressure relief valves, emergency

stop buttons etc. These devices are coupled with a special controller called a *safety PLC*, whose sole task is to prevent the system reaching an unsafe state even in case of device failures.

In critical situations, the safety PLC takes control of the system via special features and redundancies, which ensures safety even in case of hardware failures. This characteristic of hardware is called fail-safe property, and is integral to the central processing unit, and the outputs inputs of a typical safety PLC [1].

Typically, PLC code is tested before physical commissioning of a real production system using an approach called *virtual commissioning* (VC) [2]. The idea is to uncover faults using a simulation model of a real production system. However, the application of VC is often limited to the testing nominal PLC code.

The safety PLC code on the other hand is tested on real systems in conjunction with the safety sensors and equipment. This process is attributed to as *factory acceptance testing* (FAT), and can take days and even weeks if the production system is large [3]. To carry out tests during FAT, various checklists are used as references.These contain all the details of tests and their expected outcomes.

In contrast to the engineering community, which often uses real systems and checklists for testing safety, the academic community typically uses formal methods [4]. The formal approach enables interaction, verification, testing and validation of system properties with the help of formal models instead of real systems.

Whether testing is carried out using real systems, formal models (simulation or other), the purpose is to uncover faults in the implementation. However, depending on the complexities and size of the implementation, time spent for testing may increase.

Also, one must make sure that test sequences are generated in such a manner that they cover all input-output combinations. At the same time trivial testing sequences, which test the same input-output combinations multiple times must be avoided as they unnecessarily lengthen the testing process. This increase in time basically hampers the main purpose of VC, which is to cut down physical commissioning time [5].

Furthermore, when it comes to amendment of a faulty implementation, this is typically carried out manually, which is prone to human error. Especially, in

the industrial sector, where formal approaches and methods are not frequently used. So, engineers need to first find faults in the PLC code and then have to carry out amendment based on different formal and semi-formal drawings and documents.

This thesis outlines research carried out to address problems that are related to testing and amendment of an implementation, with the focal point on safety. Paper A presents an approach to bolster the current VC process. In Paper B, a formal approach to reduce testing time for an implementation by reduction in test cases is presented. Paper C, illustrates an approach to automatically amend a faulty implementation to ensure safety.

1.2 Background

A typical manufacturing system consists of a physical production system that is controlled via one or more PLCs. This closed-loop setup of the physical system with the PLCs we will call an *implementation*, and this includes all installed mechanical and electrical systems and sub-systems, together with the control systems that make them interact in a productive way.

Before the installation of a complete physical system, all its components along with the PLC code are first tested in the FAT phase for faults. The focus of testing during FAT is predominantly on safety-instrumented systems [6] and its related PLC code. This testing is typically carried out on the basis of some *specification* to asses if the implementation is free of faults. The presence of faults basically impedes the implementation to conform to the specification causing implementation to behave in an erratic manner. These faults if not uncovered can be potentially harmful to both the production system and humans.

The faults can either be in the hardware, e.g. in mechanical joints, piping, electrical wiring etc., or it can be in the software, the PLC code. An important point to note here is that testing does not guarantee absence of faults in the system but the purpose is to find faults to prevent material and human damage. Typically, the tests carried out during FAT are on individual machines at a very basic level as well as in detail on the whole production system [6].

After the installation of all machines in their right place per design documentations, *commissioning* [7] activities are carried out. Commissioning of

a physical system is a process that is carried out to validate the complete implementation. This means that the machines that were tested during FAT will now be tested in conjunction with other machines to see if they carry out the required tasks by interacting with other machines in a productive manner.

During commissioning, the testing is usually done by running various machines without the introduction of raw materials. The purpose of testing during the commissioning phase is again to uncover faults in the PLC code as well as hardware. However, this time the testing activities are expanded to cover all sub-systems. This means that during commissioning, the PLC code and its interaction with other units is also tested for faults.

To perform tests, a test engineer applies various stimuli (inputs) to the implementation and observes the resulting behavior. The observed behavior is compared to the requirements mentioned in the design documentation. If the observed behavior does not match with the specified requirements then this is registered as a fault.

When testing is carried out during the commissioning phase, typically the test engineer is oblivious to the details of the implemented PLC code and the electrical wiring. This so since mostly these activities are carried out by various contractors involved at different stages of development prior to the commissioning phase. Therefore, for a the test engineer, the whole implementation is regarded as a *black-box* [8].

Due to the similarity with black-box testing, testing of an implementation during commissioning can be related to the formalized approach of *model-based testing* (MBT) [9]. The MBT approach subjects an implementation to different tests that are derived from a specification model to find faults.

The conformance of the implementation depends on the formal approach used for testing. Some formal approaches [10], [11] are fine with the implementation being partially conformant to the specification. However, for testing safety properties, stronger notions of conformance are needed. In this work safe-IOCOS [12] is used, which requires the implementation to emit *all* specified behaviors with respect to the safety requirements.

1.3 Research questions

The following research questions, all related to testing an implementation for conformance to a safety specification, are addressed.

RQ1:

How can testing of safety PLC code be carried out using a simulation model to support FAT and strengthen the existing virtual commissioning practices?

The safety PLC code is typically tested during the FAT phase, which is a part of pre-commissioning activities of a production system. The PLC code for the nominal behavior, on the other hand, is typically tested using a simulation model to cut down the physical commissioning time. This segregation in terms of testing is an impediment to making full use of virtual commissioning. If safety PLC code testing is integrated in the virtual commissioning domain, then further reduction in commissioning time can be expected, which may enable shorter development time of production systems.

RQ2:

How can the time required for testing an implementation for safety be reduced by minimizing test cases for safe-IOCOS without compromising its intrinsic properties?

For the safe-IOCOS conformance relation, an implementation is tested for all possible traces in the safety specification. However, in many cases, various safety inputs and outputs that are associated with every nominal activity are tested multiple times. Furthermore, if the implementation contains non-determinism then the test engineer may end up testing the same behavior each time a non-deterministic trace is executed. This not only adds to the time in testing but also makes much of the testing mundane and trivial. To mitigate these problems, the number of test cases should be reduced in such a way that the intrinsic properties of safe-IOCOS are preserved.

RQ3:

How can an implementation be automatically amended to remove faults to ensure safety without disturbing the segments of implementation that are not under scrutiny?

After testing, the uncovered faults in an implementation are typically removed manually, which is a error prone process in itself and also time consuming. And this time may increase if there are numerous modifications that are required to be carried out in different segments of an implementation. An important aspect to keep in mind while carrying out such modifications is to

preserve the segments of implementation that are not under scrutiny. And this aspect is of more importance when safety properties are being tested, as any accidental amendment in the implementation that should be preserved may result in human accidents or system damage.

1.4 Contributions:

The primary contributions of this thesis are as follows:

- A formal approach to test safety PLC code using a simulation model to support the FAT phase and current virtual commissioning practices. This approach is detailed in Paper A, which attempts to give some answers to *RQ1*.
- A formal approach to abstract test cases from safety specification to reduce testing time for the given implementation. Paper B details this approach, which addresses the problem pointed out in *RQ2*.
- A procedure to amend an faulty implementation to ensure safety with respect to safety specification. This procedure is elaborated in Paper C, which addresses the problem highlighted in *RQ3*.

1.5 Method

The research method used to answer the research questions is a combination of discussions with different company personnel, literature search, and experiments. In this section, the method adopted to answer each research question is detailed.

Method adopted for RQ1

For the first research question, *RQ1*, initially discussions were carried out with engineers working at various manufacturing companies [13]. The idea of conducting discussions and interviews is to highlight the issues and expectations associated with VC. Alongside these discussions, relevant literature search was carried out to asses the current practices related to VC.

After discussions and literature study, the idea of testing safety PLC code in a virtual environment surfaced. However, to further strengthen the VC process for safety PLC code, a need to incorporate a formal approach was felt. Addition of a formal dimension in conjunction with visual validation based on a simulation model increases the reliability in the absence of real safety devices and machines.

Now, the challenge was to find a formal approach that is simple and fits with the PLC behavior. For this purpose, a formal testing relation IOCO [10], [14] stood out during the literature search. IOCO is a testing relation that assesses an implementation based on the emitted outputs. This behavior can be easily related to a PLC, because a typical PLC sends output to control different devices with respect to the inputs received from sensors.

Finally, to validate the proposed approach for testing safety PLC code in a virtual environment, a simulation model was built, based on the CAD models received from a company. Based on these CAD models, the simulation model along with PLC code for the nominal and the safety behavior was created. For test cases, a virtual human is triggered via the *human-machine interface* (HMI) to disrupt the ongoing operation in the cell.

Method adopted for RQ2

For the second research question, *RQ2*, a theoretical approach was adopted, that started with examining the formal definitions of IOCO and IOCOS [11] in conjunction with the bisimulation equivalence relation. This was then extended in terms of deterministic and non-deterministic cases. Then the approach was built to reduce the number of test cases for testing an implementation according to safe-IOCOS using bisimulation. This approach is then further enriched by including the subset construction method. Experiments were carried out in the tool Supremica [15] that indicated the benefit of the approach.

Method adopted for RQ3

The idea to automatically amend a faulty implementation arose when the faulty safety PLC code that was implemented for *RQ1* was amended manually.

Then, in regard to automatic amendment, a literature survey was carried out to find a formal process that could be used to amend the implementation

in a selective manner, i.e. only faulty segments of the implementation should be modified.

In the first round, *prioritized synchronous composition* (PSC) was tried. However, after several experiments, it was found not to be useful in solving the problem mentioned in *RQ3*, since other segments of the implementation were getting disturbed during the amendment process.

For the second round, the infimial language extension [16] was tried in the tool *Desuma* [17]. However, similar to PSC, in some cases the infimial language computation affected other segments of the implementation that were not to be affected.

To solve the problem associated with the above mentioned undesired effects of the infimial language extension, some experiments were carried out in Supremica after segregating the states that required amendment due to presence of faults from the states that do not require any amendment in the implementation. This segregation enabled the infimial language computation to only affect states relevant to the safety specification. As a result, faults present in the implementation were removed and the rest of the implementation remained in its original form.

1.6 Thesis outline

This thesis consists of two parts. Part I is a general introduction to the field and puts the appended papers into context. Part II contains the appended papers. Part I is organized in the following manner: Chapter 2 discusses the challenges faced by companies related to virtual commissioning and how it can be addressed. Chapter 3 gives a brief introduction to the formal methods and model-based testing approach. Chapter 4 gives a brief introduction to the research questions and how they are addressed in the appended papers. Chapter 5 summarizes the included papers. Finally, Chapter 6 gives some directions for future work.

CHAPTER 2

Virtual Commissioning

Virtual commissioning (VC) is a concept that was developed to support physical commissioning of a production system. The main motivation behind VC is to reduce physical commissioning time [2], [5] by uncovering faults in the PLC code prior to physical commissioning. This is important since about 70% [5] of the physical commissioning time is consumed by software debugging.

To identify these faults prior to physical commissioning under the umbrella of VC, a simulation model is built based on the available data. This data is in the shape of various mechanical and electrical drawings, and control specification documents, etc. Now, this simulation model is expected to represent the true behavior of the real production system, and any fault in the modeling of the simulation model may result in false testing of the PLC code.

False conclusions derived from the PLC code testing during VC may result in machine damage if the erroneous PLC code is deployed on the real production system. This also nullifies a key advantage of VC, which is to protect real systems from damages when testing is carried out. Therefore, its necessary to have a simulation model, which mimics the required behavior of the real production system.

After creating the required simulation model, engineers create the PLC

code, which is tested against the simulation model. Sometimes this code is created *in-house*, by engineers within the company. This is commonly seen in cases where the size of production system is small and if a new section within the existing production system has been implemented. However, in most cases, this part is outsourced to contractors. This is particularly true in cases when a brand new production system is being implemented.

In terms of VC, the created simulation model can be commissioned either using a *soft-PLC* [18], which is a virtual PLC, or a real standard PLC. In the next step, the input-output connections are set up using an emulator that emulates the input-output behaviors in the absence of real input-output cards in the PLC. An example of such a setup is given in [19].

Now, the assessment of the implemented PLC code can be carried out using a simulation model. This assessment is done by observing the behavior of the simulation model in relation to the executed PLC code. The noted observations are then compared with the desired specified behavior.

In case of discrepancies between the specified behavior and the observed behavior of the simulation model, the implemented PLC code is examined to uncover the faults. An important point to note here is that the simulation model is considered correct hence cannot be blamed for the unspecified behavior. Therefore, attention to detail is a necessity while creating the simulation model.

Mostly, the uncovered faults in the PLC code are manually corrected by engineers, that means finding the faulty unit in the implemented code and carrying out the required modifications. The time spent during testing and amendment depends on the size of the production system.

Typically, the whole VC procedure is carried out to test the PLC code associated with the nominal behavior of a production system. Nominal behaviors are related to nominal operations of the production system, those operations that make the system produce whatever it is intended to produce.

For the testing of safety PLC code, using the VC setup is not common practice and safety testing is typically pushed to the FAT phase of the engineering life cycle. The safety PLC code basically prevents the production system from going to forbidden states that could lead to either material or human damage. Some examples of safety operations are emergency shutdown and pausing of a robot in the case of human presence.

The VC phase is considered as the last step before physical commission-

ing [20], and all the engineering activities before VC are carried out in an isolated manner from each other with little or no information reuse. These activities include CAD modeling, kinematic modeling, robot simulations etc.

Due to this lack of information reuse, the simulation model that is required for the VC phase is typically built from scratch. This practice is not only prone to human errors but also adds more time as the simulation model is created manually by engineers. Additional time spent for the creation of a simulation model basically hampers the whole concept of VC that is to cut down physical commissioning time.

To counter this issue, the basic concept of VC needs enrichment that enables information reuse among all steps of the engineering tool chain. However, the use of different proprietary software at each different steps in the engineering tool chain hinders the information reuse and needs to be standardized.

A concept that enables integration of different engineering steps is proposed by the authors of [20]. The proposed approach is about parallel development of the simulation model, which gets enriched at different engineering steps as the project progresses. A similar concept is proposed in [21], where formal methods are used that enable information reuse among different steps in the engineering tool chain.

Formally specified information can be used by computer algorithms, which enables automatic generation of models. This reduces the effort of the engineers that they have to put in for the creation of the simulation model and also the faults associated with manual creation. An approach, where plant data is used for automatic creation is presented in [22], where the data exchange among tools is carried out using standard XML format.

The reason behind extending the creation of a simulation model in the earlier phases of the engineering tool chain is to further reduce physical commissioning time. If the testing starts earlier then ideally the faults in the PLC code can be amended earlier. However, for that reason, the development of the PLC code needs to begin earlier too.

Furthermore, if safety PLC code testing is incorporated in the earlier engineering phase then it can help to support the FAT phase and further reduce the physical commissioning time. However, for this reason, the created simulation model needs to be enriched with all the details related to the safety aspects of the actual production system. If any detail is left out during the modeling process then the test results cannot be fully trusted.

The addition of safety aspects in the simulation model along with the development and testing of safety PLC code in the earlier engineering phases adds more burden on the engineers. And the need for automatic practices for the creation of simulation model and PLC code testing can be felt.

To support the concept of automatic generation of simulation models and PLC code along with safety, a concept of ready-made models provided by manufacturing companies has been discussed for some time now. A similar concept of ready-made components called *clever-components* based on cloud-based repositories are presented in [23].

The authors of [24] proposed a conceptual procedure on the development of such ready-made components. The proposed procedure is based on the concept of *functional-mockup units* (FMUs) and how manufacturers can provide these components on a *functional-mock-up-interface* (FMI) standard [25]. The use of the FMI standard, which is an open and free standard, allows information exchange among the component models provided by manufacturers. This information exchange can help in enabling these provided models to be used directly in the VC model.

From the above discussion, it can be concluded that the time spends for creating a simulation model and PLC code can be saved if companies provide ready-made models for components. However, information formalization is required to enable automatic generation of models and the PLC code. For that, formal methods domain can be explored.

CHAPTER 3

Formal Methods

Formal methods [4] are techniques used to mathematically specify, verify, develop, and test software and hardware.

When using formal methods, we typically deal with formal models of physical systems. The behavior of physical systems along with the controller is in our case predominantly discrete in nature and hence can be modelled as *discrete event systems* [16].

Discrete event systems occupy at each time instant one of multiple *states*, and transit between the states on the occurrence of *events*. In a particular state, certain conditions hold, and on the occurrence of an event all or some of these conditions change.

If the system on the occurrence of a single event can transit to multiple states then the system is said to be *non-deterministic*. However, non-deterministic systems can always be transformed into deterministic system by using, for instance, the*subset construction method* [26].

The interaction of discrete event systems can be modelled using different formalisms, such as *full synchronous composition* (FSC), *broadcast synchronous composition* (BSC), and *prioritized synchronous composition* (PSC) [27].

FSC requires the simultaneous occurrence of events shared between the

systems participating in the composition. Thus, a shared event is blocked if it cannot happen in one of the systems from the current state. The BSC on the other hand does not allow blocking of any event, therefore, any of the participating systems in the composition can emit events independently. PSC is a combination of FSC and BSC. In PSC, some events are *prioritized* and synchronize according to FSC, while other events are *un-prioritized* and synchronize according to BSC.

The selection of these compositions varies with respect to the nature of the application. For example in the *supervisory control theory*(SCT) framework, FSC is used for the computation of the supremal controllable sublanguage $\mathbf{sup\,C}(G||K_x)$ as well as the infimial controllable superlanguage $\mathbf{inf\,C}(G||K_x)$.

Computation of $\mathbf{sup\,C}(G||K_x)$ is typically carried out to restrict the plant within the prescribed behavior of specification . This restriction is imposed by eliminating certain transitions i.e. *uncontrollable events* in the computed language of the controller that leads to unsafe states in the system.

$\mathbf{inf\,C}(G||K_x)$, on the other hand computes the language of the controller by extending the specification within the confines of the plant. This enables the *implementation*, which is the controlled plant, to carry out more activities. This freedom to the implementation is given by adding certain uncontrollable events in the specification that were missing before the $\mathbf{inf\,C}(G||K_x)$ computation.

After computing the controller's language, typically testing is carried out. The purpose of testing is to find faults and unlike formal verification it does not guarantee correctness. In formal methods, *model-based testing* [9] is a well known approach that tests an implementation with respect to a specification.

The *input-output conformance simulation* (IOCOS) [11] is a simulation relation under the umbrella of *model-based testing* [28] that subjects an implementation to sequences of tests generated with respect to a specification.

Compared to the input-output conformance relation (IOCO) [14], IOCOS is a stronger relation as it embodies the basic property of IOCO and enriches it with more details. Where IOCO assesses an implementation with respect to the emitted outputs in relation to a specification, IOCOS considers both outputs and inputs in relation to the specification.

A simulation relation is a uni-directional relation that is established when one system simulates another system in terms of certain behaviour. Another possibility is to have a bi-directional relationship between two systems, where

both systems simulate each other's behavior. This is called a bisimulation relation [29].

A bisimulation relation is an equivalence relation that considers two states equal if the future behavior of two states are exactly the same. States having same future behavior can be merged into a single state as they are considered *bisimilar*. This property enables application of bisimulation relation even for a single system as two bisimilar states within a system that have the same future behavior can also be merged.

To show the viability, many formal approaches have been used on real applications. For example, [30] describes a work procedure to develop formal specifications for safety PLCs, [31] proposed a method on how conformance testing results can be applied on industrial controllers by translating Grafcet into equivalent Mealy machines. Furthermore, [32] presents an approach to verify safety applications based on the PLC-open standard [33].

The application of different formal methods on real applications shows promising results in the area of formal specification, verification, and testing. However, this work presented in this thesis is mainly focused on the model-based testing approaches, i.e. IOCO and safe-IOCOS, which will be discussed in the next section.

3.1 Model-based testing

Model-based testing (MBT) [9] is a formal approach that subjects an implementation to various tests that are abstracted from a given specification model. If the implementation conforms to the specification for the executed tests cases, either partially or fully then it is said to fulfil the required conformance relation. The conformance relations have several variations, however. In this section an intuitive description of the conformance relations used in the appended papers is given.

Input-output conformance relation (IOCO)

The IOCO conformance relation requires the outputs emitted by the implementation to be a subset of the outputs emitted by the specification after the execution of each possible trace in the specification. The IOCO relation only puts requirement on outputs but there is no requirement on inputs.

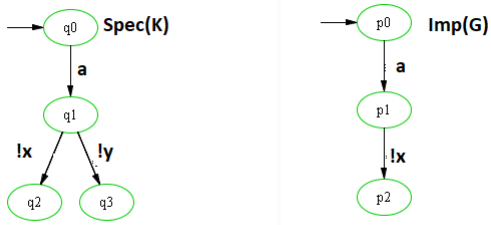

Figure 3.1: Example illustrating implementation IOCO specification

Consider the example illustrated in Fig. 3.1 that shows a specification model K and an implementation model G. The event set consists of an input, a and outputs $!x$, $!y$. Note that the outputs are denoted with an exclamation ! mark.

Now, to evaluate IOCO, first the empty trace ϵ is executed. After the execution of the empty trace, the initial states p_0 and q_0 are reached in the implementation and the specification, respectively.

From the states p_0 and q_0, when the emitted outputs by G is compared with K, it can be seen that the states p_0 and q_0 both have no outputs but rather an input a. Hence, G does emit a subset of the outputs specified by K for the executed trace; namely the empty set $\varnothing$.

For the next trace, which is the single input event a, the q_1 state reached in G is compared with the p_1 state reached in K. The comparison shows that the implementation G emits an output $!x$, while K emits the outputs $!x$ and $!y$. This means that G emits a subset of the outputs emitted by K, and thus is IOCO with respect to K for this trace, as well.

For the trace $a.!x$ there are no outputs prescribed in K and the same goes for the G, while the trace $a.!y$ is not possible in G, so a null state is reached. This means that the given implementation G is IOCO with respect to the specification for all possible traces in K.

Although the IOCO testing relation is simple and can be easily integrated in practical settings, as shown for the approach presented in Paper A, it only puts requirements on the outputs (and not the inputs). This means that faults associated with inputs cannot be uncovered using IOCO.

To remedy this the input-output conformance simulation relation (IOCOS) [11] was introduced.

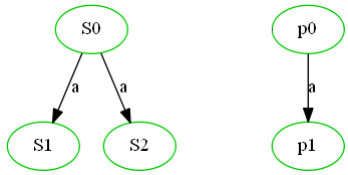

Figure 3.2: Non-deterministic system (Left), Deterministic system (Right)

Input-output conformance simulation relation (IOCOS)

The requirement posed only on outputs by the IOCO testing relation allows even empty implementations to be conformant with respect to a given specification. Thus, this property is not very useful in practical settings.

To cater also for inputs, IOCOS puts an extra requirement on inputs, while keeping the same subset requirement on outputs. Hence, the IOCOS simulation relation tests an implementation for faults associated with both inputs and outputs. A fault is registered by the IOCOS framework if the implementation rejects a mandatory input and or emits an unexpected output [34].

Theoretically, there are two possibilities if an event occurs in a system, one possibility is the occurred event enables the system to transit from a source state to a single target state. The other possibility is the occurred event enables the system to transit from the source state to multiple target states. In the former case, the system is said to be *deterministic*, while in the latter case it is *non-deterministic*.

The example given in Fig. 3.2 shows how, on the occurrence of the event a, the deterministic system transits to a single state p_1. However, in the case of a non-deterministic system two (or more) states, i.e. S_1 and S_2, are reached for the same event.

For the deterministic case, [35] defined IOCOS as an implementation is IOCOS with respect to the specification if for all possible common traces in the implementation and specification, the implementation emits a subset of the outputs and a superset of the inputs after the execution of each trace.

To understand how the IOCOS simulation relation tests an implementation, an example is illustrated in Fig. 3.3. A fault is detected in the implementation G, when the trace with input event a is executed. The fault is due to the missing input event b in state p_1 of G, which should have been there as it is prescribed by K in state q_1.

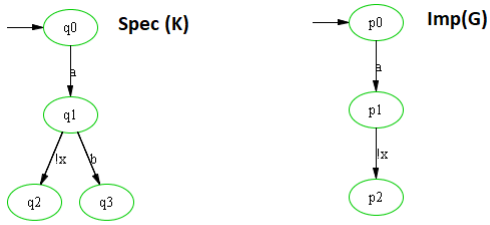

Figure 3.3: Implementation (Right) fails IOCOS with respect the specification (Left)

In practical settings, similar to IOCO, the IOCOS simulation relation may also fail to uncover faults. The reason is that some output behaviors that are prescribed by the specification may never get tested due to the subset requirement on the outputs. Another reason is that some input behaviors in the implementation that are not prescribed by the specification escapes testing due to the superset requirement on the inputs.

If testing is done to find faults related to nominal inputs and outputs in an implementation then depending on the requirement partial conformance offers by IOCOS is probably enough. However, if testing is done to find faults related to safety inputs and outputs then the partial requirements posed by IOCOS are not enough. For example IOCOS is probably not suitable to find faults in a safety PLC code as any faults in the safety code that escapes testing can have devastating consequences.

Thus, a new relation, *safe-IOCOS*, that poses a stringent requirement on safety events is proposed by [12] and will be discussed in the section to follow.

Safe input-output conformance simulation relation (safe-IOCOS)

The safe-IOCOS relation requires the implementation to emit the exact same safety events after the execution of each trace for all possible traces in the specification. However, this does not mean that the implementation have to be completely trace equivalent with respect to the specification, but rather only in terms of traces that are composed of safety events.

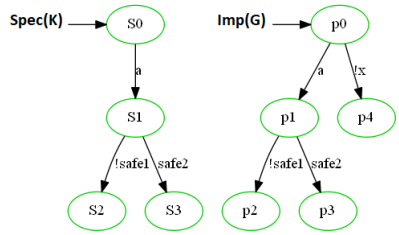

Figure 3.4: Implementation (right) simulates the specification (left) using safe-IOCOS

Compared to IOCOS that allows partial conformance for all outputs in the implementation and inputs in the specification, safe-IOCOS puts a stringent requirement on safety inputs and outputs. Apart from the traces composed of safety behaviors, the implementation is allowed to preserve its nominal behaviors.

To uncover faults using safe-IOCOS, the equality requirement is also needed for the nominal events associated with the safety events in a trace. This means that the nominal events associated with the safety events in a trace of a given implementation cannot be missing and must exist in a correlative manner with respect to the given specification.

The procedure to find faults can now be described using an example given in Fig. 3.4. Similar to IOCO and IOCOS, traces starting from the initial states are executed in both the specification and implementation. However, after reaching a state, the equality of the safety events is evaluated (but not other nominal events).

Fig. 3.4 shows an implementation G having nominal events a and *!x* in the state p_0 and two safety events *!safe1* and *safe2* in state p_1. The specification model K contains the nominal event a in state S_0 and two safety events *!safe1* and *safe2* in state S_1.

For the example in Fig. 3.4, it can be seen that after executing the trace with the nominal event a, the safety events *!safe1* and *safe2* are present in both G and K. Hence, G fulfils the safe-IOCOS relation with respect to the given specification K.

The presence of the nominal *!x* in p_0 shows that the implementation has the liberty to perform any other nominal task as long as the safety requirements posed by the specification are fulfilled.

For the work presented in this thesis, Paper B details an approach to reduce testing time for an implementation using formal approaches. The idea is to minimize test cases by omitting traces with the same safety behaviors if they have been already tested and by removing non-determinism. For this purpose, bisimulation relation and subset construction method is used.

Paper C details an approach to amend an implementation using the *infimal safety extension* that enables removal of faults from a given implementation and as a result makes the implementation safe-IOCOS.

CHAPTER 4

Testing and amending an implementation to ensure safety

In this section is given a brief introduction to problems associated with testing and correction in the industrial sector and formal methods domain.

4.1 Testing safety PLC code using a simulation model

For physical production systems, nominal PLC code testing is increasingly being supplemented by VC. However, many companies still use the *black-box testing* approach to test the PLC code [36]. This means that a test engineer manually gives various inputs to a controller, and the outputs triggered by the given inputs are observed. Based on the observed outputs, the test engineer decides whether the implemented code conforms the specified behavior or not.

Similar activities are carried out for the testing of safety PLC code. However, most of the time there is no VC involve in testing the safety aspects. Instead, it is tested on the real system in the FAT phase using various checklists along with the nominal PLC code.

Due to earlier testing of the nominal PLC code in the VC phase, the validation process for the nominal PLC code does not take a lot of time. However, this is not true for the safety code as it has not been tested before using a simulation model. Hence, the safety PLC code is required to be tested carefully, since if even minor faults in the safety code are overlooked then there could be significant repercussions at the later stage after physical commissioning.

Paper A presents an approach that can help to test and validate the safety PLC code using a simulation model. The proposed approach is supported by IOCO, which further strengthens the testing procedure by adding a formal methods dimension to it.

For the proposed approach in Paper A, the relationship between IOCO and the safety PLC code testing is established in the following manner. IOCO basically tests an implementation using the black-box approach, and the assessment is carried out on the basis of the emitted outputs. So, in Paper A the safety PLC code along with the simulation model is treated as the implementation, which was assessed on the emitted outputs with respect to the specification.

Another shortcoming of testing the safety PLC code on the real system during FAT is that no corner test cases can be implemented. This so, since it can put the production system in danger and may result in machine damage. Had there been prior testing using a simulation model, then high risk scenarios with higher risk of damaging equipment could have been tested.

However, in the virtual testing scenario, it is possible to test corner test cases without the worry to disrupt or destroy any equipment. This liberty in the presented approach in Paper A not only widens the scope of testing, but also allows engineers to get better prepared for the FAT phase. And due to the prior testing of complex test cases in the virtual environment, the time spent during the FAT phase can be reduced.

Furthermore, when safety PLC code is tested, it is tested along with the complete safety instrumented system [37] or safety related control system [38] (SRCS). This includes safety sensors, the safety PLC, and the final controlled element (valve). The reason is to test the safety hardware with all its special functions, e.g. redundancies, with the safety PLC code.

To keep the testing of the safety PLC code in the virtual environment as close to reality as possible, the SRCS concept must be correctly implemented in the simulation model. If a safety sensor of the desired behavior is not

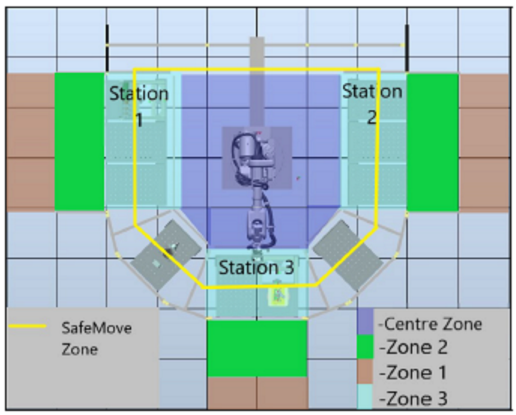

Figure 4.1: Production Cell

available in the software library, then the first choice should always be to model the sensor based on the available data of the real sensor. However, in some cases, a makeshift arrangement can do the required job, and this can be seen in Paper A, where a proximity sensor is used due to the unavailability of a component model for a floor scanner in the software library.

The approach presented in Paper A is validated with a use case that represents a real production cell that is functional at a manufacturing company. This simulation model, which is illustrated in Fig. 4.1, was created in *Process simulate* [39], and the safety code was implemented in a *Siemens-1500 PLC* [40]. However, the presented approach is not brand specific and can be implemented using any standard PLC and modeling tool of similar features. The details regarding modeling and implementation can be found in Paper A.

In terms of testing, a model of a human in the simulation tool is triggered via the human-machine interface (HMI) *WinCC* [41]. For each test case, the human disrupts the nominal operation in the production cell, which enables the corresponding safety logic to take control.

The disruption causes the safety logic outputs to change to either *high* or *low*, which is observed on the HMI screen with the help of an illuminating lamp. If the lamp does not illuminate then the executed test constitute a pass, otherwise it constitutes a fail. When the executed test fails, then the safety PLC code needs to be evaluated for faults manually.

4.2 Reducing test cases for conformance testing

When IOCOS is used to test an implementation then all reachable states in the specification and implementation must be tested per definition of IOCOS. Hence, if any reachable state-pair that is possible in both the specification and the implementation is omitted from testing, then the implementation cannot be considered IOCOS with respect to the specification.

Similarly for safe-IOCOS, which is also a simulation relation and a further refinement of IOCOS, each reachable state-pair in the implementation and the safety specification must be evaluated. And to ensure safety, the implementation is required to be trace equivalent with the safety specification only for the traces composed of safety events.

In practice, there are several safety behaviors that are common in each nominal operation. And emergency shutdown sequence is one such example of safety behavior that usually halts the whole production system if it gets triggered. Thus, if safe-IOCOS is used to test the safety PLC code, the emergency shutdown sequence will get tested for each trace in the safety specification.

Furthermore, if there is non-determinism present in the implementation then some test cases may pass sometimes, while fail at other times. This so, since the implementation can arbitrarily choose non-deterministic branch with the tester unable to influence the choice.

Due to the abovementioned problems, the time spent with testing will increase unnecessarily with no real benefits. Also, due to the presence of non-determinism, uncovering actual faults becomes difficult. Hence, to increase the time efficiency of the testing effort and actual fault detection, an approach is needed to reduce the number of test cases.

However, the test cases must be reduced in a manner such that only trivial test cases are avoided, while to ensure safety the original definition of safe-IOCOS is preserved. If the time spent on testing can be reduced, then the addition of safe-IOCOS will further strengthen the testing process in industries in combination with VC.

To address the problem of reducing test cases, an approach is proposed in Paper B. The central idea is to first maximally reduce the number of states in the given specification that is used to implement the system.

This reduction is carried out in two steps. In the first step, bisimulation abstraction [29], [42] is applied on the given specification, which merges bisimilar states. The second step is to transform the non-deterministic behavior to

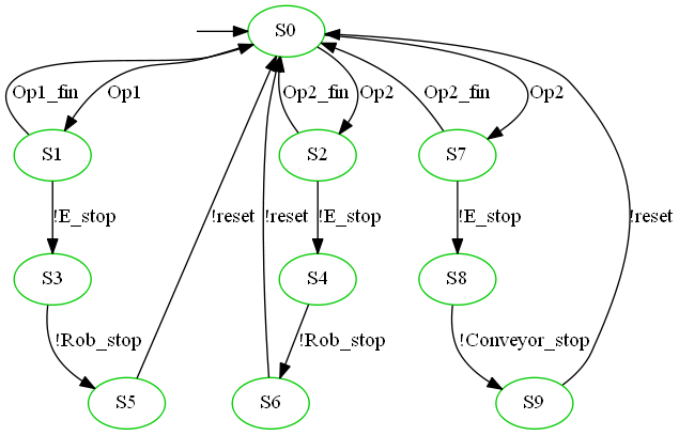

Figure 4.2: Original specification K

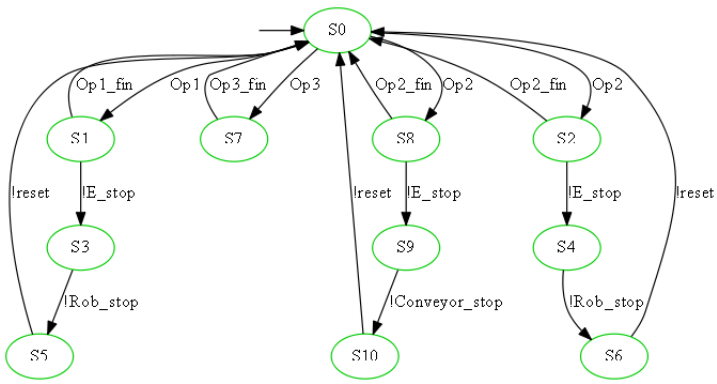

Figure 4.3: Implementation G based on K

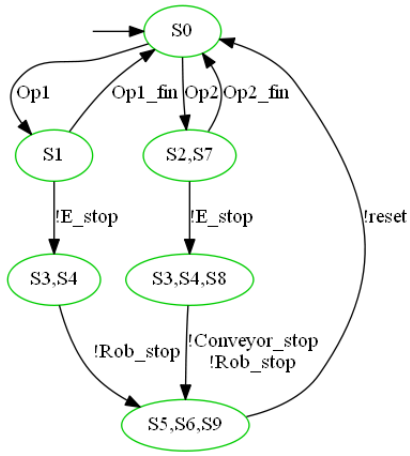

Figure 4.4: Maximally reduced specification K_{BS}

deterministic behavior by applying the subset construction method [26]. After implementing the system, from the reduced specification, the same maximally reduced specification is used to test the system.

The use of bisimulation helps in avoiding repetitive testing of certain behaviors as several traces with the same future behavior related to safety can converge to a single trace. Hence, if the common safety behaviors are tested for one trace, it does not have to be tested for other traces. Thus, all the test cases that include the same safety behavior can be excluded.

Subset abstraction on the other hand removes non-determinism from the specification. Determinism not only helps in identifying actual faults but also reduces the number of test cases. This is because, after executing a deterministic trace, the test engineer would know what behavior to expect from the implementation. However, in non-deterministic case, the same trace can give different results depending on the state reached in the implementation.

To elaborate the proposed approach in Paper B, consider an example of a specification K illustrated in Fig. 4.2. In the given specification model, it can be seen that the traces starting with the events *Op1* and *Op2* have the same safety behaviors, i.e. *!E_stop.!Rob_stop.!reset*. In addition, there is another, non-deterministic, trace with event *Op2* that has *!E_stop.!Conveyor_stop.!reset*.

Based on K, an implementation G of a system is shown in Fig. 4.3. The

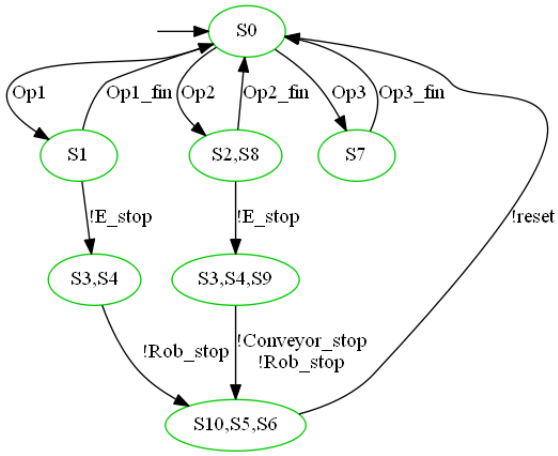

Figure 4.5: Implementation based on K_{BS}

nominal loop *Op3.Op3_fin* is irrelevant for the safety but of course important for the nominal behavior. Now, when a test engineer tests the given implementation for safe-IOCOS with respect to K, the implementation must be tested for each trace per the safe-IOCOS definition. This means that the safety events *!E_stop*, *!Rob_stop*, and *!reset* gets tested twice, first for the trace starting with the nominal event *Op1* and then for the trace staring with the nominal event *Op2*.

Furthermore, if the intention is to test the trace *!E_stop.!Rob_stop.!reset*, the execution of *Op2* in state S_0 does not guarantee that the trace *!E_stop.!Rob_stop.!reset* gets tested. This is because due to non-determinism each time *Op2* is executed, the implementation may choose the trace associated with the events *!E_stop*, *!Conveyor_stop*, and *!reset*.

According to the approach that is presented in Paper B, if we reduce the specification before implementation by applying bisimulation and subset abstraction, this can help in reducing test cases. This is so, since bisimulation will reduce number of states by converging traces with the same future safety behavior to a single state. And subset construction will remove non-determinism, which helps in raising the confidence on the testing procedure.

Now, for the given specification in Fig. 4.2, the result of applying bisimulation and subset construction can be seen in Fig. 4.4. K_{BS} is free of non-

determinism as well as repetitive behaviors that were plaguing the original (non-reduced) specification. and so is the resultant implemented system based on K_{BS} see (Fig. 4.5).

Now, if this system is tested according to K_{BS}, the test engineer will be able to confirm the results more confidently due to the absence of non-determinism. This is because whenever, the event *Op2* is executed only state (S_2, S_7) is reached. As a result, each time this event is executed, the result will be congruent with respect to K_{BS}.

Furthermore, the use of bisimulation merges multiple traces having *!Rob_stop* and *!reset* to a single trace. Thus, these behaviors do not have to be tested again and hence can be omitted from the test cases.

4.3 Automatic correction of faults to ensure safety

The goal of testing is to uncover faults in an implementation. Now, in an industrial setting, an implementation is typically controller in closed-loop setting with a physical plant, whose nominal aspects are implemented in a standard PLC, while the safety aspects are implemented in safety PLC. And to uncover the faults either a black-box testing approach [36] or VC are typically used.

In Paper A, the approach presented to test the safety PLC code helped to uncover faults. However, if there are faults associated to safety inputs then IOCO is not helpful due the absence of requirements on inputs. Hence, it is not suitable to test safety PLC code in a comprehensive manner.

IOCOS on the other hand have an extra condition on inputs in addition to the outputs. But according to definition of IOCOS, the implementation is allowed to have more inputs. Among these inputs, there is a possibility that certain input behaviors can lead to unsafe states and hence should be removed. In addition, the implementation does not have to fully comply with the specification in terms of outputs, which means that certain specified outputs can go missing in the implementation. These two conditions does not make IOCOS a good candidate for testing safety PLC code as certain inputs can be of spurious nature and certain outputs can be missing. This partial requirements on inputs and outputs are not good for testing safety properties.

Hence, for testing safety properties, safe-IOCOS is proposed by [12] and in Paper C, it is further elaborated. The safe-IOCOS relation requires the implementation to emit the exact same safety inputs as well as outputs for

all possible traces. This equality requirement is only for safety behaviors and does not apply on nominal behavior of the implementation. Thus, the implementation is allowed to have more nominal behavior, but in terms of safety it is restricted by the specification.

Now, for testing safety properties, the safe-IOCOS relation works better due to the equality requirements for safety inputs and outputs. However, when it comes to fault removal from the implementation, engineers typically down and manually amend the code. This manual procedure is on the one hand time consuming, and on the other hand prone to human errors. And in the case of safety PLC code, faults caused by manual amendment can have more dangerous consequences.

Therefore, to avoid the problems caused by manual procedures, an automatic procedure to amend the implementation based on safety specification would be useful. Especially when we are dealing with the safety PLC code. However, first we need to uncover the types of faults that can occur in the safety PLC code.

There are two possible types of faults uncovered in relation to the shortcomings of IOCOS. The first type of fault is due to spurious events, i.e. events in the implementation that are not prescribed by the specification. The second type of fault is due to missing safety events that are prescribed by the specification but are not implemented. The presence of these faults not only makes the implementation unsafe but also non-safe-IOCOS with respect to the given safety specification.

To solve this problem, Paper C proposes a procedure based on the *infimal controllable super-language* ($\mathbf{inf}\,\mathbf{C}(G||K_x)$) [16]. The $\mathbf{inf}\,\mathbf{C}(G||K_x)$ is a form of language extension typically used to generate supervisors algorithmically by expanding the language of the specification after adding uncontrollable events as mentioned in Sec. 3.

The proposed procedure in Paper C extends the language of the implementation (not the specification) for safety, and is termed the *infimal safety extension*. In this procedure, the safety events are treated as uncontrollable events, so that they are added and removed with respect to the safety specification from the implementation.

Furthermore, the proposed procedure in Paper C does not modify other segments of the implementation. This is achieved by segregating the set of states that require amendment in terms of safety from the states that should

remain untouched. To segregate the states, FSC of the implementation and the safety specification is carried out. However, in FSC, only events that are prescribed by the safety specification are allowed to participate, while the other events are disabled.

Now, after the identification of relevant states, the first step is to add a new state, called $DEAD$, in the implementation model and add self-loop transitions of all safety events to it. After adding the self-loops to the $DEAD$ state, undefined safety transitions are added from each relevant state to the $DEAD$ state. This gives birth to new augmented implementation.

During the development of the augmented implementation, all the states that are not faulty in the implementation remained intact. In the next step, we carry out the FSC again of the augmented implementation with the safety specification. This enables removal of spurious events from each relevant state in the implementation, and also adds the missing safety events to the corresponding states.

A problem that may occur in some cases is that the resultant implementation becomes blocking. This problem is detailed in Paper C and is solved by merging states with the same state labels. After the merger of states, the generated non-blocking model is controllable as well as safe-IOCOS with respect to the given safety specification.

CHAPTER 5

Summary of included papers

5.1 Paper A

Adnan Khan, Petter Falkman, Martin Fabian
Testing and Validation of Safety Logic in the Virtual Environment
CIRP Journal of Manufacturing Science and Technology,
Volume 26, August 2019, Pages 1-9
 DOI: 10.1016/j.cirpj.2019.07.002 .

In this paper, an approach to test safety PLC code using a simulation model is presented. The approach is substantiated with the IOCO testing relation. The industrial sector is increasingly using virtual commissioning to reduce the physical commissioning time by testing the PLC code for nominal behavior before physical commissioning. However, the safety PLC code is typically tested on the real physical system with the help of checklists. This practice of testing safety PLC code is impeding the industry to take full advantage of virtual commissioning. Using the proposed approach, better preparation can be done for the factory acceptance phase, which can help in further reduction of the physical commissioning time.

5.2 Paper B

Adnan Khan, Sahar Mohajerani, Martin Fabian
On test case reduction for testing safety properties of manufacturing systems
Submitted to Journal of Manufacturing Systems. March 2021, SMEJMS-S-21-00206 .

This paper presents an approach to reduce testing time by abstracting test cases for the safe-IOCOS relation in order to reduce testing time. safe-IOCOS is a simulation relation that specifically tests an implementation's safety properties according to a specification to uncover faults related to safety. The validity of safe-IOCOS is confirmed if the implementation is trace equivalent to the specification for traces containing safety behaviors. However, in real production systems, certain safety behaviors are part of many nominal operations and hence get tested more than once. And according to the nature of safe-IOCOS, the implementation is tested for all possible state combinations in the specification and implementation, which makes the testing time consuming. Furthermore, the presence of non-determinism makes testing difficult as well as time consuming, since the safe-IOCOS requirements for equality of safety behaviors cannot be convincingly confirmed in the presence of non-determinism and consequently can lead to erroneous fault detection. As a result, the test engineer ends up spending more time to validate the implementation. The root of the problems highlighted in this paper is related to the construction of the specification used for implementation and testing. It is shown that if the number of states in the given specification are reduced before implementing the manufacturing system, then this can help in reducing test cases and hence testing time. To address these problems, the paper studies two formal approaches, i.e. *bisimulation* and *subset construction*. The bisimulation relation merges two states that have same future behavior. This property of bisimulation enables reduction of test cases associated with repetitive safety behaviors, as traces with same future safety behavior need not be tested more than once. The subset construction method on the other hand removes non-determinism from the LTS and hence can be employed to convert a non-deterministic LTS to a deterministic one. The paper proves a theorem that shows that the proposed reduction of a specification preserves the safe-IOCOS relation.

5.3 Paper C

Adnan Khan, Martin Fabian
On testing and automatic mending of safety PLC code
CIRP Journal of Manufacturing Science and Technology,
Invited for resubmission after revision. March 2021,
CIRPJ-D-21-00010 .

This paper presents an approach for the amendment of an implementation with respect to a safety specification to ensure safety. Manufacturing companies increasingly rely on virtual commissioning and formal methods for testing nominal PLC code, as well as safety PLC code. This paper discusses a new conformance relation called safe-IOCOS for testing an implementation for safety properties and a procedure to automatically compute the *infimial safety extension* for automatic amendment of a faulty implementation to ensure safety. Conformance testing approaches, e.g. IOCO and IOCOS, have been around and are used for testing safety PLC code. However, due to the partial conformance requirement on inputs and outputs, they are not suitable for testing safety properties. Also, for amending an implementation to remove faults the procedure is usually manual in nature. This means that engineers have to put in immense amounts of manual effort and time to find the faulty segments in the PLC code to remove the faults. Furthermore, during amendment, engineers have to be really careful as they must not disturb other segments of the implementation that are fault free. For safety PLC code this is even more important as any error made while carrying out amendments can have devastating results. To solve this issue, a procedure to automatically amend the faulty implementation to ensure safety is presented. This procedure computes the so-called *infimial safety extension*, which in addition to fault removal also makes the implementation safe-IOCOS.

CHAPTER 6

Concluding Remarks and Future Work

The research work presented in this thesis deals with testing and amending an implementation such that safety properties are ensured.

Paper A presents an approach to test the safety PLC code using a simulation model to assist the FAT (Factory Acceptance Testing) phase. The main purpose behind this approach is to exploit VC (Virtual Commissioning) to its full potential by incorporating safety PLC code testing in a virtual environment. Typically, VC is used to test and validate PLC code associated with nominal behavior, and the safety PLC code is usually tested on the real system along with other safety hardware functions. The approach presented in Paper A can help in cutting down time spent during the FAT phase by allowing prior testing of safety PLC code. This reduction in time during the FAT phase enables overall reduction in physical commissioning time, which is the main goal of VC.

Furthermore, the approach proposed in Paper A is strengthened by the addition of the IOCO (Input/Output Conformance) testing relation to supplement the visual validation procedure based on VC. The IOCO testing relation tests an implementation with respect to the emitted outputs. So its relationship with the safety PLC code only concerns the outputs of the safety

PLC.

In Paper B, an approach based on bisimulation and the subset construction method is presented to reduce test cases for the safe-IOCOS simulation relation. The safe-IOCOS relation is established by achieving equality for the prescribed safety events by adding missing safety events and removing spurious safety events with respect to the specification. However, practically there are several safety behaviors that are implemented for each nominal behavior. These safety behaviors get tested multiple times. As a result, the time spent by the engineers on testing is unnecessarily increased. Furthermore, the presence of non-determinism makes testing difficult, because executing an event does not guarantee transition to a specific state. Thus, the test engineer ends up spending more time to either confirm faults or the validity of safe-IOCOS.

To reduce the testing time for safe-IOCOS, the approach in Paper B enables abstraction of test cases from a given specification. The idea is to merge states having the same safety behavior to a single state using bisimulation and then remove non-determinism using subset construction. This results in efficient fault detection due to determinism as well as omission of certain traces that has the same future behavior. This so, since if one trace that reaches the merged state is tested and fulfills safe-IOCOS then the remaining traces having common safety behaviors need not be tested.

Paper C presents an approach to automatically amend a faulty implementation to ensure safety. The approach takes advantage of the *infimal controllable superlanguage* [16]. The main goal is to amend a given implementation in such a manner that faults related to safety are removed. The amendment not only makes the implementation safe by removing faults, but also makes it safe-IOCOS with respect to the given specification.

A future work idea is to enrich the safe-IOCOS relation to capture timed event safety sequences. This would enable the safe-IOCOS relation to test safety PLC code with timers that can later be incorporated in the virtual commissioning domain to further enhance testing using safe-IOCOS.

Furthermore, the approach presented in Paper C can be extended to address the problem of scalability; a *compositional* approach similar to the one presented in [43] can be explored. This approach would enable several implementations and safety specifications related to different parts of a global system to interact with each other via synchronous composition, which can be advantageous when amending complex industrial systems.

References

[1] J. Ždánsky and K. Rástočný, "Influence of safety PLC parameters to response time of safety functions," in *2013 International Conference on Applied Electronics*, 2013, pp. 1–4.

[2] C. G. Lee and S. C. Park, "Survey on the virtual commissioning of manufacturing systems," *Journal of Computational Design and Engineering*, vol. 1, no. 3, pp. 213–222, 2014.

[3] L. Fransén, *National Electric Vehicle Sweden, Manufacturing Engineer, Private conversation*, 2016.

[4] G. Frey and L. Litz, "Formal methods in PLC programming," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, IEEE, vol. 4, 2000, pp. 2431–2436.

[5] Z. Liu, C. Diedrich, and N. Suchold, *Virtual Commissioning of Automated Systems*. INTECH Open Access Publisher, 2012.

[6] Areej, "Difference between FAT and SAT," *https://automationforum.co/difference-between-fat-and-sat/*, 2020.

[7] D. Horsley, *Process plant commissioning: a user guide*. IChemE, 1998.

[8] M. Krichen and S. Tripakis, "Black-box conformance testing for real-time systems," in *International SPIN Workshop on Model Checking of Software*, Springer, 2004, pp. 109–126.

[9] M. Utting and B. Legeard, *Practical Model-Based Testing: A Tools Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[10] J. Tretmans, "Model based testing with labelled transition systems," in *Formal methods and testing*, Springer, 2008, pp. 1–38.

[11] C. Gregorio-Rodríguez, L. Llana, and R. Martínez-Torres, "Input-output conformance simulation (IOCOS) for model based testing," in *Formal Techniques for Distributed Systems*, Springer, 2013, pp. 114–129.

[12] A. Khan and M. Fabian, "On the safe IOCOS relation for testing safety PLC code," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1449–1452.

[13] GKN and NEVS, "Virtual preparation and commissioning for everyone," *https://www.virtcom.se/*, 2016.

[14] G. Tretmans, "Test generation with inputs, outputs and repetitive quiescence, 1996," *URL http://doc. utwente. nl/65463*, vol. 46, 1996.

[15] R. Malik, K. Åkesson, H. Flordal, and M. Fabian, "Supremica-–an efficient tool for large-scale discrete event systems," in *IFAC World Congress, Tolouse, France*, 2017.

[16] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, ser. SpringerLink Engineering. Springer US, 2009, ISBN: 9780387333328.

[17] L. Ricker, S. Lafortune, and S. Genc, "DESUMA: A tool integrating GIDDES and UMDES," in *2006 8th International Workshop on Discrete Event Systems*, 2006, pp. 392–393.

[18] ESA, "Advantages of SoftPLC vs traditional PLC," *https://www.esa-automation.com/en/advantages-of-softplc-vs-traditional-plc/*, 2020.

[19] J. Swathanandan and G. Kristoffer, "Virtual commissioning for safety verification of a collaborative robotic cell," M.S. thesis, 2018.

[20] M. Oppelt and L. Urbas, "Integrated virtual commissioning an essential activity in the automation engineering process: From virtual commissioning to simulation supported engineering," in *IECON 2014-40th Annual Conference of the IEEE Industrial Electronics Society*, IEEE, 2014, pp. 2564–2570.

[21] M. Dahl, K. Bengtsson, P. Bergagård, M. Fabian, and P. Falkman, "Integrated virtual preparation and commissioning: Supporting formal methods during automation systems development," *IFAC-PapersOnLine*, vol. 49, no. 12, pp. 1939–1944, 2016.

[22] O. Mathias, W. Gerrit, D. Oliver, L. Benjamin, S. Markus, and U. Leon, "Automatic model generation for virtual commissioning based on plant engineering data," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 635–11 640, 2014.

[23] A. Khan, P. Falkman, and M. Fabian, "Virtual engineering framework for automatic generation of control logic including safety," in *Automation Science and Engineering (CASE), 2017 13th IEEE Conference*, IEEE, 2017, pp. 648–653.

[24] S. Süß, A. Strahilov, and C. Diedrich, "Behaviour simulation for virtual commissioning using co-simulation," in *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, IEEE, 2015, pp. 1–8.

[25] FMI, "Functional mock-up interface," *https://fmi-standard.org//*, 2020.

[26] J. E. Hopcroft and J. D. Ullman, "Introduction to automata theory, languages and computation. adison-wesley," *Reading, Mass*, 1979.

[27] A. Hellgren, M. Fabian, and B. Lennartson, "Prioritised synchronous composition of inhibitor arc Petri nets," in *Discrete Event Systems*, Springer, 2000, pp. 459–466.

[28] M. Utting and B. Legeard, *Practical model-based testing: a tools approach*. Elsevier, 2010.

[29] R. Milner, *Communication and concurrency*. Prentice hall Englewood Cliffs, 1989, vol. 84.

[30] O. Ljungkrantz, K. Akesson, C. Yuan, and M. Fabian, "Towards industrial formal specification of programmable safety systems," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 6, pp. 1567–1574, 2012.

[31] J. Provost, J.-M. Roussel, and J.-M. Faure, "Translating Grafcet specifications into Mealy machines for conformance test purposes," *Control Engineering Practice*, vol. 19, no. 9, pp. 947–957, 2011.

[32] D. Soliman and G. Frey, "Verification and validation of safety applications based on PLC open safety function blocks using timed automata in UPPAAL," *IFAC Proceedings Volumes*, vol. 42, no. 5, pp. 34–39, 2009.

[33] PLCopen, "What is PLCopen," *https://plcopen.org/*, 2020.

[34] C. Gregorio-Rodríguez, L. Llana, and R. Martínez-Torres, "Effectiveness for input output conformance simulation IOCOS," in *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*, Springer, 2014, pp. 100–116.

[35] A. Khan, D. Thönnessen, and M. Fabian, "On-the-fly conformance testing of safety PLC code using QuickCheck," in *2019 IEEE 17th Transactions on Industrial Informatics (INDIN'19)*, "in press", IEEE, 2019.

[36] B. Fernández, E. Blanco, and A. Merezhin, "Testing & verification of PLC code for process control," in *Proc. of the 14th Int. Conf. on Accelerator & Large Experimental Physics Control Systems*, 2013, pp. 1258–1261.

[37] Laihua Fang, Zongzhi Wu, Lijun Wei, and Ji Liu, "Design and development of safety instrumented system," in *2008 IEEE International Conference on Automation and Logistics*, 2008, pp. 2685–2690.

[38] K. Rástočný and J. Ždánsky, "Specificities of safety PLC based implementation of the safety functions," in *2012 International Conference on Applied Electronics*, 2012, pp. 229–232.

[39] Siemens, "Process simulate," *https://www.plm.automation.siemens.com*, 2018.

[40] ——, "Simatic-s7-1500 PLC," *https://www.siemens.com/global/en/home /products /automation/systems/industrial/plc/simatic-s7-1500.html*, 2018.

[41] ——, "Wincc," *https://w3.siemens.com/mcms/ human-machine- interface/en /visualization-software/scada/simatic-wincc/pages/default.aspx*, 2018.

[42] J.-C. Fernandez, "An implementation of an efficient algorithm for bisimulation equivalence," *Science of Computer Programming*, vol. 13, no. 2-3, pp. 219–236, 1990.

[43] S. Mohajerani, R. Malik, and M. Fabian, "Compositional synthesis of supervisors in the form of state machines and state maps," *Automatica*, vol. 76, pp. 277–281, Feb. 2017.

Part II

Papers

PAPER A

Testing and Validation of Safety Logic in the Virtual Environment

Adnan Khan, Petter Falkman, Martin Fabian

CIRP Journal of Manufacturing Science and Technology,
Volume 26, August 2019, Pages 1-9
 DOI: 10.1016/j.cirpj.2019.07.002

The layout has been revised.

PAPER B

On test case reduction for testing safety properties of manufacturing systems

Adnan Khan, Sahar Mohajerani, Martin Fabian

Submitted to Journal of Manufacturing Systems. March 2021, SMEJMS-S-21-00206

The layout has been revised.

PAPER C

On testing and automatic mending of safety PLC code

Adnan Khan, Martin Fabian

CIRP Journal of Manufacturing Science and Technology,
Invited for resubmission after revision. March 2021,
CIRPJ-D-21-00010

The layout has been revised.