On the Safe IOCOS relation for Testing Safety PLC Code

Citation for the original published paper (version of record):
Khan, A., Fabian, M. (2019)
On the Safe IOCOS relation for Testing Safety PLC Code
IEEE International Conference on Emerging Technologies and Factory Automation, ETFA,

On the Safe IOCOS relation for Testing Safety PLC Code

Adnan Khan*, Martin Fabian*
* Department of Electrical Engineering
Chalmers University of Technology, Göteborg, Sweden
Email: {adnan.khan, fabian}@chalmers.se

***Abstract*—In this paper, limitations of the IOCOS testing relation in regard to testing safety PLC code is examined and a modification of the current IOCOS relation, called *safe-IOCOS* is proposed. In the IOCOS testing relation, an implementation is IOCOS with respect to a specification, if it emits a subset of the specified outputs and a super-set of the specified inputs after the execution of each trace in the specification. However, for testing safety PLC code, the IOCOS relation is not detailed enough as the subset requirement on the respective inputs and outputs could allow some safety behaviors to go untested. These limitations of the IOCOS relation may thus pose threats to humans. So the notion of *safe-IOCOS* is defined, which strengthens IOCOS to require equality between the implementation and the specification in relation to the inputs and outputs, respectively. An example shows these shortcomings of IOCOS and how the proposed safe-IOCOS relation is better suited for testing safety PLC code.**

***Index Terms*—Safety, PLC , Input-output conformance**

I. Introduction

In the manufacturing and production arena, different machines and robots are controlled in such a manner that they interact with each other to produce the desired goods. Many of the behaviours exhibited by these machines are event based, thus of a discrete nature. Typically, sensor events are regarded as output events and the actuator events are regarded as input events. The internal events related to internal transitions between states are not visible and remains unaffected from the outside.

Due to this event based behavior, manufacturing systems behaviour can be modelled as *discrete event systems* [1]. Discrete event systems evolve with respect to occurring events, while occupying a specific state at each time instant, where certain conditions are valid. Many formal approaches and methods can be used to specify, implement and verify the behaviour of such systems. One of such approach is *model-based testing* [2].

Model-based-testing [2] is a formal approach to uncover errors that subjects a model of an implementation to various tests based on a specification. A variant of this approach is called *input-output conformance* testing (IOCO) [3]. In the IOCO approach, the model of an implementation is subjected to scrutiny based on a specification. The status of the executed test is assessed based on the emitted outputs by the implementation after the execution of each trace possible in the specification. When the outputs emitted by the implementation is a subset of the specified outputs, the implementation is said to be IOCO with respect to the specification, else it is non-IOCO and requires amendment.

The IOCO has a requirement on the emitted outputs but not on the inputs, this allows the implementation with few or no specified inputs to be IOCO. To counter this, the *input-output conformance simulation* (IOCOS) [4] is introduced, which puts a requirement on the inputs, in addition to the requirement posed on the outputs by the IOCO relation. The implementation is considered IOCOS, if it emits a subset of the specified outputs and a super-set of inputs after the execution of each trace possible in the specification. If the implementation fails to satisfy either the input requirement or the output requirement, it is non-IOCOS and needs to be amended.

In industrial settings, programmable logic controllers (PLCs) are used to control the installed machines. The PLCs are of two types. One type is the standard PLCs that control the nominal behavior of a production system, e.g. welding. The second type are *safety PLCs*, which control the machines in safety critical scenarios to prevent human accidents and machine damage.

To uncover errors in the PLC code related to the nominal behaviour, *virtual commissioning* [5]–[10] is typically used, which has been integrated in the engineering tool chain by many companies. For virtual commissioning, the PLC code developed for the nominal behaviour is tested on a simulation model of a physical system. If the simulation model exhibits behaviour contrary to the specification, manual inspection of the PLC code is carried out. Upon inspection, if the PLC code is found to be erroneous then it is manually amended.

Both the IOCO and IOCOS testing relation have been used to test safety PLC logic [11] and legacy system's simulation model [12]. However, the IOCOS relation has a subset requirement on both inputs and outputs, which may allow some safety behaviour in the implementation to be overlooked; due to this, human accidents may happen.

A. Contribution

In this paper, a new relation called the safe-IOCOS (see Def. 8) is proposed, after examining limitations of the IOCOS relation. The safe-IOCOS relation allows all the inputs and outputs in the specification to be tested in the implementation due to an equality relation, which makes it more suitable for testing safety PLC code. Finally, the viability of the proposed safe-IOCOS relation is shown using an example.

B. Outline

This paper is structured in the following way. In Section II, the IOCO and IOCOS testing relations are reminded. In Section IV, an overview of some limitations in the existing IOCOS relation for testing safety PLC code along with the proposed modification is detailed with an example. Section V concludes the paper with future work direction.

II. Input-Output Conformance Relation

The input-output conformance testing relation [3] assesses a model of an implementation based on a specification. This assessment of the IOCO relation is carried out by executing all possible traces of the specification on the implementation.

To give the formal definition of IOCO, consider two disjoint sets of input actions I and output actions O. The output actions are the actions initiated by the system under test and are expressed with an exclamation mark, such as $!a \in O$. The input actions are commands to the system and are expressed with a question mark such as $a? \in I$. Now, we consider a labelled transition system in this section to elaborate the concept of IOCO and give the formal definition.

Definition 1: An *I/O labelled transition system* (LTS) is a 4-tuple $\langle S, s_0, L, \rightarrow\rangle$ where:

- S is a non-empty set of states;
- $s_0 \in S$ is the initial state;
- L is a countable set of labels. These represent observable actions of a system i.e. $L = I \cup O$ where I and O are as above. Consider also a *quiescence* symbol $\delta \notin L$, and define the sets $L_\delta = L \cup \{\delta\}$ and $O_\delta = O \cup \{\delta\}$;
- $\rightarrow \subseteq S \times L_\delta \times S$ is a transition relation such that, $p \xrightarrow{a} q$ implies $\langle p, a, q\rangle \in \rightarrow$ and $p \xrightarrow{a}$ for $a \in L_\delta$, if there exists $q \in S$ such that $p \xrightarrow{a} q$. Similarly, $p \overset{a}{\not\rightarrow}$, for $a \in L_\delta$, if there exist no q such that $p \xrightarrow{a} q$. In addition, only coherent quiescent systems are allowed, so $\rightarrow$ should also satisfy the following:
  - if $p \xrightarrow{\delta} p'$, then $p = p'$ i.e. a quiescent transition is always reflexive.
  - if $p \overset{!o}{\not\rightarrow}$ for all $!o \in O$, then $p \xrightarrow{\delta} p$, i.e. a state with no outputs is quiescent.
  - if $p \xrightarrow{!o}$ for some $!o \in O$, then $p \overset{\delta}{\not\rightarrow}$, i.e. a state with some output is not quiescent.

Furthermore, a trace t is a finite sequence of symbols of L_δ i.e. $t \in L_\delta^*$, including the empty trace ϵ.

When the transition relation is restricted to be a function, and thus for $p \xrightarrow{a} q$ and $p \xrightarrow{a} q'$ it holds that $q = q'$, the resulting LTS is said to be *deterministic*.

Additional definitions needed to express the IOCO relation in Definition 6 are as follows.

Definition 2: The set of traces from a state p in an LTS is

$$\textbf{traces}(p) = \{t \in L_\delta^* \,|\, p \xrightarrow{t}\}. \tag{1}$$

For an LTS $A = \langle S, s_0, L, \rightarrow\rangle$, its set of traces are the ones defined from its initial state

$$\textbf{traces}(A) = \textbf{traces}(s_0). \tag{2}$$

Definition 3: The set of states reached *after* a trace t from a state p is

$$\textbf{after}(p, t) = \{p' \in S \,|\, p \xrightarrow{t} p'\}. \tag{3}$$

For an LTS $A = \langle S, s_0, L, \rightarrow\rangle$, the set of states reached *after* a trace t is

$$\textbf{after}(A, t) = \{p' \in S \,|\, s_0 \xrightarrow{t} p'\}. \tag{4}$$

For a deterministic LTS, $\textbf{after}(\cdot,\cdot)$ always returns a singleton set. Then we write $\textbf{after}(p, t) = p'$.

Definition 4: The set of *outputs* from a state p is

$$\textbf{outs}(p) = \{!x \in O_\delta \,|\, p \xrightarrow{!x}\}. \tag{5}$$

Definition 5: The set of *inputs* for a state p is

$$\textbf{ins}(p) = \{x? \in I \,|\, p \xrightarrow{x?}\}. \tag{6}$$

The formal definition of the IOCO testing relation [4] can now be stated.

Definition 6: For two deterministic LTSs A and B with equal sets of labels, A is said to be IOCO with respect to B if

$$\forall t \in \textbf{traces}(B) : \textbf{outs}(\textbf{after}(A, t)) \subseteq \textbf{outs}(\textbf{after}(B, t)) \tag{7}$$

The formal IOCO definition (Def. 6) is interpreted as an implementation A conforms to a specification B, if for all the traces in the specification the outputs possible from the state reached by the implementation after a trace form a subset of the possible output events from the state reached by the specification after the same trace. Whenever this subset relation between the respective sets of output events exist, the implementation is said to be IOCO with respect to the specification, for that particular trace. If the implementation is IOCO with respect to the specification for all the traces defined by the specification, then the implementation is said to be IOCO with respect to the whole specification.

This paper uses the modified definition of IOCO given by [4], which relaxes the original assumption of the implementation being *input enabled* [3]. In addition, the original version of IOCO considers *suspension traces*, which are the traces containing quiescent behavior (states without any output). But the modified definition takes all the traces into account, since the quiescent behavior is included in the modified definition by introducing a special symbol for it.

The formal definition of IOCO has a requirement on the emitted outputs but not on the inputs, this allows the implementation with few or no specified inputs to be IOCO. To deal with this issue, [4] proposes a modified relation called the *input-output conformance simulation* (IOCOS). The IOCOS relation puts, in addition to IOCO, a requirement on the implementation that it must conform to at least one of the specified input behaviours, as formally defined in Def. 7. In addition to the IOCO requirements on the implementation having a subset of the specified outputs, IOCOS requires the implementation also to have a super-set of the specified inputs.

The formal definition of the IOCOS simulation relation can now be given.

Definition 7: For two deterministic LTSs A and B with equal sets of labels, A is said to be IOCOS with respect to B if in addition to (7), it also holds that

$$\forall t \in \textbf{traces}(B) : \textbf{ins}(\textbf{after}(B, t)) \subseteq \textbf{ins}(\textbf{after}(A, t)) \tag{8}$$

The formal IOCOS definition (Def. 6 + Def. 7) states that the implementation A conforms to a specification B, if for all the traces in the specification the inputs possible from the state reached by the implementation after a trace form a super-set of the possible input events from the state reached by the specification after the same trace. If this super-set relation between the respective sets of inputs, and the subset relation between the outputs expressed in Def. 6 exist, the implementation is IOCOS with respect to the specification for the executed trace. If the implementation is IOCOS with respect to the specification for all the traces defined by the specification, then the implementation is said to be IOCOS with respect to the whole specification.

III. Example

To highlight shortcomings of the IOCOS testing relation for safety systems, a true industrial example, which inspired the work presented in this paper is given. This concerns a machining cell, consisting of a robot and a laser, which can be accessed by the operator via the cell doors. The cell also has multiple emergency shutdown buttons located at different points to activate emergency shutdown. There are two possible safety scenarios, one related to the input event $Emergency\_shutdown$, and one related to the $Operator\_Door$ input event. In both cases, both the robot and the laser should shut down in any order, i.e. either the robot shuts down first or the laser.

Two alternative implementations, *G1* and *G2*, are tested individually with respect to the specification K. Fig. 1 illustrates the models of the implementations *G1*, *G2*, and the specification K.

First, the validity of the IOCOS relation is checked for the implementation *G1* with respect to the specification *K*. In *G1*, when the emergency shutdown button is pushed, only the laser shuts down. However, according to the specification *K*, when the emergency shutdown button is pushed, both the laser and robot should have shut down. In practice, the missing output $!robot\_shutdown$ in *G1* would go untested using the IOCOS testing relation, which could lead to human accident due to being hit by the moving robot.

Now, the implementation *G2* is tested with respect to the specification *K*. When, the emergency shutdown button is pushed, both the robot and the laser are shut down per the specification *K*. However, when the operator opens the door, only the robot stops while the laser remains operational as the output $!Laser\_shutdown$ is absent in *G2*, see Fig. 1. Thus, the operator door opening scenario has gone untested due to the missing input $Operator\_Door$ in K and could damage either the operator's skin or eyes.

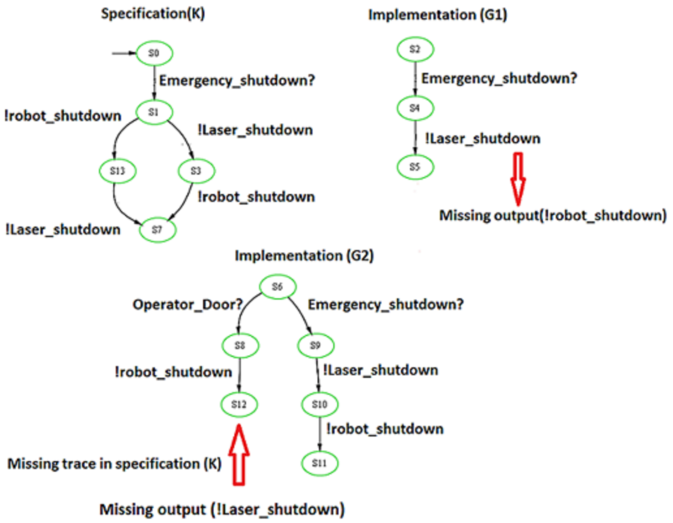

Fig. 1. Example of IOCOS shortcomings.

The IOCOS testing relation is valid for both *G1* and *G2* as it fulfills the IOCOS definition (Def. 6 + Def. 7). However, in practice both these implementations are unsafe and can lead to human accidents due to missing input $Operator\_Door$ in the specification K and the missing output $!robot\_shutdown$ in the implementation *G1*. Hence, the IOCOS relation in its current form is not rich enough for testing safety PLC logic.

IV. safe-IOCOS for Testing Safety PLC Code

From the IOCOS perspective, the control loop of the safety PLC S and the physical plant G conceptually represents an implementation $G||S$, which is tested with respect to a specification K. The implementation $G||S$ is a *synchronous composition* [1], which models the closed-loop interaction between the uncontrolled plant G and the safety PLC S, where an event can occur only if it is simultaneously enabled in both.

The example presented in Section III highlighted two shortcomings that may hinder the IOCOS relation to uncover errors in the safety PLC code. The first shortcoming is due to the subset requirement on the outputs. The second shortcoming is due to the superset requirement on the inputs. Due to this, some unspecified safety behaviours in the implementation could go untested.

To counter these shortcomings in the IOCOS relation, two modifications are proposed to the existing definition of the IOCOS. The first modification is related to the outputs emitted by the implementation that after an executed trace must be exactly the same as the outputs specified by the specification. The second modification is related to the inputs possible in the specification from the state reached that after the executed trace must be exactly the same as the inputs in the state reached in the implementation. These requirements on the inputs and outputs requires the traces of the specification to be exactly the same as the traces of the implementation.

Based on the equality requirement, the formal definition for the *safe-IOCOS* relation can now be given.

Definition 8:

For two deterministic LTSs $G||S$ and K with equal sets of labels, $G||S$ is said to be safe-IOCOS with respect to K if.

$$\begin{aligned} &\forall t \in \textbf{traces}(K): \\ &\quad \textbf{outs}(\textbf{after}(G||S,t) = \textbf{outs}(\textbf{after}(K,t)) \wedge \\ &\quad \textbf{ins}(\textbf{after}(G||S,t) = \textbf{ins}(\textbf{after}(K,t)) \end{aligned} \tag{9}$$

The safe-IOCOS definition (Def. 8) is interpreted as the implementation $G||S$ conforms to a specification K, if for all the traces in the specification, the inputs and the outputs possible from the state reached by the implementation after a trace are equal to the possible input and output events from the state reached by the specification after the same trace. If this equality relation between the respective sets of inputs and the outputs exist, the implementation is safe-IOCOS with respect to the specification for the executed trace. If the implementation is safe-IOCOS with respect to the specification for all the traces defined by the specification, then the implementation is said to be safe-IOCOS with respect to the whole specification.

These modifications in the IOCOS relation requires the language of the implementation $G||S$ and the specification K to be exactly equal i.e. $L(G||S) = L(K)$.

A. Applying safe-IOCOS for Safety PLC Testing

Now, we apply the proposed safe-IOCOS relation to test the implementations *G1* and *G2* with respect to the specification *K* to uncover errors that were missed during testing by the IOCOS testing relation.

First, for *G1*, when the proposed safe-IOCOS is used, the error related to the missing output $!robot\_shutdown$ is uncovered due to the equality relation on the outputs. Because the implementation *G1* would have to emit the exact outputs i.e. $!Laser\_shutdown$, and $!robot\_shutdown$, after the input event $Emergency\_shutdown$ per definition (Def. 8). Hence, according to the proposed safe-IOCOS relation the implementation is non-safe-IOCOS and requires amendment.

Similarly, for the implementation *G2*, the proposed safe-IOCOS relation would be able to uncover the missing input event $Operator\_Door$ in the specification *K* due to the equality requirement on the inputs and also the missing output associated with it i.e. $!Laser\_shutdown$ in *G2* per definition (Def. 8). Hence, making the proposed safe-IOCOS relation more appropriate for safety PLC code testing.

V. Conclusion

In this paper, limitations of the input-output conformance simulation relation, IOCOS, for testing safety PLC code is highlighted and a modification called the safe-IOCOS is proposed. The implementation is safe-IOCOS with respect to the specification, if for all possible traces in the specification, the inputs and outputs possible from the state reached in the implementation after the executed trace are equal to the possible inputs and outputs from the state reached in the specification. To show the viability of the proposed safe-IOCOS relation, an example is presented, which shows how the equality relation rather than subset is better suited to test and validate the safety PLC code. The safe-IOCOS relation allows the absent inputs and outputs in the implementation to be uncovered based on the specification, which could prevent human accidents.

This is work in progress and in the future the proposed safe-IOCOS relation will be elaborated and corroborated with relevant proofs. Furthermore, a comparison between the safe-IOCOS relation and other formal approaches e.g. bi-simulation, testing based on formal verification and etc. will be made. Also, the safe-IOCOS relation will be examined from the perspective of supervisory control theory [13] based on the notion of controllability.

Acknowledgements

This work has been carried out at the Wingquist Laboratory VINN Excellence Centre within the Chalmers Production Area of Advance. It has been supported by VR SyTeC (ref 2016-06204).

References

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

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

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

[4] 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.

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

[6] P. Hoffmann, R. Schumann, T. M. Maksoud, and G. C. Premier, "Virtual commissioning of manufacturing systems a review and new approaches for simplification." in *24th European Conference on Modelling and Simulation (ECMS 2010)*, 2010, pp. 175–181.

[7] A. Jain, D. Vera, and R. Harrison, "Virtual commissioning of modular automation systems," *IFAC Proceedings Volumes*, vol. 43, no. 4, pp. 72–77, 2010.

[8] 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.

[9] S. Süß, S. Magnus, M. Thron, H. Zipper, U. Odefey, V. Fäßler, A. Strahilov, A. Kłodowski, T. Bär, and C. Diedrich, "Test methodology for virtual commissioning based on behaviour simulation of production systems," in *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*. IEEE, 2016, pp. 1–9.

[10] 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.

[11] 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)*. IEEE, 2019, "in press".

[12] A. Khan, P. Falkman, and M. Fabian, "Digital twin for legacy systems: Simulation model testing and validation," in *Automation Science and Engineering (CASE), 2018 14th IEEE Conference*. IEEE, 2018.

[13] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.