IEEJ Journal of Industry Applications
Vol.2 No.6 pp.306–314 DOI: 10.1541/ieejjia.2.306

Paper

Formal Verification of Logic Control Systems with Nondeterministic Behaviors

Saifulza Alwi[*,**] Non-member, Yasutaka Fujimoto[*] Member

(Manuscript received Nov. 14, 2012, revised June 2, 2013)

This paper describes a formal modeling and verification of an arm pick-and-place system, in which nondeterministic behaviors of the arm state condition and timer function blocks are applied. We design an appropriate PLC program using a ladder diagram (LD) for the arm pick-and-place operation and apply in it a situation where the arm may drop the product or material being gripped because of an external force. In addition, the timer function blocks are used with formalization of their finite-state logical properties. We use an actual model of the arm to verify that safe operations are established for normal product pick-and-place, as well as when the product has fallen. In addition, we perform arm model verifications for five important temporal properties using the NuSMV model checker. We present two types of experiments to validate the safety of the designed LD program. We also verify that the nondeterminism that appears as a result of the system behaviors can be formalized and used to represent logical assumptions for the properties that need to be verified.

Keywords: programmable logic controllers, formal methods, formal verification, model checking, timer function blocks

1. Introduction

Logic control system design using programmable logic controllers (PLCs) is currently becoming more complicated because of the various embedded features and functionality requirements that are now used. The main concern when designing such a control system, especially in manufacturing automation processes, is how to deal with safety and reliability. In industry, for example, formal methods are used as verification procedures to check whether a system satisfies the desired safety specifications [(1)] so that if any unintended behavior is identified, it can be eliminated before a mass production process is started.

Among several formal verification approaches, temporal logic model checking is a popular technique for verifying that a system satisfies its specifications. The system is usually formalized as a Kripke structure and the specification are described as a temporal logic formula. Model checking is intended to check that the Kripke structure is a model of the temporal logic specification. Otherwise, the model checker produces a counterexample that shows the location at which the model violates the specified temporal properties.

PLC formal verification is an essential procedure to verify the correctness of a control program prior to implementation so that any design errors can be amended. In many cases, a PLC program that represents a control system is often designed deterministically, such that the outputs are already predicted according to the given inputs, even if the same inputs are given repeatedly at different times. However, when the environment of the system is not fully predictable because of the presence of an underlying behavior or external input forces, the system can be considered nondeterministic. Nondeterminism often exists due to underspecification of the process design and development, or implementation choice.

In this paper, we introduce a formal modeling and verification of PLC nondeterministic behavior in which a ladder diagram (LD) is implemented. We provide an illustrative example using an arm pick-and-place model, and design the LD program for the arm, with the application of timer ON and timer OFF (namely TON and TOF, respectively) function blocks. The nondeterministic behaviors are configured from an arm grip condition and also from the logical formalism of both timer properties. By applying five temporal properties to the arm model, we verify the deterministic and nondeterministic behaviors, and present two experimental results for validation of safe arm operation. The contributions of this work are:

(1) safety verification and validation through experiments using an arm pick-and-place operational model with nondeterministic behaviors
(2) model verification using temporal properties with inclusion of nondeterministic behaviors from state operations and timer functions
(3) application of TOF timer function block for safety experiments with reachability and the implementation of liveness property checking

2. Related Studies

In a paper describing a comprehensive study of the application of formal methods, Seidner et al. [(2)] proposed a formalization technique for function block diagrams that preserve the behavioral semantics for system design application.

[*] Department of Electrical and Computer Engineering
Yokohama National University
79-5, Tokiwadai, Hodogaya-ku, Yokohama 240-8501, Japan
[**] Faculty of Electrical Engineering, Universiti Teknikal Malaysia Melaka (UTeM)
Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia

© 2013 The Institute of Electrical Engineers of Japan.

Moreover, an approach of reusable automation components (RACs) [3] that introduces a formal specification defining the correct use and behavior of industrial system's components has led to shorter development and modification times. In recent publication, the same authors propose and apply a work procedure of developing safety PLC programs with formal specifications that relevant for implementation in industry [4].

Additionally, the use of timing functions play an important role in PLC for real time applications. The formalization of timers application for PLC control programs at abstract level is explained thoroughly by Hai Wan et al. [5]. In this paper, they have formally proved the correctness of PLC programs by using Coq theorem prover, in which the timer control is applied. The same theorem prover tool was then utilized to formalize the semantics of Instruction List (IL), one of the common programming language in PLC, that also includes *on-delay timers* [6]. Several safety properties have been proved by using this semantics.

Prior to that, a formal semantics for LD programs that use timer function blocks was introduced in [7]. They deal with *Time ON delay* function blocks (TON) at a logical level. Some of our works presented in this paper are inspired from their approach. Instead of LD, Loeis et al. utilized IL to describe the control system behavior and conduct the verification using Binary Decision Diagram (BDD) and SAT techniques [8]. Rausch and Krogh [9] in their paper also presented the same approach by using Symbolic Model Verifier (SMV), an old version of NuSMV.

The studies in PLC modeling for logic verification was also conducted because of the awareness of the importance of safety and reliability of control system in industry [10]. Furthermore, an advanced method by performing model checking directly from IL was conducted, and the advantage is that the counterexample is produced in the IL itself [11]. The other language in PLC programming also is implemented for model checking approach for the purpose of evaluating the safety of PLC control systems [12].

Generally, control systems with deterministic specifications such as the ones designed using PLCs are easy to be verified. However, with complex and modern system implementations, nondeterministic behaviors due to concurrency, non-controllable elements etc. cannot be neglected [13]. To overcome the difficulties in dealing with nondeterminism, a method of test case generation from model checker counterexamples was proposed. This approach is based on modular model checking of mutant composition by fault modeling and specifications [14]. Gordon et al. implemented a test case derivation technique from nondeterministic models by applying counterexample evaluation [15].

3. Operational Semantics for PLC Program

We give an overview of operational semantics for PLC program, which is described in LD. This also can be considered as a formal definition of abstract behavior of a LD program. There are some restrictions apply for simplification of formal semantics definition, based on IEC 61131-3 standard [16]. However, still the considered LD program has taken into account all contacts described in the IEC 61131-3, including TON and TOF function blocks and set/reset output coils. We assume in detail that:

(1) all variables are boolean
(2) each LD rung is composed of a *testing part* (test contacts and combination wires) followed by an *assignment part* (output coils)
(3) the rungs are evaluated *sequentially*, from top to bottom

Finally, we assume that the LD program runs in a *cyclic PLC*. This behavior is described in the IEC 61131-8 standard.

4. Timer Function Blocks

In this section, we explain the operational semantics of two most common-used timer function in PLC, named as time ON delay (TON) and time OFF delay (TOF) function blocks.

4.1 Time ON Delay (TON) Time ON delay function block (afterwards referred as TON) is basically used to describe ON timing delays in LD programs. Their behavior is explained in detail in the IEC 61131-3 standard. An example of a TON and its operation is illustrated in Fig. 1. The TON is always used in the testing part of the rung. We give the standard definition of the TON behavior as follow: An internal float variable ET measures the *elapsed time* (increases along with the internal clock of the PLC) as long as input IN holds TRUE, and then output Q is FALSE. ET is set to 0 when IN is FALSE. It is also compared to input PT: if $ET = PT$, then Q is set to TRUE, and ET is no longer increased. Our formal model is an abstract simplification of the above behavior. We only keep a logical view of the trigger condition ($ET = PT$) without taking into account the explicit timing values.

We assume there are three internal states that can be observed from the abstract model of a TON: *inactive* (I), *running* (R) or *triggered* (T). The transition system from one to another state of the TON module is described in detail in 5.: a running (R) state of TON can either stay in state R or T, in a nondeterministic way. Q is set to TRUE when the TON is triggered. During some evolution of the TON block, the input IN holds FALSE, then the state of the TON is set to I.

4.2 Time OFF Delay (TOF) Similar to the TON function block (afterwards referred as TOF), an example of a TOF function and its operation is illustrated in Fig. 2. The standard behavior of the TOF is as follows: An internal float variable ET measures the *elapsed time* (increases along with the internal clock of the PLC) once the input variable IN becomes FALSE, and then output Q remains TRUE if $ET \neq PT$. ET is set to 0 when Q is FALSE. It is also compared to input PT: if $ET = PT$, then Q is set to FALSE, and ET is no longer increased.

Similar to TON, the assumption is that there are three internal states of a TOF function block:*inactive* (I), *running* (R) or *deactivated* (D). A running TOF can either be in state R or D, in a nondeterministic way.

These finite-state logical views of both TON and TOF lead to a simple model that can be smoothly integrated in the transition relation for any arbitrary logical system. The approach presented here has its limitations compare to quantitative modelings of the real time aspects, but it is safe: any safety property satisfied by the model is also true in the quantitative model based on timed-automata.

4.3 Integration of Timer Function Block and LD The operational semantics of LD has to be modified when a TON or TOF blocks are applied. Let say if a TON appears

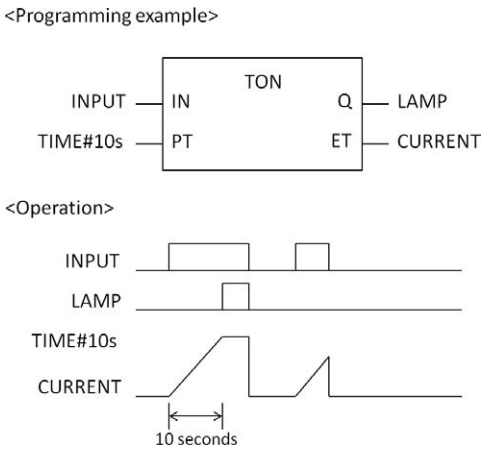

Fig. 1. TON and its operation

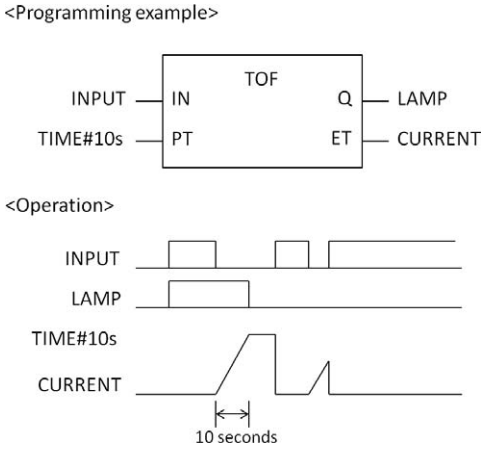

Fig. 2. TOF and its operation

in any rung, an additional internal variable encodes the internal state (Q, R or I) of that TON block is required. Evaluating the rung with a TON block requires the evaluation of the next state of the TON block, which may depend on the testing part. Here the *IN* input value is not uniquely determined since it may become TRUE or FALSE nondeterministically. Then the current value of output Q is known and usual evaluation of a rung can be pursued. The behavior can be modeled in the associated variable by inserting a new control point in the rung where a TON block occurs. If there are more than one TON block exist, they sequentially evaluated from left-to-right, top-to-bottom order.

5. Modeling and Experiments

This section reports an illustrative example, providing an arm pick-and-place system for formal modeling of deterministic and nondeterministic behaviors. Generally, the arm pick-and-place system is commonly used for product or material handling system. In our case, we use the arm model for experiments as in Fig. 3. We provide a plant model of the arm that describes the model configuration, then we design the LD program on the basis of its regular operation, and also integrating the program with TON and TOF function blocks. To verify whether the designed LD program meets the specified properties, we consider a number of temporal logic specifications, such as invariance and liveness property described in CTL formulae. We verify the arm system

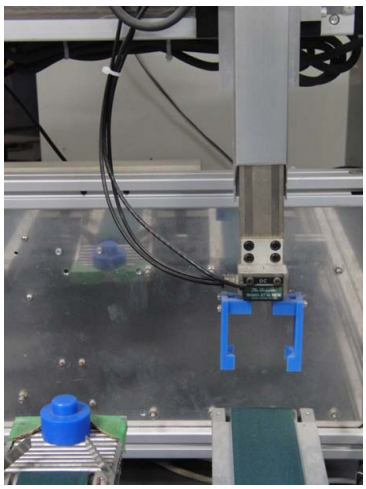

Fig. 3. Actual arm model

whether it satisfies the specified properties by using NuSMV tool, a SMV based extended model checker.

The arm pick-and-place system uses a MICREX-SX SPH series of PLC with D300win programming software (Fuji Electric Corporation), whereas the PLC programming tool is based on the IEC 61131-3 standard.

5.1 Experimental Setup The operational diagram is illustrated in Fig. 4 to facilitate the analysis and design of the LD program. The definition of states, where all variable x stands for *state inputs* and all variable u stands for *actuator outputs* are summarized in Table 1. All the state variables are represented with x_i and u_i. Here, we set u_5 and u_6 as internal variables because they are not the actual output variables of the system. We also integrate the LD program with a TON and two TOF function blocks. The internal states for both type of the timer function blocks are summarized in Table 2 and Table 3, respectively. We particularly focus on the implementation of nondeterministic behavior to grip condition and the timer functions, which are described in detail in 5.4 and Section 7.

5.2 Arm Pick-and-Place Operation A complete operation of the arm pick-and-place can be described in the following order:

(1) arm moves backward to the rear-top position from the front-top (home) position
(2) from the rear-top position, arm moves downward to the rear-bottom position
(3) at the rear-bottom position, arm hand picks and grips the product (state of the grip changes from close = FALSE, open = TRUE to close = FALSE, open = FALSE)
(4) arm moves upward to the rear-top position while gripping
(5) arm moves forward to the front-top position while gripping
(6) arm moves downward to the front-bottom position while gripping
(7) at the front-bottom position, arm places and releases the product (state of the grip changes from close = FALSE, open = FALSE to close = FALSE, open = TRUE)
(8) arm moves back to the home position and waits for

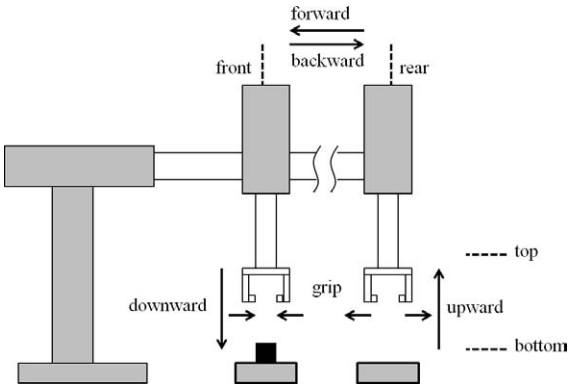

Fig. 4. Arm diagram

Table 1. States of arm

x_0	rear
x_1	front
x_2	bottom
x_3	top
x_4	open state (grip)
x_5	close state (grip)
u_0	backward
u_1	forward
u_2	downward
u_3	upward
u_4	grip
u_5	to pick
u_6	to place
u_7	grip_fall

next execution

5.3 Transition System of Arm Model To represent the full arm model as a finite state system, we describe it as a tuple $TS = \langle S, Act, \delta, I, AP, L\rangle$ where

S: set of states

Act: set of actions

δ: transition relation of $S \times Act \times S$

I: set of initial states

AP: set of atomic proposition

L: labeling function of $S \rightarrow 2^{AP}$

Then from the model, we obtain the set of states as in the following equations, represented by $s_0, s_1, \cdots, s_5$.

$$\begin{aligned} s_0 &= (x_1, x_3) \\ s_1 &= (x_0, x_3) \\ s_2 &= (x_0, x_2) \\ s_3 &= (x_1, x_2) \\ s_4 &= (x_0, x_2, \overline{x_4}, \overline{x_5}) \\ s_5 &= (x_1, x_2, x_4, \overline{x_5}) \end{aligned} \quad \cdots\cdots (1)$$

Therefore, the full model can be represented by the following equations.

$$S = \{s_0, s_1, s_2, s_3, s_4, s_5\} \quad \cdots\cdots (2)$$

$$Act = \{u_0, u_1, u_2, u_3, u_4, u_5, u_6\} \quad \cdots\cdots (3)$$

$$\begin{aligned} \delta = \{&(s_0, s_0), (s_0, s_1), (s_1, s_2), (s_2, s_4), (s_2, s_2), \\ &(s_4, s_4), (s_4, s_1), (s_1, s_0), (s_0, s_3), (s_3, s_5), \\ &(s_5, s_5), (s_5, s_0)\} \end{aligned} \quad \cdots\cdots (4)$$

Table 2. States of TON

State	Description
inactive (I)	state where the input variable *IN* is FALSE
running (R)	standby mode where the preset time is calculated until ET = PT
triggered (T)	the output variable *Q* is TRUE

Table 3. States of TOF

State	Description
inactive (I)	state where the input variable *IN* remains TRUE
running (R)	standby mode where the preset time is calculated until ET = PT
deactivated (D)	the output variable *Q* is FALSE

$$I = \{s_0\} \quad \cdots\cdots (5)$$

$$\begin{aligned} L = \{&(s_0, \{x_1, x_3\}), (s_1, \{x_0, x_3\}), (s_2, \{x_0, x_2\}), \\ &(s_3, \{x_1, x_2\}), (s_4, \{x_0, x_2, \overline{x_4}, \overline{x_5}\}), \\ &(s_5, \{x_1, x_2, x_4, \overline{x_5}\})\} \end{aligned} \quad \cdots\cdots (6)$$

Finally, we obtain the state transition diagram of the model as in Fig. 5.

5.4 Arm Nondeterministic Condition In normal situation of the arm operation for product pick-and-place as illustrated in Fig. 4, the arm moves from its home position and picks the product at the rear-bottom position. At this position, the arm hand grips the product and brings it to the front-bottom position before releases it and moves back to home position. Based on this pick-and-place movements, we assume that the nondeterministic situation might be occurred by considering the following conditions:

(1) a normal condition where arm hand continues to grip (close = FALSE, open = FALSE) the product until front-bottom position, or otherwise;

(2) an unexpected condition where the product fell down in the middle of the operation because of external force (state of the grip changes from close = FALSE, open = FALSE to close = TRUE, open = FALSE)

Both conditions are illustrated in Fig. 6.

6. NuSMV and CTL Temporal Logics

NuSMV is a model checker that has been reengineered from Symbolic Model Verifier (SMV), its earlier version, developed by Carnegie Mellon University (CMU). It is designed to be a well structured, open, flexible and documented platform for model checking. User can easily getting familiar with NuSMV because its code is relatively easy to modify, achieving outstanding efficiency in order to control the state explosion during verification process. The NuSMV input language is designed to allow for the description of finite state systems.

Requirements for verification of a finite state system are basically its state transition system and specifications that expressed in propositional temporal logics. Two useful temporal logics are *Computation Tree Logic* (CTL) and *Linear Temporal Logic* (LTL)[17]. Both of them can be described in NuSMV specifications. However, for application purpose, later in this paper we only use CTL temporal logics for automated verification to check if the model satisfies the

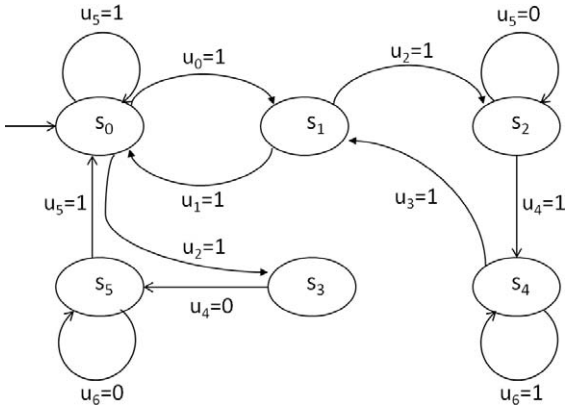

Fig. 5. State transition diagram for arm model

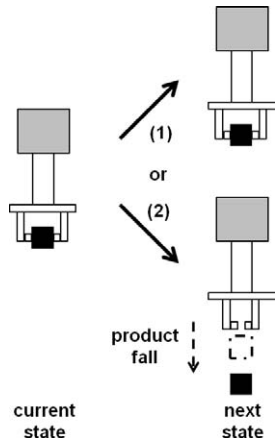

Fig. 6. Arm nondeterministic condition

specification of intended properties.

The syntax of CTL formulas is given by the following rules:

(1) any atomic proposition is a CTL formula;

(2) if α and β are CTL formulas, then $\alpha \bullet \beta$ and $\neg\alpha$ are CTL formulas, where $\bullet$ is any boolean connective ($\wedge$, $\vee$, $\cdots$);

(3) if α and β are CTL formulas, then **AF**α, **EG**α, **E[**$\alpha \cup \beta$**]** are CTL formulas.

The CTL formulas are constructed from *path quantifiers* and *temporal operators*. Intuitively, path quantifiers are represented by alphabet **(A)** which means "*for every path*" and **(E)** for "*there exists a path*". In a CTL formula, the path quantifiers can be mixed to form an intuitive temporal logic. Meanwhile, the temporal operators are represented by **(X)** for "*next time*", **(F)** for "*sometime in the future*", **(G)** for "*globally in the future*" and **(U)** for "*until*". Some of typical CTL formulas are:

- EF($p \wedge \neg q$): it is possible to get to a state where p holds but q does not hold.
- AG(EFp): from any state it is possible to get to p state.
- AG(p− >AFq): if p occurs, then it will be eventually q.
- $p \cup q$: p holds until q holds.

7. NuSMV Model

A part of NuSMV model for the arm operation is shown in Fig. 7, which describes the transition relations for all variables with nondeterministic cases, including the TON and TOF timer functions. The transition relations are written in terms of the *current* and *next* values of the state variables, using ***next*** and ***case*** syntax. Any current state/next state pair is in the transition relation if and only if it satisfies the formula. Below are some descriptions of the model.

- There are one TON and two TOF (TOF1 and TOF2) timer functions used in the model
- Each variable **T_ON**, **T_OF1** and **T_OF2** represents the state for TON, TOF1 and TOF2, respectively
- The transition relation for T_OF1 is nondeterministic ({TRUE,FALSE} in line 4 of Fig. 7 means the output can be TRUE or FALSE, nondeterministically)
- The *internal* state of TON is represented by nondeterministic variable **T_ON_state** as in transition relation in lines 53–60
- The *internal* state of TOF1 is represented by nondeterministic variable **T_OF1_state** as in transition relation in lines 8–15
- The *internal* state of TOF2 is represented by nondeterministic variable **T_OF2_state** as in transition relation in lines 31–38

This NuSMV model is directly transformed from the designed LD program for experimental verification. The LD program is illustrated in Fig. 8. Note that in the LD program,

- Variable T_ON is represented by TON function block at rung 8.
- Variable T_OF1 is represented by TOF_1 function block at rung 5.
- Variable T_OF2 is represented by TOF_2 function block at rung 4.

To model the TOF1, we evaluate the contact variables on the left part of TOF_1 function block. The transition relation is described in lines 1–6 of Fig. 7. Similar procedure is performed for TON and TOF2 where the transition relations are described in lines 46–51 and lines 23–29, respectively. The only difference is that the TOF1 is evaluated nondeterministically.

Meanwhile, the internal state of TOF1, represented by variable T_OF1_state as explained previously is also evaluated nondeterministically, leads to any one of three conditions, which is **inactive (I)**, **running (R)** or **deactivated (D)**. The transition relation for this variable is shown in lines 8–15. Similar case for internal state of TON (variable T_ON_state) and TOF2 (variable T_OF2_state) where the transition relations are described in lines 53–60 and lines 31–38, respectively.

Furthermore, the transition relation for **grip_fall** variable is shown in lines 17–21, which the value is determined by the internal condition of T_OF1_state. The actuation variable **grip** is evaluated by the internal condition of T_OF2_state, described in lines 40–44 and **forward** variable is determined by T_ON_state, as in lines 62–66.

8. Verification Results

We perform five temporal properties checking to verify whether the LD program for the arm model that is transformed into the NuSMV codes satisfies the temporal specifications below.

8.1 Invariance (Safety) Invariance or safety property is often characterized as *nothing bad should happen*. For

example, *always at most one state only for critical operation* is stating that bad thing (having two or more critical states simultaneously) should never occurs [18]. Another case of typical safety property is deadlock freedom. The safety properties are particularly *invariants*. Invariants are the properties that are given by a condition Φ for the states and require that Φ holds for all reachable states.

To verify this property, we provide two examples of invariant condition for the arm model:

(1) *product is gripped* (close = FALSE) and *product is fell* (close = TRUE) must not be ON simultaneously

(2) forward or backward actuation is not allowed when arm is at the bottom position

The verification results using the NuSMV model checker with CTL temporal logics for both specified conditions above are TRUE as in lines 1–3 of Fig. 9, which prove that the arm safe operation holds for both temporal specifications.

The second NuSMV specification above is actually too simple for verifying safety. In real implementation, the safety conditions for a system could be more complicated, with larger number of variables in a single specification. We prove this case by adding other variables to the NuSMV temporal specification in line 3, becoming a more complex specification as shown in lines 4 and 5 of the same figure, then verify whether the model satisfies this extended safety property. This specification is also intuitively TRUE, saying that the model is correct to satisfy the safety property.

8.2 Reachability An example of reachability property checking is by investigating that a system is *eventually resettable* no matter whatever happens in the middle of the operation. We verify this property by evaluating the situation of *whatever happens to arm operation, it is possible to get back to home position*. The home (initial) position for the arm pick-and-place is the *front-top* position. Verification result of this property checking is described in line 6 of Fig. 9, proving that for any circumstances of the arm state operation, it is possible to get back to home position. Another option for the reachability property checking is to reformulate the specification to be more strict, evaluating *whatever happens to arm operation, it is **always** return to home position,* described by specification in line 7 of Fig. 9 that also resulted to TRUE.

8.3 Safe Operation Experiment with Reachability In order to verify the reachability through actual system, we conduct two types of experiment, which are:

(1) a *safe operation* experiment; the arm stops at home position after product has fallen

(2) an *unsafe operation* experiment; the arm moves to the front-bottom position even after product has fallen and return to home position

to prove that the arm model is performing operation that satisfy the reachability properties described in lines 6 and 7 of Fig. 9. Result of experiments are described using the timing diagram that represents ON-OFF state changes of each variable involved. The timing diagram in Fig. 10 represents the safe condition experiment, while Fig. 11 shows the timing diagram for the unsafe condition experiment. Both timing diagram of experiments are captured by applying the same LD shown in Fig. 8.

We define the safe operation experiment by assuming that *when the product has fallen, the arm should stop or return to its original position without unnnecessary operation*. The result in Fig. 10 shows this condition. The product fallen at top-rear position just after TON starts to run and forward actuation occurs because TON is triggered. The arm reaches top-front position and stops there, waiting for the hand to be fully open, then returning to initial state (home position) by utilizing the *elapsed time* of TOF_1 for *grip_fall* and TOF_2

```
1   next(T_OF1) :=
2     case
3      (((!open & close) | grip_fall) & ((!rear | !top) & !front)):
4      {TRUE,FALSE};
5      TRUE : FALSE;
6     esac;
7
8   next(T_OF1_state) :=
9     case
10     T_OF1 : inactive;
11     !T_OF1 & T_OF1_state=triggered : running;
12     !T_OF1 & T_OF1_state=running : {running,deactivated};
13     !T_OF1 & T_OF1_state=deactivated : deactivated;
14     TRUE : T_OF1_state;
15    esac;
16
17  next(grip_fall) :=
18    case
19     T_OF1_state=deactivated : FALSE;
20     TRUE : TRUE;
21    esac;
22
23  next(T_OF2) :=
24    case
25     ((((rear & !front & bottom & !top & !to_pick & !to_place &
26     open) | grip) & !front & (!grip_fall | !rear)) | ((!front |
27     !bottom) & !to_place & !grip_fall & !open & !close)) : TRUE;
28     TRUE : FALSE;
29    esac;
30
31  next(T_OF2_state) :=
32    case
33     T_OF2 : inactive;
34     !T_OF2 & T_OF2_state=triggered : running;
35     !T_OF2 & T_OF2_state=running : {running,deactivated};
36     !T_OF2 & T_OF2_state=deactivated : deactivated;
37     TRUE : T_OF2_state;
38    esac;
39
40  next(grip) :=
41    case
42     T_OF2_state=deactivated : FALSE;
43    TRUE : TRUE;
44    esac;
45
46  next(T_ON) :=
47    case
48     (rear & !front & !bottom & top & !to_pick & !upward &
49     !backward & !downward ) : TRUE;
50     TRUE : FALSE;
51    esac;
52
53  next(T_ON_state) :=
54    case
55     !T_ON : inactive;
56     T_ON & T_ON_state=inactive : running;
57     T_ON & T_ON_state=running : {running,triggered};
58     T_ON & T_ON_state=triggered : triggered;
59     TRUE : T_ON_state;
60    esac;
61
62  next(forward) :=
63    case
64     T_ON_state=triggered : TRUE;
65     TRUE : FALSE;
66    esac;
```

Fig. 7. A part of NuSMV model for arm operation

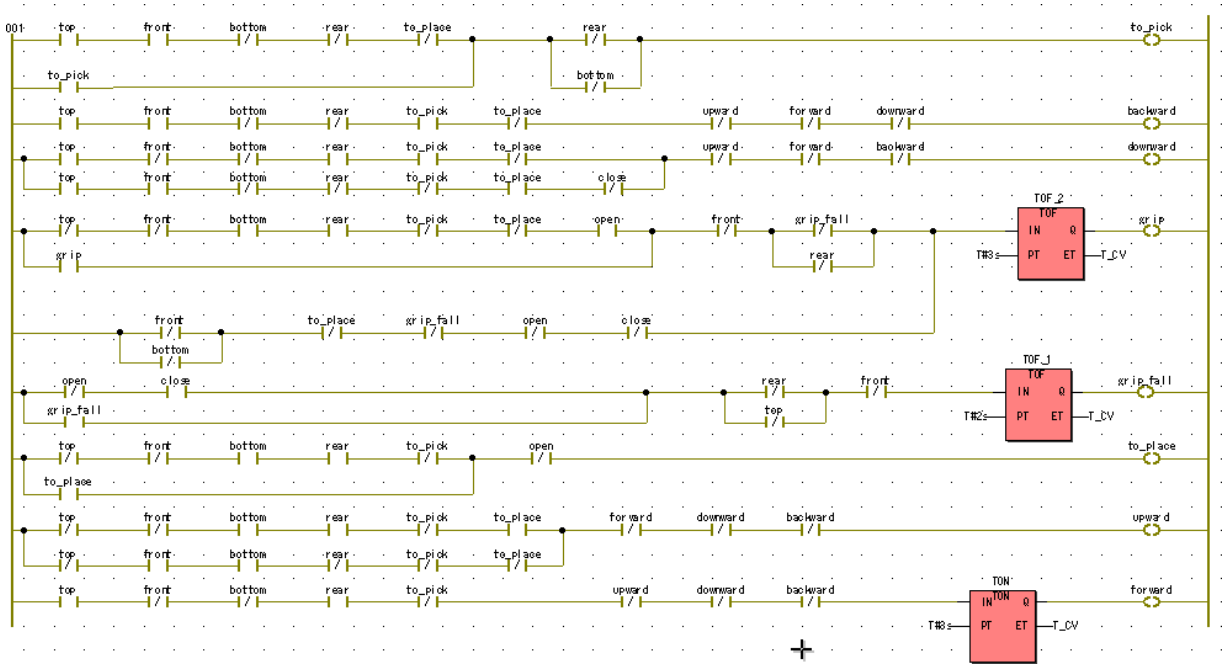

Fig. 8. LD for arm pick-and-place with integration of timer function blocks

```
1  -- specification AG !((!close & grip) & (close &
2     grip_fall)) is true
3  -- specification AG !(bottom & forward)  is true
4  -- specification AG !(((bottom & forward) & backward) &
5     (top & upward)) is true
6  -- specification AG (EF (front & top))  is true
7  -- specification AG (AF (front & top))  is true
8  -- specification AG (!close -> EF bottom)  is true
9  -- specification AG (close -> EF !(front & bottom))
10    is true
11 -- specification AG (T_ON_state = running ->
12    AF T_ON_state = triggered)  is false
13 -- specification AG (T_ON_state = running ->
14    EF T_ON_state = triggered)  is true
```

Fig. 9. Verification results

for *grip*.

For unsafe operation, we give a condition that the product has fallen not at the top-rear, but when *forward* actuation is occuring. Just after the product has fallen, the arm starts to fully close and reaches front-top position. However, this time the *downward* actuation is occured and TOF_1 starts to count for deactivation. A moment after, *grip* is fully open at front-bottom position and TOF_2 is deactivated. TOF_1 deactivation occurs just before the *upward* actuation and the arm returns to home position. In other words, a slight delay of product fallen condition causes the difference in the arm grip open-close timing, resulted to safe and unsafe movements of arm pick-and-place system.

In both of the timing diagrams, the horizontal axis represents time while vertical axis represents the ON-OFF state change for each variable used in the LD program.

8.4 Liveness As the safety properties specify that *something bad never happens*, the complementary properties can be specified as *something good* will happen in the future. This is called *liveness* properties. The violation of the liveness properties is run in infinite time, while for safety, the properties is violated in finite time, i.e., by finite system run.

A typical example of a liveness property is the requirement that certain events occur infinitely often. This description means the *good event* of a liveness property is a condition on the infinite behaviors. Based on the experimental model, we give two examples of liveness property to be verified stated by the conditions that:

(1) when the arm starts to grip, it is possible to reach bottom position, and conversely;

(2) when product is fell, the arm will not suppose to reach front-bottom position.

We formulate the CTL temporal formulas for both conditions in lines 8 and 9–10 of Fig. 9 respectively and the verification results indicate TRUE, proving that the liveness property holds for both cases. Similar to the case of reachability property described previously, the CTL temporal formula AG(p − > EFq) is sufficient to express the possibility of a required state holds in the future, rather than AG(p − > AFq). The latter formula is a stronger requirement, which interpreted as *when p occurs, then it will be eventually q*.

8.5 Liveness Property for Timer In section 5, we model the logical properties of the TON and TOF function blocks with the assumption that they only allow *eventual* triggering or deactivating, but not enforcing them since we do not consider the real timing properties. For example, the TON function block is not always being triggered in the future, but eventually be triggered when its internal state shows *running (R)*. To verify the property described, we stated the CTL temporal specification as shown in lines 11 and 12 of Fig. 9. As a result, the specification is FALSE, proving that the TON is not always being triggered in the first place. This result is actually produces counterexamples to indicate the location where the TON model is not hold for the liveness property described (the details of counterexample produced is not discussed in this paper). Therefore, in this case we may use the definition of *fairness constraint* to adequately utilize the liveness property, which means that the TON (TOF as well)

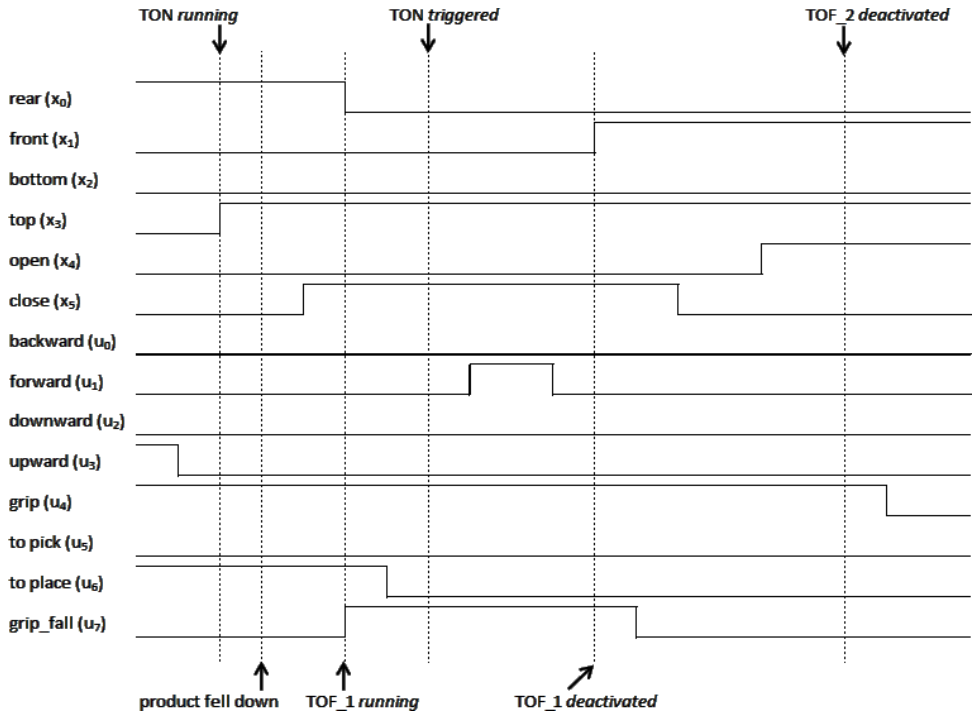

Fig. 10. Safe condition experiment (the arm stops at home position after product has fallen)

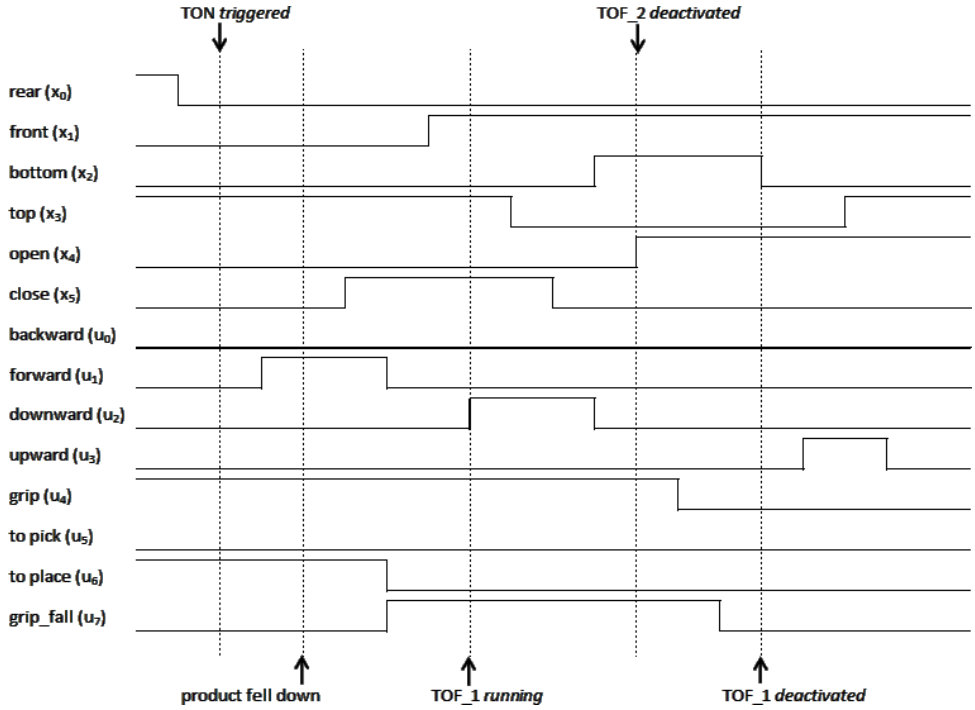

Fig. 11. Unsafe condition experiment (the arm moves similar to normal operation even the product has fallen)

will eventually be triggered upon satisfying the input variable condition.

8.6 Fairness for Liveness As previously explained in subsection 8.5, the operational model of both TON and TOF described only allows eventual triggering or deactivating. The restriction of this nondeterministic behavior would be appropriate by assuming that the TON will eventually trigger by associating with the definition of fairness constraint (restriction the set of possible runs). For example, we can state that if the TON is in the (R) state and infinitely often evaluated, then if the input variable holds, the TON will eventually trigger (T). The logical evaluation in lines 13 and 14 of Fig. 9 describes this condition. In the LD program for the arm model, once the TON is triggered, the arm moves to the front position.

In general, this type of fairness assumption is crucial to prove liveness, as well as other properties stating that the timer is making some progress even there is nondeterminism in the transition system. The fairness property is actually to resolve the nondeterminism without ignoring possible

options. We assume that the specified fairness constraint for the TON is an *unconditional fairness*, because both (R) as well as (T) states are ready to be execute infinitely often.

Similar situation is observed from the TOF liveness property because it depends on the implementation choice of whether the arm might be fully closed (close = TRUE) after the product has fallen, or the arm continues gripping the product (close = FALSE) before placement. The difference compares to TON is that its input variable need to be FALSE before the selection of {*running*, *deactivate*} is made.

Five temporal properties verification for the arm model explained above are examples to prove the intuitiveness of NuSMV model with specifications, as well as to validate that the designed LD program is performing desired operation that meets the specified requirements. The nondeterministic behaviors from the arm grip condition and the timer logical properties are formally verified using the CTL temporal specifications. For the case of timer, we take an example of TON behavior, and suppose the TOF is also operated in a similar condition. In addition, from the experiments we prove that the reachability property is fulfilled for both safe and unsafe operations.

9. Conclusion

In this paper, we presented a formal method approach, which is to formally verify a model with nondeterministic conditions occured from the system behaviors. We use an example of a material handling system, which is an arm pick-and-place model to verify five substantial temporal properties by using NuSMV model checking tool, and also conducted safety experiments to check that the designed LD program for the arm model satisfies the predetermined safety condition. The experimental results also prove the reachability property, which the arm return to its initial state, regardless of safe or unsafe operation. The logical properties of TON and TOF timer function blocks can be used to automatically prove some liveness properties with fairness constraint evaluation. The formalism of these timer functions from logical viewpoint avoids the complexity of model checking method by timed-automata. We conclude that the NuSMV model with CTL temporal specifications used in this work can be applied to validate the correctness of LD programs, as well as to verify the nondeterministics behaviors of logic control system.

Acknowledgment

The author would like to thank Universiti Teknikal Malaysia Melaka (UTeM) and Ministry of Higher Education, Malaysia for their financial support and advice in pursuing a Ph.D. study at Yokohama National University.

References

(1) G. Frey and L. Litz: “Formal methods in PLC programming”, in Proc. 2000 IEEE Conf. Syst. Man, Cybern., Vol.4, pp.2431–2436 (2000)

(2) C. Seidner and O.H. Roux: “Formal Methods for Systems Engineering Behavior Models”, IEEE Trans. Industrial Informatics, Vol.4, No.4, pp.280–291 (2008)

(3) O. Ljungkrantz, K. Akesson, M. Fabian, and C. Yuan: “Formal Specification and Verification of Industrial Control Logic Components”, *IEEE Trans. Automation Sci. and Engineering*, Vol.7, No.3, pp.538–548 (2010)

(4) O. Ljungkrantz, K. Akesson, C. Yuan, and M. Fabian: “Towards Industrial Formal Specification of Programmable Safety Systems”, *IEEE Trans. Control Systems Technology*, Vol.20, No.6, pp.1567–1574 (2012)

(5) H. Wan, G. Chen, X. Song, and M. Gu: “Formalization and Verification of PLC Timers in Coq”, in 33rd Annual IEEE Int. Computer Software and Applications Conf., USA (2009)

(6) S.O. Biha: “A formal semantics of PLC programs in Coq”, IEEE Computer Software and Applications, COMPSAC (2011)

(7) O. Rossi and Ph. Schnoebelen: “Formal modelling of timed function blocks for the automated verification of ladder diagram programs”, in Proc. Int. Conf. Automation of Mixed Processes and Hybrid Dynamic Systems, Germany, pp.177–182 (2000)

(8) K. Loeis, M.B. Younis, and G. Frey: “Application of Symbolic and Bounded Model Checking to the Verification of Logic Control Systems”, in 10th IEEE Conf. Emerging Technologies and Factory Automation (2005)

(9) M. Rausch and B.H. Krogh: “Formal verification of PLC Programs”, in American Control Conference, USA (1998)

(10) II Moon: “Modeling programmable logic controllers for logic verification”, *IEEE Control Systems Magazine*, Vol.14, No.2, pp.53–59 (1994)

(11) B. Schlich, J. Brauer, J. Wernerus, and S. Kowalewski: “Direct Model Checking of PLC Programs in IL”, in 2nd IFAC Workshop on Dependable Control of Discrete Systems (2009)

(12) R. Huuck, B. Lukoschus, and N. Bauer: “A Model-Checking Approach to Safe SFCs”, in IMACS Multiconference on Computational Eng. in Syst. Applications, France (2003)

(13) A. Jain, K. Nelson, and R.E. Bryant: “Verifying Nondeterministic Implementations of Deterministic Systems”, in Formal Methods in CAD, Springer-Verlag, pp.109–125 (1996)

(14) S. Boroday and A. Petrenko: “Can a Model Checker Generate Tests for Non-Deterministic Systems?”, Electrical Notes in Theoretical Computer Science 190, Elsevier, pp.3–19 (2007)

(15) G. Fraser and F. Wotawa: “Test-Case Generation and Coverage Analysis for Nondeterministic Systems Using Model-Checkers”, in Int. Conf. on Software Engineering Advances (2007)

(16) IEC, Programmable Controllers-Part 3: Programming Languages, 2nd ed. ser. International standard IEC 61131-3, International Electrotechnical Commission (2003)

(17) H. Iwashita, T. Nakata, and F. Hirose: “CTL model checking based on forward state traversal”, in Proc. of IEEE/ACM Int. Conf. of Computer Aided Design, Washington (1996)

(18) C. Baier and J.P. Katoen: “Principles of Model Checking”, The MIT Press, Cambridge (2008)

Saifulza Alwi (Non-member) received a Master degree in Computer Science and Systems from Kyushu Institute of Technology in 2003, and is presently a Ph.D. candidate at Yokohama National University. He is also currently attached to Universiti Teknikal Malaysia Malaysia (UTeM) as a lecturer. His research interests include automated V&V of logic control system and PLC software technology for factory automation.

Yasutaka Fujimoto (Member) received a Ph.D. degree in electrical engineering from Yokohama National University in 1998, where he is currently a professor. His research interests include industrial automation, e.g., optimization, production planning/scheduling, discrete event systems, manufacturing automation, motion control and robotics.