Symbolic Analysis of Programmable Logic Controllers

Hehua Zhang, Yu Jiang, William N.N. Hung, Xiaoyu Song, Ming Gu, and Jiaguang Sun

Abstract—Programmable Logic Controllers (PLC) are widely used in industry. The reliability of the PLC is vital to many critical applications. This paper presents a novel approach to the symbolic analysis of PLC systems. The approach includes, (1) calculating the uncertainty characterization of the PLC system, (2) abstracting the PLC system as a Hidden Markov Model, (3) solving the Hidden Markov Model with domain knowledge, (4) combining the solved Hidden Markov Model and the uncertainty characterization to form a regular Markov model, and (5) utilizing probabilistic model checking to analyze properties of the Markov model. This framework provides automated analysis of both uncertainty calculations and performance measurements, without the need for expensive simulations. A case study of an industrial, automated PLC system demonstrates the effectiveness of our work.

Index Terms—PLC, Hidden Markov model, probabilistic analysis

1 Introduction

PLCs are widely used in industry. A PLC system is composed of a microprocessor, memory, input devices, output devices, and an embedded PLC program. The embedded PLC program is executed through periodic scanning. Signals are received from input devices such as sensors or switches, and sent to output devices such as actuators. In each cycle, the inputs are sampled and read first. Then, the embedded PLC program instructions are executed. Finally, the outputs are updated and sent to actuators. Most of the PLC systems are safety critical, especially those in application areas such as nuclear power plants or spaceport device controls.

There have been many studies that have modeled and verified PLC programs. PLC programs are usually modeled as automata [1] or Petri nets [2]. Formal verifications are then proposed for analysis. For example, [3], [4] translated the PLC program into an SMV model, and checked the properties with the model checker. Most prior works considered only static PLC programs, and verified the functional numeric properties by exploring the state space of the transferred model. However, they did not consider the effect of the operating environment on the PLC program. The uncertainty errors [5] caused by noise, environment, or hardware were neglected, thus it is difficult to carry out some performance measurements with these methods.

Uncertainty error analysis and performance measurements are important to safety critical systems. Traditionally, the assessment of uncertainty errors was achieved by utilizing combinatorial methods such as the Fault Tree method [6] and Reliability Block Diagrams [7]. The Fault Tree method involves specifying a top event to analyze, such as the failure of the system, followed by identifying all associated events that could lead to the top event. It can be solved with techniques such as Binary Decision Diagrams [8]. Analysts also extend the traditional Fault Tree method by associating a particular Markov process to a leaf node [9]. Embedding the Markov process into a leaf node makes it easy to model dynamic systems. A Reliability Block Diagram is a graphical depiction of the system components and connectors. Reliability Block Diagrams can be used to determine the reliability of a system, when the reliability of each component is given. Recently, a more flexible framework called Bayesian Network, has been applied to system reliability analysis [10], [11]. It is based on the theory of graphical and probabilistic reasoning. These methods present the distribution of system components and events with high level abstraction.

Corresponding to the reliability assessment methods used at the macro-system component level, there are also many methods and tools developed for the nano-circuit level. The recent works on the reliability assessment of logic circuits are mostly based on Bayesian Network [12], Dynamic Bayesian Network [13], Probabilistic Transfer Matrices [14], [15], and the probabilistic model checker PRISM [16], [17]. In the Bayesian Network based algorithm, the circuits are translated into a Bayesian Network to capture the dependencies between each gate, and to construct a conditional probability table for each node. In this method, a long runtime problem will arise due to the large, conditional probability tables. In Probability Transfer Matrix based methods, probabilistic transfer matrices are constructed for each gate. Matrix multiplications and tensor

- *H. Zhang, M. Gu, and J. Sun are with the School of Software, Tsinghua Information Science and Technology National Laboratory (TNLIST), Tsinghua University, Beijing 100084, China. E-mail: {zhanghehua, guming, sunjg}@mail.tsinghua.edu.cn.*
- *Y. Jiang is with the Department of Computer Science and Technology, Tsinghua Information Science and Technology National Laboratory (TNLIST), Tsinghua University, Beijing 100084, China. E-mail: jiangyu10@mails.tsinghua.edu.cn.*
- *W.N.N. Hung is with Synopsys Inc., Mountain View, CA 94035. E-mail: william_hung@alumni.utexas.net.*
- *X. Song is with the Electrical and Computer Engineering (ECE) Department, Portland State University, Portland, OR 97207. E-mail: song@ece.pdx.edu.*

Manuscript received 16 June 2012; revised 24 Nov. 2012; accepted 05 June 2013. Date of publication 13 June 2013; date of current version 12 Sep. 2014. Recommended for acceptance by C. Metra. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TC.2013.124

product operations are then adopted to combine the probabilistic transfer matrices, with the help of an Algebraic Decision Diagram. The storage and manipulation of a large Algebraic Decision Diagram also leads to significant runtime for circuits. PRISM based approaches have been shown to be useful in analyzing NAND multiplexing, fault-tolerant architectures, with the understanding that the actual values of inputs and outputs of each logic block are not important. However, it does not generally address the circuit-specific failures due to the dedicated observations. When applied to other problems, it will lead to the well-known state space explosion problem.

A PLC system is a type of embedded system consisting of some hardware components and an embedded control program. The complex relationships between the hardware components and the control program will need to be considered. For example, different embedded control programs will give rise to different times for memory reading and writing. This will lead to different reliability values. In this paper, we propose a more straightforward method to handle these relationships. We translate the embedded control program into a logic expression, and built an abstract syntax tree for the logic expression. The probabilities of hardware uncertainty errors are embedded into the probabilities of the operators and the parameters in the expression. We process those operators from the bottom, up. The final probability of the root operator denotes the reliability of the system. Based on the reliability of the PLC system, we can also carry out some performance measurements. Generally speaking, the main contributions of this paper include:

- A complete symbolic framework to address the reliability assessment and the performance measurements of PLC systems.
- A new approach to characterize the uncertainty errors of PLC systems in a probabilistic way, which harmonizes the complex relationships between hardware component errors and the embedded control program.
- Presentation of how to abstract a PLC system as a Hidden Markov Model, and to extend the Baum-Welch method to solve the Hidden Markov Model based on the domain knowledge of its dedicated operating environment.
- Combining the solved Hidden Markov Model and the static reliability of the system to form a regular Markov Model, and to do some reward based probabilistic model checking. The performance measurements of the probabilistic model checking are more accurate and closer to real-world run-time situations.

Background concepts are introduced in Section 2, and a symbolic framework for a formal analysis of PLC systems is presented in Section 3, including an introduction of the main procedures such as uncertainty calculation, Hidden Markov Model construction, and reward based model checking. Verification of the framework with a case study is presented in Section 4, and conclusions are presented in Section 5.

2 Background

In this section, we introduce some basic concepts of the PLC program, and the Hidden Markov Model. The International Electrotechnical Committee [18] has defined four standard programming languages for PLC: Ladder diagram, Functional block diagram, Instruction list and Structured text. This paper focuses on the ladder diagram, which is a popular programming language in PLC applications.

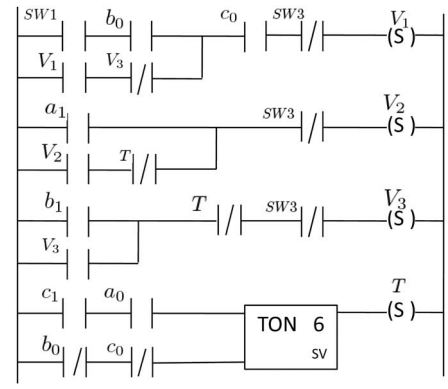


Fig. 1. A simple ladder program.

2.1 Ladder Diagram

The Ladder diagram is a widely used graphical programming language for PLC. The language itself can be seen as a set of connections between contacts and coils. If a path can be traced between the left side of the rung and the output, the rung is true and the output coil storage bit is asserted to be true. If no path can be traced through asserted contacts, the output coil storage bit is asserted to be false. Fig. 1 shows a simple ladder program with several common instructions. It is made up of four ladder rungs.

The symbol $-||-$ is a normally open contact, representing a primary input. When the value of V_1 is 1, the contact will stay in the closed state and the current will flow through the contact. The symbol $-|/|-$ is a normally closed contact. When the value of V_3 is 0, the contact will stay in the closed state, and the current will flow through the contact. The symbol $-||-\,||-$ represents a serial connection of two contacts. In the third rung, V_3 and b_1 are connected in parallel. If either V_3 or b_1 has the value 1, the current will flow through the trace and the output coil storage bit will be asserted to be true. The symbol $\mathbf{S}/\mathbf{R}$ is the set-reset instruction. When the results of the ladder rung turns out to be true, $\mathbf{S}$ will refresh the corresponding coil storage bit with 1 and $\mathbf{R}$ will refresh the bit with 0. Detailed explanations and other instructions can be found in [18].

2.2 Hidden Markov Model

The simplest Markov model is the Markov chain. The Markov chain is a random process with the property that the next state depends only on the current state. It models the state of a system with a random variable. The value of the variable changes over time. A discrete time Markov model can be defined as a tuple $\langle S, \pi, A, La \rangle$, where

- $S = \{S_1 \cdots S_N\}$ is a set of states. We use q_t to denote the state of the system at time t ($t \in \mathbf{N}^+$).
- $\pi = \{\pi_1 \cdots \pi_N\}$ is the initial state distribution, where $\pi_i = P(q_1 = S_i)$ is the probability that the system state at time unit 1 is S_i.
- $A = \{a_{ij}\}(\forall i, j \in N)$ is the state transition probability matrix for the system. $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$ denotes the probability that the system transfers from the state S_i to S_j.
- La is a set of propositions labeling states and transitions: $\{S_i, a_{ij}\} \rightarrow La$, where $1 \leq i, j \leq N$.

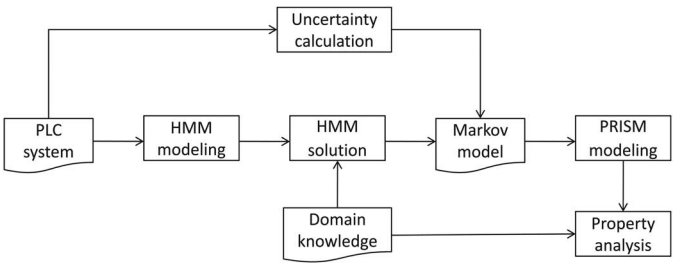


Fig. 2. Framework for the analysis of PLC systems.

In a regular Markov model, the state is directly visible to the observer, and the state transition probabilities are the only parameters. Each state corresponds to a physical event. This model is too restrictive to be applied to many problems of interest, including those cases where each state corresponds to many physical events, and the observation is a probabilistic function of the state. As a result, we utilize the Hidden Markov Model. In a Hidden Markov Model, the state is not directly visible, but the output observations which depend on the state are visible. Each state has a probability distribution over the possible output observations. Therefore, the observation sequence generated by a Hidden Markov Model gives some information about the sequence of the states.

Formally, a Hidden Markov Model (HMM) can be defined as a tuple $M = \langle S, O, \pi, A, B, La \rangle$. The items S, π, A and La are the same as previously defined. The other two components are defined as:

- $O = \{O_1 \cdots O_M\}$ is a set of observations that the system can generate. We use v_t to denote the observation of the system at time t ($t \in N^+$).
- $B = \{b_{ik}\}$ is the observation state probability matrix. $b_{ik} = P(v_t = O_k | q_t = S_i)(\forall S_i \in S, O_k \in O)$ is the probability that the system generates observation O_k in state S_i.

Given an observation sequence Q_o, there are three basic problems that must be solved for the Hidden Markov Model in real-world applications:

- How to calculate probability $P(Q_o|M)$ efficiently when the parameters of M are known.
- How to choose a state sequences that produce the observation sequence Q_o when the parameters of M are known.
- How to adjust the model parameters A and B to maximize the probability $P(Q_o|M)$.

In our framework, we need to focus on the third problem. We attempt to optimize the model parameters so as to best describe how a given observation sequence will turn out. It allows us to optimally adapt model parameters to observed training data, in order to create the best model for real phenomena. The problem has been solved using an iterative procedure such as the Baum-Welch method [19]-[21] and equivalently the EM method [22]-[24], or by using gradient techniques [25]. In this paper, we utilized the method introduced by Baum-Welch. It appears that the physical meaning of various parameter estimations can be easily visualized with this procedure. If there is enough domain knowledge based on simulations, we can solve the Hidden Markov Model with an easier method presented in Section 3.3.

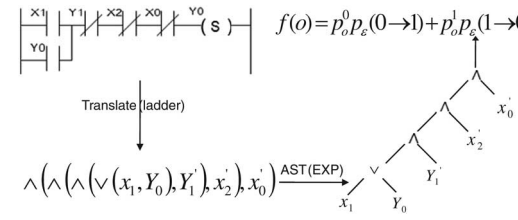


Fig. 3. An example of the uncertainty calculation.

3 Symbolic Framework

In this section, we present a symbolic framework for the reliability calculation and performance measurements of PLC systems. The framework mainly contains three main steps defined as follows: (1) The uncertainty characterization of the PLC system with static probabilistic calculations, which can reflect the relationships between the reliability of the hardware components and the embedded control program, (2) Hidden Markov Model construction and solving, which reflects the actual operational environment of the PLC system, (3) Reward based probabilistic model checking, which analyzes the performance properties of the system on the combined Markov model. Detailed information about the components of the framework is shown in Fig. 2. We modeled a PLC system by HMM, and solved the HMM with different types of domain knowledge. Then, we combined the solved HMM with the effect of uncertain errors to get a regular Markov model. Finally, we described the regular Markov model in PRISM, described some properties based on domain knowledge, and executed reward based property analysis to obtain performance measurements.

3.1 Uncertainty Calculations for the PLC

A PLC program is executed with a periodic scanning mechanism. Each cycle consists of reading the sampled inputs, executing the PLC program instructions, and updating the outputs to actuators, sequentially. The uncertainty characterization calculation refers to evaluating the effects of errors caused by input sampling and program execution. Sampling errors happen when the actual input is 1 (or 0), but the sensor samples 0 (or 1). Program execution errors happen when the output of each ladder logic is 1 (or 0), but the actual output of the logic turns out to be 0 (or 1). The program execution errors mainly occur at the serial connection AND logic ($a \wedge b$) and the parallel connection OR logic ($a \vee b$). The probabilities of the two execution errors depend on the environment, the hardware, etc.

The uncertainty calculation is based on our previous work presented in [26], and divided into three steps: (1) defining the output of each ladder rung on the contacts and the special instructions, such as timers and counters, as they are connected by ladder logic ($a \wedge b$, $a \vee b$), (2) building an abstract syntax tree for the output expression, and (3) processing the sampling errors and executing through the abstract syntax tree from the bottom up, until the root node is reached. An example is shown in Fig. 3. First, we applied the **Translate (ladder)** algorithm to translate the ladder program into an expression. Then, we executed the **AST(exp)** algorithm to build an abstract syntax tree for the expression, shown at the bottom right of Fig. 3. Finally, we applied the formula $f(o)$ at the top right of the figure to process each node. The formula $f(o)$ will be explained in later paragraphs.

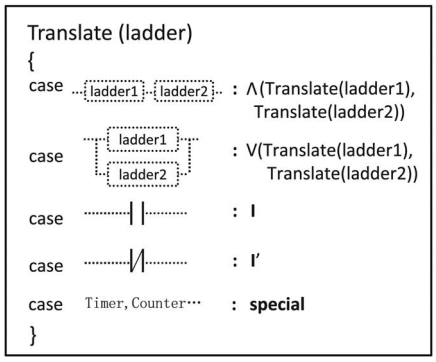


Fig. 4. The logic expression generation algorithm.

The first step can be executed with the **Translate(ladder)** algorithm shown in Fig. 4. The algorithm translates the PLC ladder program into a logic expression. The contacts and connection logic are translated into the operators and operands in the expression. In the first case in Fig. 4, ladder2 denotes the minimal ladder logic block that is serially connected to the rest of the ladder logic block. In the second case, ladder1 and ladder2 denote the maximal ladder logic blocks that are in parallel connections. We use the expression Exp, which consists of $\wedge$, $\vee$, $\mathbf{I}$ and $\mathbf{I}'$, to denote the translated result. The symbol $\wedge$ represents the serial connection AND logic in the ladder. The symbol $\vee$ represents the parallel connection OR logic in the ladder. The symbol $\mathbf{I}$ denotes the normally open contact, while $\mathbf{I}'$ denotes the normally closed contact in the ladder and **special** denotes the special instructions. The result of this step is an expression of the ladder rung.

In the second step, we construct an abstract tree for the output expression in the first step. We use the syntax-directed translation method to parse the expression. The syntax of the output expression is:

$$Exp \rightarrow \wedge(Exp, Exp)|\vee(Exp, Exp)|\mathbf{I}'|\mathbf{I}|\mathbf{special}.$$

The main structure of the recursive-descent parsing algorithm for the expression is shown in Fig. 5. The algorithm makes use of the data structure:

$$Struct\, node\{char * value; node * left; node * right;\}$$

The sampling of input devices and logic execution of the PLC microprocessor are mapped to nodes of the tree in a structured manner. The result of this step is an abstract syntax tree, which is also a binary tree. The leaf nodes are contacts and special instructions, while the other nodes are logic connections.

The third step processes the sampling errors and the logic execution errors individually, from the bottom node up to the root of the abstract syntax tree. For each node in the tree, there are two kinds of error sources, (1) those inherited from their child's nodes, and (2) those caused by logic execution of the PLC microprocessor. We should combine the two error sources to find the error which will be propagated to its parent node. The propagation error $P_\epsilon(L_{0\rightarrow1})$ means that the output of the ladder logic L should be 0, but the actual output is 1, and $P_\epsilon(L_{1\rightarrow0})$ means that the output of the ladder logic L should be 1, but the actual output is 0 ($(L \in \{\wedge, \vee\})$).

First, we assume that there are no ladder logic execution errors for the PLC microprocessor. In this case, we need to consider the first kind of error source.

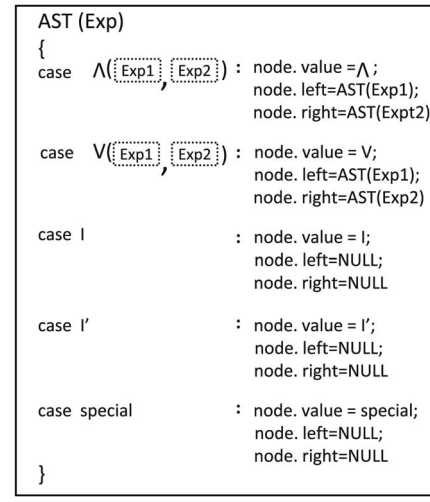


Fig. 5. Abstract syntax tree construction algorithm.

Take a ladder logic node $\vee(I, J)$ from the tree as an example. It has two child nodes, I and J. As a result, the child node I will propagate the error probability $P_\epsilon(I_{0\rightarrow1})$ and $P_\epsilon(I_{1\rightarrow0})$ to the node $\vee$. The other child node J will also propagate the error probability $P_\epsilon(J_{0\rightarrow1})$ and $P_\epsilon(J_{1\rightarrow0})$ to the node $\vee$. We need to combine these four error probabilities to get $P(\vee_{0\rightarrow1})$ and $P(\vee_{1\rightarrow0})$ due to their child nodes' errors. According to the semantics of the ladder logic $\vee$, when the results of the child nodes I and J are 0 (denoted by P_{00}), the result of the node $\vee$ is 0 (denoted by $P(0)$). We find that $P(0) = P_{00}$ and $P(1) = P_{11} + P_{10} + P_{01}$. If I propagates an error $P_\epsilon(I_{0\rightarrow1})$ and the result of node I changes from 0 to 1, the node $\vee$ will inherit this error and the result of the node $\vee$ will change to 1 (denoted by $P^{(0,0)}_{(0\rightarrow1)}$), which is a contribution to $P(\vee_{0\rightarrow1})$. Other cases and probabilistic logic are similar. Hence, the error transmission mode of the logic $\vee(I, J)$ can be described in the lemma below. It shows how the ladder logic $\vee$ inherits errors from its child nodes. The case for the ladder logic $\wedge$ is similar.

Lemma 1. *The transmission error that the current node $\vee(I, J)$ inherits from its child nodes is:*

$$P(\vee_{0\rightarrow1}) = P^{(0,0)}_{(0\rightarrow1)},$$
$$P(\vee_{1\rightarrow0}) = P^{(1,1)}_{(1\rightarrow0)} + P^{(0,1)}_{(1\rightarrow0)} + P^{(1,0)}_{(1\rightarrow0)}.$$

Proof. According to the semantics of the ladder logic $\vee(I, J)$, the two kinds of transmission probabilities and the value of the probabilities can be defined as follows.

$$P^{(I,J)}_{(1\rightarrow0)} = \begin{cases} P_{11}P_\varepsilon(I_{1\rightarrow0}) \cdot P_\varepsilon(J_{1\rightarrow0}) & (I,J) = (1,1), \\ P_{01}(1 - P_\varepsilon(I_{0\rightarrow1})) \cdot P_\varepsilon(J_{1\rightarrow0}) & (I,J) = (0,1), \\ P_{10}P_\varepsilon(I_{1\rightarrow0}) \cdot (1 - P_\varepsilon(J_{0\rightarrow1})) & (I,J) = (1,0). \end{cases}$$

$$P^{(I,J)}_{(0\rightarrow1)} = \begin{cases} P_{00}(P_\varepsilon(I_{0\rightarrow1}) + P_\varepsilon(J_{0\rightarrow1}) - \\ P_\varepsilon(I_{0\rightarrow1}) \cdot P_\varepsilon(J_{0\rightarrow1})) & (I,J) = (0,0). \end{cases}$$

The sums of these values are the two kinds of transmission probabilities from the child nodes. □

We now need to combine the first kind of error source (transmission error denoted by $P(L_{1\to0})$ and $P(L_{0\to1})$) with the second kind of error source (the ladder logic execution error of the PLC microprocessor, denoted by ε) to get the final propagation error (denoted by $P_\varepsilon(1\to0)$ and $P_\varepsilon(0\to1)$).

Theorem 1. *The final propagation error that the current node L transmits to its parent node is*

$$P_\varepsilon(0\to1)=\begin{cases}(1-\varepsilon)(P(L_{0\to1})/P(0))+\\ \varepsilon(1-P(L_{0\to1})/P(0)) \quad L\in(\vee,\wedge),\end{cases}$$
$$P_\varepsilon(1\to0)=\begin{cases}(1-\varepsilon)(P(L_{1\to0})/P(1))+\\ \varepsilon(1-P(L_{1\to0})/P(1)) \quad L\in(\vee,\wedge).\end{cases}$$

Proof. The first formula shows how a ladder logic L propagates $P_\epsilon(L_{0\to1})$ to its parent node. There are two possible outcomes, (1) the PLC microprocessor execution of this ladder logic node is not in error and the ladder logic node inherits $P(L_{0\to1})$ from its child nodes, and (2) the PLC microprocessor execution of this ladder logic node is in error and the ladder logic node does not inherit $P(L_{0\to1})$ from its child nodes. The sum of the two cases is the result. The proof for the second formula is similar. □

We can then apply theorem 1 to process each node in the abstract tree, until the root node is reached. The leaf nodes are contacts and special instructions. The error probability of the contacts can be set as the reliability of the corresponding sensors, while the error probability of the special instructions can be set according to the work in [26] as a combination of the basic execution of the microprocessor. Then, the uncertainty characterization of this ladder rung can be defined as:

$$f(o)=P_o^0P_\varepsilon(0\to1)+P_o^1P_\varepsilon(1\to0).$$

The first factor denotes the probability that the output of this ladder rung should be 0 (denoted by P_o^0), but the root node of the abstract syntax tree propagates a $P_\varepsilon(0\to1)$ error. The second factor is similar.

Since each ladder diagram could have several ladder rungs, the uncertainty characterization of the whole PLC system is defined as

$$f=1-\prod_{i=1}^{i=n}(1-f(o)_n).$$

This expression denotes that there is at least one ladder rung in error, and $f(o)_n$ represents the uncertainty characterization of the n ladder rung. This overall uncertainty characterization represents the probability that the PLC system will reach an abnormal state from any normal state.

3.2 The Construction of the Hidden Markov Model

Unlike existing methods which only consider static PLC programs, we have constructed an HMM to reflect the operating environment of a PLC system. Our model manifests the dynamic characteristics of all the possible execution paths of a PLC system. HMM depicts how a PLC system transfers from one state to another with a hidden probability. As mentioned in Section 2, HMM can be defined as a tuple $\langle S,O,\pi,A,B,La\rangle$. We need to abstract S,O and La from the PLC system.

The element S is the set of all normal states of the PLC system. Each normal state of a PLC system is composed of all the actuators' states. Since actuators are actuated by the ladder program, each normal state of the PLC system can be identified by the primary outputs (coils) of the ladder rungs. When the number of states is small, we can list all the states easily. When the number of states is large, we utilize two analysis methods to construct the S of HMM. The first method is from the system state to the hidden state. We refer to the design document, designer, implementer and deployer of the system to get a system state diagram, and then map the state from the system state diagram to the state in HMM. The second method is more complicated. First, we determine the initial state of the embedded ladder program. Then, we apply a static analysis method to the ladder program and controlled PLC system to find all the possible transformations. With this conversion process, we can discover the valid states.

The element O contains the observations corresponding to the state set S. It can be abstracted from the basic functional requirements and the events corresponding to the physical outputs of the system. Each observation is linked to the corresponding states by a probabilistic function.

The element La is a set of atomic propositions $\{La\}$ labeling states and transitions. Since the PLC system works in a periodic scanning manner and there may be timer instructions in the embedded ladder diagram, we extend the label with the time attribute. The recursive descent syntax of the label can be defined as:

$$La\to KDE;K\to K\ 0|K\ 1|0|1;D\to\mathbf{N}^+.$$

The label is composed of three components. The first component K is the primary input sequence of the PLC ladder program. The sequence determines the outgoing transitions of each state. The second component D is the time attribute related to the timer instructions in the ladder program. It denotes that the corresponding state S_i transfers to the state S_j with a time delay D. The value of D is a positive integer $\mathbf{N}^+$. E is the set of observations that the primary input triggers in the dedicated state S_j.

3.3 Solving the Hidden Markov Model

We have obtained the necessary knowledge about the observations, states, and transitions of a PLC system. We now need to solve the unknown parameters of the HMM, especially the state transition probability matrix A. We provide two methods to solve the HMM based on three kinds of domain knowledge. If the domain knowledge comes from a domain expert and runtime monitoring, the problem can be addressed by the extended the Baum-Welch algorithm. If the domain knowledge is from the simulation, we can adopt a more direct method to find the transition probability matrix.

3.3.1 Extended Baum-Welch Method

The extended Baum-Welch algorithm is based on two kinds of domain knowledge. The first one relies on domain experts. For a particular PLC application, the domain expert is inquired to provide a set of observation sequences. The observation sequences represent the actual operating environment of the PLC system. The second kind of domain knowledge is

from runtime monitoring. After the PLC system is deployed and in use, we can observe the execution of the system, time after time, to attain the observation sequences.

With these two kinds of domain knowledge, we can get the observation sequence $O(O_1 \cdots O_t O_{t+1} \cdots O_T)$ from the time unit 1 to time unit T. Then, we need to adjust the HMM parameters to maximize the probabilities of the observation sequence. The Baum-Welch algorithm applies a dynamic programming technique to estimate the parameters. It is an iterative procedure, which starts from a possibly random model $M(A, B, \pi)$. It then applies the observation sequence for the iterative update, and improves the parameters of M to get the local maxima of the likelihood function. It makes use of a forward-backward procedure based on two variables:

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i | M),$$
$$\beta_t(i) = P(O_{t+1} O_{t+2} \cdots O_T | q_t = S_i, M),$$

where $\alpha_t(i)$ is the probability that the partial observation sequence $O_1 O_2 \cdots O_t$, and the system model are in the state S_i at time unit t. $\beta_t(i)$ is the probability that the partial observation sequence from the time unit $t+1$ to the end, given the state S_i at time unit t. The induction relation of the two variables are:

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(j) a_{ij}] b_j(O_{t+1}),$$
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j),$$

where $b_j(O_{t+1})$ denotes the probability that the system is in state j, and the observation is O_{t+1}. Then, we define some variables on the forward and backward variables to compute the parameter a_{ij}.

$$\gamma_t(i) = P(q_t = S_i | O, M) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}.$$

$\gamma_t(i)$ is the probability that the system is in state S_i at time t, given the observation sequence O.

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, M) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}.$$

$\xi_t(i,j)$ is the probability that the system is in state S_i at time t and in state S_j at time $t+1$, respectively, given the observation sequence O. The events leading to the appearance of $\xi_t(i,j)$ are illustrated in Fig. 6.

Therefore, the expected number of all the transitions starting from S_i can be defined as $\sum_{t=1}^{T-1} \gamma_t(i)$. Similarly, the

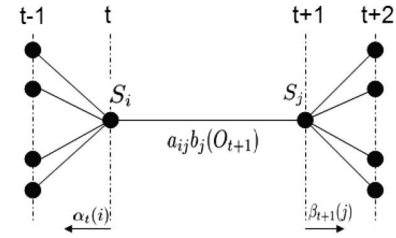


Fig. 6. The events leading to the appearance of $\xi_t(i,j)$.

expected number of the transitions from S_i to S_j can be defined as $\sum_{t=1}^{T-1} \xi_t(i,j)$. The expected number of occurrences of observation O_k in the state S_i can be defined as $\sum_{t=1, v_t=O_k}^{T} \gamma_t(i)$. For a single observation sequence, the iterative calculation formulas for the state transitions and the observation probabilities can be defined as:

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \quad \overline{b_{ij}} = \frac{\sum_{t=1, v_t=O_j}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)},$$

where the numerator of $\overline{a_{ij}}$ denotes the expected number of transitions from the state S_i to S_j, and the denominator denotes the expected number of transitions from the state S_i. The numerator of $\overline{b_{ij}}$ denotes the expected number of times observing the symbol O_j in the state S_i, and the denominator of it denotes the expected number of times when the system is in the state S_i.

We now consider the multiple observation sequences $Os(O^1, O^2 \cdots, O^s)$, where $O^k(O_1^k \cdots O_t^k O_{t+1}^k \cdots O_T^k)$ is the k_{th} observation sequence, and W_k is the frequency of sequence O^k. The parameters of model M need to be adjusted to maximize $P(Os|M)$. We extend the method presented in [27] with a weight W_k for each O^k. We define $P_k = P(O^k|M)$ and $P(Os|M) = \prod_{k=1}^{s} P_k W_k$. The iterative calculation formulas for the state transitions and the observation probabilities can be changed to:

$$\overline{a_{ij}} = \frac{\sum_{k=1}^{s} \frac{1}{P_k W_k} \sum_{t=1}^{T-1} \xi_t^k(i,j)}{\sum_{k=1}^{s} \frac{1}{P_k W_k} \sum_{t=1}^{T-1} \gamma_t^k(i)},$$
$$\overline{b_{ij}} = \frac{\sum_{k=1}^{s} \frac{1}{P_k W_k} \sum_{t=1, v_t=O_k}^{T} \gamma_t^k(i)}{\sum_{k=1}^{s} \frac{1}{P_k W_k} \sum_{t=1}^{T} \gamma_t^k(i)},$$

where the numerator and denominator of the extended formulas have the same meaning as the original ones.

We can now choose an initial model $M(A, B, \pi)$ and take M to compute the right side of the iterative calculation formulas. Once we get the new model $\overline{M}(\overline{A}, \overline{B}, \pi)$, we can use $\overline{M}$ to replace M. This procedure is repeated until the stop condition is met. For example, the probabilities of the observation sequence satisfies $|P(Os|\overline{M}) - P(Os|M)| < \theta$, where θ is the user-defined precision limit.

3.3.2 Simulation-Based Method

The second method is based on the domain knowledge from simulations of a PLC system. We take simulation techniques to get the state sequences and the frequencies of each state transition. Then we can calculate the state transition matrix directly. For example, we identify a state S_i with two outgoing transitions to S_j and S_k. We monitor the simulations for 100 system cycles. There are 100 transitions, and the frequencies of the transitions a_{ij} and a_{ik} are 80 and 20, respectively. As a result, the transition probabilities for the two transition are 0.8 and 0.2, respectively. This method decreases the inaccuracy of the HMM with more precise domain knowledge.

With the two methods, a solved HMM is built and denotes the real operating environment in a particular application. we can get the solved state transition matrix A, where a_{ij} is the probability that the system transfers from the state S_i to the state S_j.

3.4 The Construction of the Combined Regular Markov Model

The uncertainty characterization of a PLC system itself denotes the inner reliability of the system, which evaluates the effects due to the errors from the sampling of input devices and the errors from the program execution of the PLC microprocessor. The solved HMM shows the operating environment of the PLC system in a particular application, with the use of the normal states and transitions. We construct a new Markov model to combine these two properties.

The first step is to add the abnormal state caused by the uncertainty into the solved HMM. Since the PLC system would go into an abnormal state from any normal states, we build an abnormal state U for all the normal state S_n. When a normal state transmits to an abnormal state, it can be recovered to the initial state by the system itself, or by human intervention. We also need to add the related transitions to the solved HMM. Afterwards, we can get all the states and transitions of the new Markov model M'.

The second step is to initiate the transition matrix of the new model M'. We need to assign values to different transitions. The probability of a normal state transmitting to an abnormal state depends on the uncertainty characterization. The recovering transitions depend on the design of the PLC system. After these states and transitions are added to matrix A, the original value of a_{ij} needs to be adjusted with a coefficient. The new transition matrix A' is defined as:

$$\begin{pmatrix} a_{00}(1-f) & a_{01}(1-f) & \dots & a_{0n}(1-f) & f \\ a_{10}(1-f) & a_{11}(1-f) & \dots & a_{1n}(1-f) & f \\ \vdots & \vdots & \ddots & \vdots & \\ a_{n0}(1-f) & a_{n2}(1-f) & \dots & a_{nn}(1-f) & f \\ r_{U0} & 0 & \dots & 0 & 1-r_{U0} \end{pmatrix},$$

where the last row and the last column are for the abnormal state U, and the remaining rows and columns are for the normal states S_n. The probability of the normal states changing to the abnormal state is f, which is the value of uncertainty characterization. We know that the error probability is the same for all the normal states, since the uncertainty characterization f is the inner quality of the PLC system. The recovering transition probability from the abnormal state U to the initial state S_0 is r_{U0}. The system remains in the abnormal state with a probability of $1 - r_{U0}$ in the case of uncertainty that can not be recovered. The transition probability among the normal states is $a_{ij}(1-f)$. The result of a_{ij} multiplied by the coefficient $1-f$ is the new transition probability combined with the uncertainty characterization.

The constructed Markov model combines the inherent reliability property with the operating environment of the PLC system. It closely mimics the actual executions of the PLC system in real life applications. Based on the combined Markov model, we can now analyze several runtime properties of the PLC system by model-checking.

3.5 Domain Property Analysis with PRISM

After building the combined Markov model, we can perform probabilistic model checking using PRISM [28]. First, the Markov model needs to be specified with PRISM modeling language. Second, we need to add some rewards to the model, to specify additional quantitative measurements. Finally, we specify the properties to be analyzed on the PLC system.

A PRISM model comprises a set of modules that represent different aspects of a PLC system. The behavior of a PRISM model is specified by guarded commands. Synchronization among different modules can be implemented by augmenting guarded commands with the same action label. We now describe the combined Markov model $\langle S', \pi', A', La' \rangle$ in PRISM. The module is derived from the transition matrix A'. We declare a variable H, whose value ranges in $[0, \mathrm{n}+1]$. We then build a label command for each row of the matrix A' based on the variable $[L_i]H = i \rightarrow (a'_{i0} : H' = 0) + (a'_{i1} : H' = 1) + \dots (a'_{in} : H' = n) + (f : H' = n+1)$. Subsequently, we focus on extending the model with rewards. The structure of a reward is:

$$\textbf{rewards} \quad \text{“}reward\_name\text{”} \quad component \quad \textbf{endrewards}.$$

There are two types of rewards: (1) the $component$ for the state reward is $guard : reward$, and (2) the $component$ for the transition reward is $[Label]guard : reward$. $Label$ is the command label in each module, $guard$ is a predicate over state variables, and $reward$ is a real-value expression. When states and transitions satisfy the $guard$, $reward$ assigns quantitative measurements to them.

In a PLC system, the main performance measurements we are concerned with are the timing and reliability properties. Therefore, we define two representative rewards. The first is a transition reward $component$ regarding the time property. It can be derived from element D of the transition label La. We add a $component$ $\{[L_i]true : D\}$ to each transition into the reward. The reward reflects the elapsed time of each transition. The other reward is a state reward $component$ regarding the reliability property. We associate the number 1 to all the normal states with a $component$ $\{r : 1\}$. This reward can be used to determine the long-run availability of a PLC system. We can also define other kinds of rewards, such as the power consumption of each transition or state.

After the probabilistic model and rewards are described in PRISM, we can analyze some properties. The properties are specified in PRISM property specification language. We take the $\mathcal{P}$, $\mathcal{S}$, and $\mathcal{R}$ operators to specify quantitative time instant properties, and the long-run properties formula Φ. For

example, we use the following three properties to describe the execution state of a PLC system in the long-run (where U denotes the abnormal state).

- $\mathcal{S}_{=?}[!U]$, shows the probability that the PLC system is not in failure in the long-run.
- $\mathcal{P}_{=?}[G^{[0,t]}!U]$, represents the probability that the PLC system has no errors during t time units.
- $\mathcal{P}_{=?}[F^{[t,t]}!U]$, denotes the probability that the PLC system is not in failure at time instant t.

We can also specify the properties formula Φ regarding time as follows.

- $\mathcal{R}_{\text{“}r\text{”}=?}[C^{\leq t}]$, yields the cumulative time that the PLC system is in the normal states during t time units.
- $\mathcal{R}_{\text{“}D\text{”}=?}[FU]$, yields the cumulative time passing in the PLC system before the first failure.

4 Case Studies

In this section, we present our our application of the proposed framework to an actual industrial PLC system, which was originally published in [29]. The system is shown in Fig. 7. The input devices of this PLC system are the sensors. The actuators are the three pistons, (A, B, C) which are operated by solenoid valves (V_1, V_2, V_3). Each piston has two corresponding, normally open, contacts. Three push buttons are provided to start the system (switch SW_1), to stop the system normally (switch SW_2) and to stop the system immediately in an emergency (switch SW_3). In the manufacturing facility, such piston systems are used to load or unload parts from a machine table, and extend or retract a cutting tool spindle.

There are many PLC ladder programs that can be used to control this system. Referring to the example in Fig. 1, this ladder program is similar to the third ladder diagram in [29]. In order to facilitate our analysis and make the system easier to understand, we deleted the counter instruction that was in the original ladder diagram. There are four ladder rungs, which includes 8 primary input contacts $(SW_1, SW_3, a_0, a_1, b_0, b_1, c_0, c_1)$. SW_1 and SW_3 are changed by human operations, while the others are automatically changed by the movements of the pistons. We can construct an automata model for the operating principle of the PLC system. The state is denoted by the outputs of the ladder. The details of the system control theory are shown in Fig. 8. The number of ladder outputs is 4, and the number of all possible states is 2^4. Based on the description of the system above, we used the methods mentioned in 3.2 to discover six valid states and one abnormal state. Then, the S and O components and the transitions between the states of the HMM are derived. The system has six normal states $(S_0, S_1, S_2, S_3, S_4, S_5)$, and an abnormal state corresponding to many types of failures caused by the uncertainty characterization. The six normal states are the states of the HMM. In Fig. 8, each normal state has some corresponding observations linked by dotted lines. In this system, there are four states that have only one corresponding observation each, and state S_5 has three corresponding observations. The transitions are labeled with the primary input sequences. The time for each transition is one time unit, except that the transition between the state S_3 and S_4 is six time units. We present below the detailed control theory of the system model.

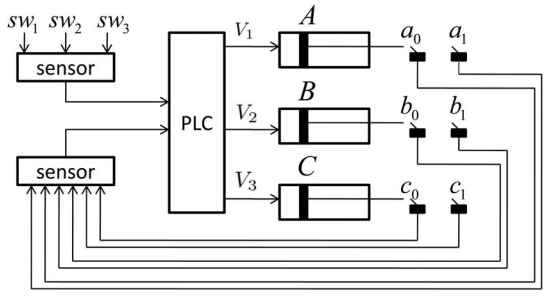


Fig. 7. The industrial automated system.

Initially, the system is in a blank state, S_0. In this state, the pistons remain on the left side. As a result, the values of $(a_0, a_1, b_0, b_1, c_0, c_1)$ are $(1, 0, 1, 0, 1, 0)$. In the first execution cycle of the PLC system, when the worker presses the start switch (SW_1), and the system is activated. The values of (V_1, V_2, V_3, T) are then $(1, 0, 0, 0)$. Piston A will move to the right side (A^+). The values of $(a_0, a_1, b_0, b_1, c_0, c_1)$ change to $(0, 1, 1, 0, 1, 0)$. In the second execution cycle, the values of (V_1, V_2, V_3, T) are $(1, 1, 0, 0)$. Piston B will move to the right side (B^+). The values of $(a_0, a_1, b_0, b_1, c_0, c_1)$ change to $(0, 1, 0, 1, 1, 0)$. In the third execution cycle, the values of (V_1, V_2, V_3, T) are $(0, 1, 1, 0)$. Piston A will move back to the left side and piston C will move to the right side, simultaneously (A^-C^+). The values of $(a_0, a_1, b_0, b_1, c_0, c_1)$ change to $(1, 0, 1, 0, 0, 1)$. If we press SW_3, pistons B and A will move to the left side (A^-B^-). In the fourth execution cycle, since the values of (c_1, a_0) are $(1, 1)$, the timer instruction is activated. In the next five time units, the value for the output (V_1, V_2, V_3, T) will not change. Therefore, the system will keep static for five time units. At the sixth time unit, the values of (V_1, V_2, V_3, T) are $(0, 0, 0, 1)$. Pistons B and C will then move to the left side (B^-C^-). The values of $(a_0, a_1, b_0, b_1, c_0, c_1)$ change to $(1, 0, 1, 0, 1, 0)$.

We can build an HMM for the PLC system using the operating principles presented in Fig. 8. The states of the HMM are the normal states in the automata. The transition label La, between two hidden states, is also derived from the automata, where element K is the eight primary inputs on the automata label and element D is the time for each state transition of the original automata. The set of observations is composed of the content inside the rectangle in Fig. 8. Hence, we can produce matrix A for the HMM as:

$$\mathbf{A} = \begin{pmatrix} & S_0 & S_1 & S_2 & S_3 & S_4 & S_5 \\ S_0 & a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} \\ S_1 & a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ S_2 & a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ S_3 & a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ S_4 & a_{40} & a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ S_5 & a_{50} & a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix}.$$

There is a real-life application, where the operating environment of this PLC system is representative of a movement sequence O: $[A^+, B^+, C^+A^-, B^-C^-]$. In this case, we need to solve matrix A and B using the Baum-Welch algorithm, or by using simulations to achieve the maximum $P(O|M)$.

Then, we need to combine the solved HMM with the uncertainty state caused by the uncertainty characterization. We assume that the sensor error for each primary input

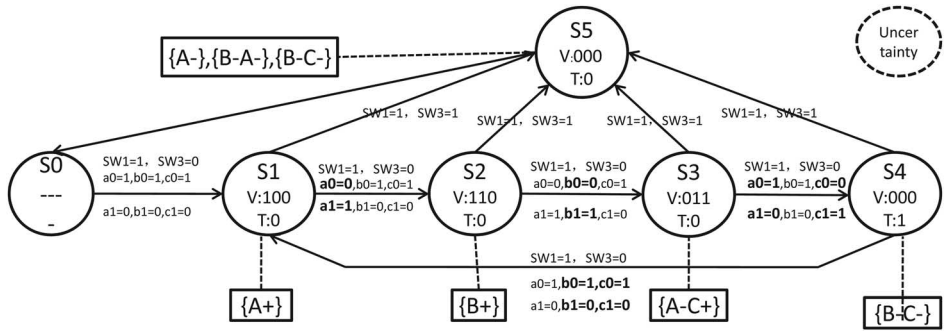


Fig. 8. The details of the system control theory.

TABLE 1
Property Results Get by the Symbolic Framework

property	env	U=0.01	U=0.03	U=0.05	U=0.07	U=0.09	U=0.1	U=0.15	U=0.2	U=0.25
time	1	198	98	38	32	27	21	18	8	6
time	2	233	73	43	30	23	21	13	9	7
availability	1	0.990	0.971	0.952	0.934	0.917	0.908	0.868	0.830	0.795
availability	2	0.989	0.968	0.947	0.928	0.909	0.900	0.857	0.818	0.783
power	1	1462	1391	1325	1261	1200	1171	1032	908	797
power	2	1234	1181	1130	1081	1034	1011	903	804	714

contact is 0.05, and the ladder logic execution error of the PLC microprocessor is 0.3 [30]. Applying the method presented in Section 3.1, the uncertainty characterization of this PLC system is 9.8 percent, meaning that the system has a 0.098 probability of going into an abnormal state from any normal state. Assuming that the system will recover to the initial state from the abnormal state with a 0.9 probability, the combined matrix A' is presented as follow:

$$\begin{pmatrix} & S_0 & S_1 & S_2 & S_3 & S_4 & S_5 & U \\ S_0 & 0 & 0.902 & 0 & 0 & 0 & 0 & 0.098 \\ S_1 & 0 & 0 & 0.902 & 0 & 0 & 0 & 0.098 \\ S_2 & 0 & 0 & 0 & 0.902 & 0 & 0 & 0.098 \\ S_3 & 0 & 0 & 0 & 0 & 0.902 & 0 & 0.098 \\ S_4 & 0 & 0.902 & 0 & 0 & 0 & 0 & 0.098 \\ S_5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ U & 0.9 & 0 & 0 & 0 & 0 & 0 & 0.1 \end{pmatrix}.$$

Let us see an example of multiple observation sequences, whose operating environment is representative of three kinds of observation sequences. The observation sequences O^1, O^2, and O^3 are $[A^+, A^-], [A^+, B^+, B^-A^-]$ and $[A^+, B^+, A^-C^+, B^-C^-]$, respectively. We assume that during 1000 observations, O^1 appears 200 times, O^2 appears 200 times, and O^3 appears 600 times. We then apply the extended Baum-Welch algorithm for the multiple observation sequences, and the combined matrix A' is obtained as follows:

$$\begin{pmatrix} 0 & 0.902 & 0 & 0 & 0 & 0 & 0.098 \\ 0 & 0 & 0.722 & 0 & 0 & 0.180 & 0.098 \\ 0 & 0 & 0 & 0.677 & 0 & 0.225 & 0.098 \\ 0 & 0 & 0 & 0 & 0.902 & 0 & 0.098 \\ 0 & 0.902 & 0 & 0 & 0 & 0 & 0.098 \\ 0.902 & 0 & 0 & 0 & 0 & 0 & 0.098 \\ 0.9 & 0 & 0 & 0 & 0 & 0 & 0.1 \end{pmatrix}.$$

We can also take the system state diagram of the fourth ladder in [29] to construct a more complex HMM that contains 17 normal states and 26 transitions. When we apply the four observation sequences $[A^+, B^+, B^-A^-]$, $[A^+, B^+, A^-C^+, B^-C^-]$, $[A^+, B^+, A^-C^+, 6s, B^-C^-]$, $[3[A^+, B^+, A^-C^+, 6s, B^-C^-], 10s, A^+, B^-A^-]$ to solve the HMM, we can obtain a more complicated matrix, which is too big to present here.

As described in Section 3.5, we now describe these models in PRISM, and extend these models with rewards to analyze the system. The time property is derived from the time element D of the transition label La. In addition, we assign a reward $power$ to each state. The reward denotes the power consumption for valid piston movements in each state. Then, we can initiate some performance properties based on the rewards $time$ and $power$.

- $S_{=?}[H < 6]$: the long term availability of the system.
- $R_{\{\text{“}time\text{”}\}=?}[FH = 6]$: the first time of a system failure.
- $R_{\{\text{“}power\text{”}\}=?}[C^{\leq 1000}]$: the valid power consumption of piston movements during 1000 time units.

In the following paragraphs, we will demonstrate how the operating environment affects the performance measurements of a PLC system. Three operating environment examples were described above, which are denoted by three sets of observation sequences. We compare the three properties in the first two operating environments with PRISM. The results are shown in Table 1. All the results are generated by PRISM within 0.1 second. We can see from Table 1 that the long term availability for the second example ($\mathrm{env} = 2$) is always better than for the first example ($\mathrm{env} = 1$). The first failure time for the second example comes faster than for the first example. The total number of valid piston movements for the first operating environment is greater than for the second operating environment. More results are shown in Figs. 9-11.

This case study demonstrates that the performance properties of the system are different in different operating

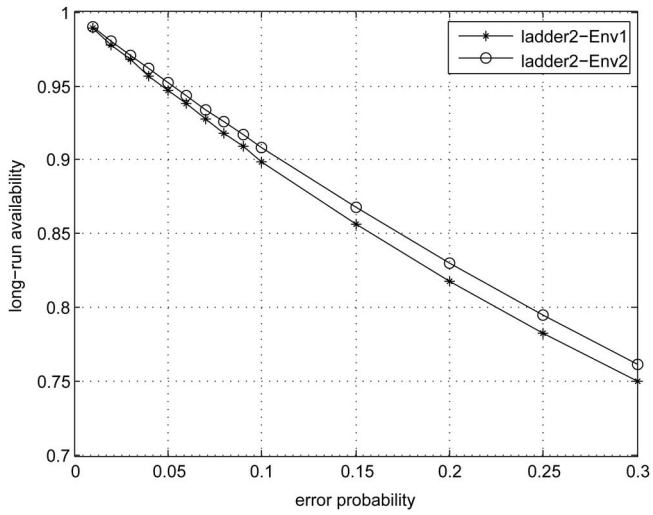


Fig. 9. Long-term availability. The horizontal axis is for logic execution error probabilities of the PLC microprocessor. The vertical axis is for the long-term availability of the system in two operating environments.

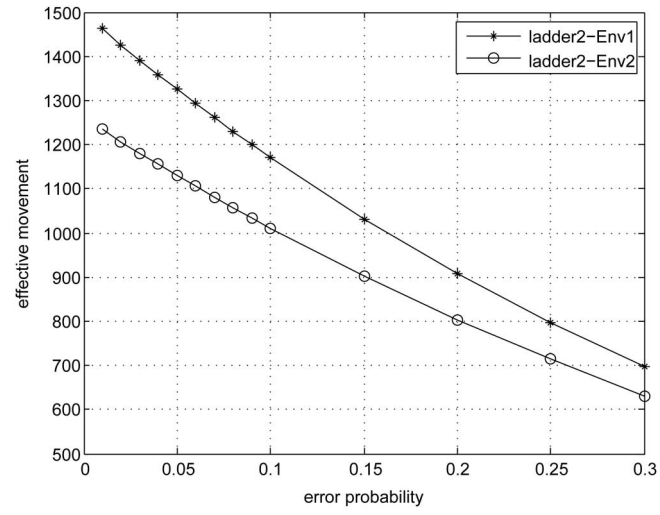


Fig. 11. Total movements. The horizontal axis is for logic execution error probabilities of the PLC microprocessor. The vertical axis is for the valid number of piston movements in two operating environments during 1000 time units.

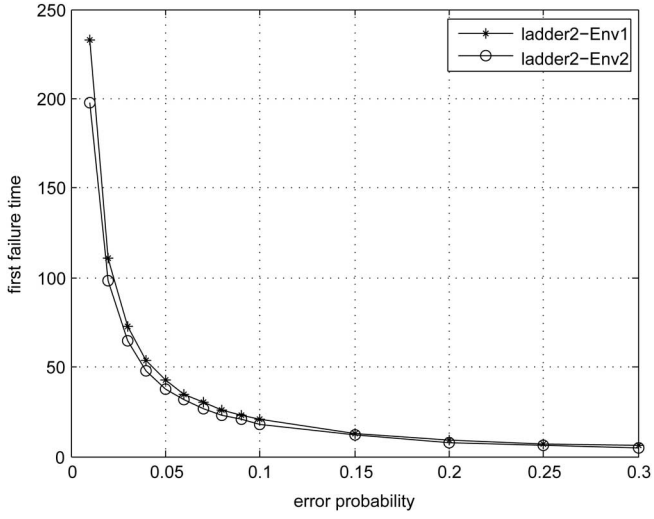


Fig. 10. Time. The horizontal axis is for logic execution error probabilities of the PLC microprocessor. The vertical axis is for the first failure time of the system in two operating environments.

environments, even for the same PLC systems controlled by the same ladder program. We also performed additional experiments and complexity analysis to demonstrate the scalability of our proposed method, particularly for uncertainty calculations and the HMM constructions and solutions.

There are four ladder programs presented in [29]. Different ladder programs lead to different arrangements of primary inputs and logic executions of the PLC microprocessor. At the same time, we also constructed several large, random PLC ladder programs to test the uncertainty calculation methods. The examples of Random1, 2 and 3 consist of 200 400, and 600 normally open/closed contacts, respectively. We set the sensor sampling error probability to 0.05 and changed the logic execution error probability of the PLC microprocessor (ε). Applying the method presented in Section 3.1, we obtained the uncertainty characterization of the PLC system controlled by the four ladder programs, shown in Table 2, and the three constructed random programs shown in Table 4. We have also devised simulations to verify the accuracy of the proposed methods. A simulator was implemented for a Monte Carlo framework of uncertainty analysis based on fault injection. The simulations were run on a 3.06 GHz windows based system, with 2 GB of memory. The values acquired by the random simulations are presented in Tables 2 and 4. The uncertainty characterization obtained by the original component-based Bayesian Network, and the Fault Tree method are presented in Table 3. The results are the same for each failure probability of the processor. The main focus is on the distribution of the system components and the signal dependencies amongst them, while the complex relationships caused by the execution logic of the program are ignored. The values are not similar to the simulation results. We used one million random patterns for the simulations.

These additional experiments demonstrate that (1) the uncertainty characterizations of the PLC system are different when controlled by different embedded ladder programs, even with the same sensor error probability and the same logic execution error probability of the microprocessor, (2) the results acquired by our method are nearer to the run time station, and the difference between the results of the simulations and our method is within 0.5 percent.

With the case study and the additional experiments, the main procedures of our framework have been demonstrated including the uncertainty calculations, the HMM constructions and solutions, and the reward based model checking. During the uncertainty calculation procedure, the two points of importance are the ladder translation and the construction of an abstract syntax tree, both with the complexity $O(N)$, where N is the number of the primary input contact. Subsequently, the error probability was processed from the bottom up through the abstract syntax tree according to theorem 1 . As presented in the experiments above, a large size problem may be processed with little loss of accuracy. During the HMM constructions and solutions procedures, we constructed an HMM for the PLC system. As described in section 3.2, this would not take a lot of effort when the system state diagram is already known. The complexity of the Baum-Welch algorithm, when

TABLE 2
Uncertainty Characterization of the Four Exited Ladder Programs by Simulation and Calculation

method	ladder	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.15$	$\varepsilon = 0.2$	$\varepsilon = 0.25$	$\varepsilon = 0.3$
simulation	ladder1	0.69%	17.02%	13.82%	12.66%	10.29%	8.75%	5.54%
simulation	ladder2	0.74%	17.82%	14.05%	13.35%	11.91%	10.63%	6.67%
simulation	ladder3	1.12%	28.22%	25.51%	24.71%	22.14%	21.27%	9.89%
simulation	ladder4	1.21%	30.37%	26.68%	25.98%	24.78%	22.95%	11.16%
calculation	ladder1	0.67%	16.87%	13.67%	12.54%	10.23%	8.69%	5.51%
calculation	ladder2	0.70%	17.63%	13.89%	13.21%	11.82%	10.57%	6.62%
calculation	ladder3	1.08%	28.04%	25.36%	24.57%	22.03%	21.18%	9.80%
calculation	ladder4	1.15%	30.13%	26.51%	25.82%	24.65%	22.83%	10.94%

TABLE 3
Uncertainty Characterization of the Four Exited Ladder Programs by Bayesian Networkand Fault Tree

method	ladder	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.15$	$\varepsilon = 0.2$	$\varepsilon = 0.25$	$\varepsilon = 0.3$
BN	ladder 1, 2, 3, 4	0.46%	2.10%	9.24%	12.96%	13.32%	15.75%	16.14%
FT	ladder 1, 2, 3, 4	0.46%	2.10%	9.24%	12.96%	13.32%	15.75%	16.14%

TABLE 4
Uncertainty Characterization of the Three Constructed Ladder Programs by Calculation and Simulation

method	ladder	contacts	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.15$	$\varepsilon = 0.2$	$\varepsilon = 0.25$	$\varepsilon = 0.3$
calculation	Random1	200	1.16%	31.21%	29.15%	20.02%	15.31%	10.45%	5.51%
calculation	Random2	400	8.21%	33.12%	24.71%	16.17%	13.64%	8.31%	7.36%
calculation	Random3	600	16.51%	38.41%	30.24%	24.57%	14.18%	10.65%	8.11%
simulation	Random1	200	1.14%	32.02%	29.66%	20.30%	15.42%	10.59%	5.58%
simulation	Random2	400	8.11%	33.91%	25.13%	16.41%	13.89%	8.47%	7.41%
simulation	Random3	600	16.71%	39.14%	30.81%	24.98%	14.40%	10.82%	8.17%

transitions are labeled, is determined to be $O(A^2 \cdot B \cdot O)$, where A is the number of states, B is the number of events, and O is the length of the training observation sequences [31]. Afterwards, we can solve the HMM using the extended Baum-Welch method, and perform some performance measurements using rewards based model checking with PRISM. Since the model has been reduced in the HMM construction process, PRISM will not suffer from the state space explosion problem. The overall time complexity for model checking a property formula Φ against an HMM is linear in $|\Phi|$ and polynomial in $|\mathrm{A}|$ [32]. The size of $|\Phi|$ equals the number of logical connectives and temporal operators in the formulas plus the sum of the sizes of the temporal operators. Based on the the case study, additional experiments, and complexity analysis, we can draw the conclusion that our framework can be applied to the analysis of complicated PLC systems.

5 Conclusion

This paper presents a symbolic framework for the formal analysis of PLC systems. The framework allows us to obtain the reliability calculations and the performance measurements of PLC systems by automated analysis. The reliability calculation is carried out through the use of a novel probabilistic method. An abstract syntax tree is constructed to capture the execution logic of the embedded control program, and the uncertainty hardware errors are embedded into the corresponding nodes of the tree. To the best of our knowledge, it is the first time that we have modeled both the execution logic of the embedded control program and the hardware components. The performance measurements are implemented by reward based model checking. We abstracted the PLC system as a Hidden Markov Model, and extended the Baum-Welch method to obtain a solution based on the domain knowledge of a dedicated operating environment. Then, we combined the solved Hidden Markov Model and the reliability characterization, and defined some properties and rewards. Finally, the performance measurements can be obtained with the help of some probabilistic model checking tools like PRISM. Our future efforts will focus on the automatic techniques that transform an PLC system into an HMM, and facilitate more accurate calculation of the uncertainty characterization of PLC systems.

Acknowledgments

This research was supported in part by NSFC Programs (61202010, 91218302) and 973 Program (2010CB328003) of China.

References

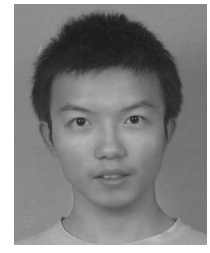
[1] G. Canet, S. Couffin, J.-J. Lesage, A. Petit, and P. Schnoebelen, "Towards the Automatic Verification of PLC Programs Written in Instruction List," *Proc. IEEE Conf. Systems, Man, and Cybernetics*, pp. 2449-2454, Oct. 2000.

[2] H.-M. Hanisch, J. Thieme, A. Luder, and O. Wienhold, "Modeling of PLC Behaviour by Means of Timed net Condition Event Systems," *IEEE Int'l Symp. Emerging Technologies and Factory Automation (EFTA)*, pp. 361-369, 1997.

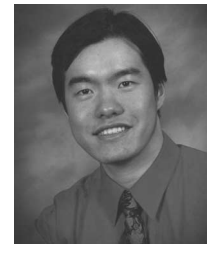
[3] K. Loeis, M. Younis, and G. Frey, "Application of Symbolic and Bounded Model Checking to the Verification of Logic Control Systems," *Proc. 10th IEEE Conf. Emerging Technologies and Factory Automation*, vol. 1, pp. 4-16, Sept. 2005.

[4] O. Pavlovic, R. Pinger, and M. Kollmann, "Automated Formal Verification of PLC Programs Written in IL," *Proc. Conf. Automated Deduction (CADE)*, pp. 152-163, 2007.
[5] T.L. Johnson, "Improving Automation Software Dependability: A Role for Formal Methods?" *Control Eng. Practice*, vol. 15, no. 11, pp. 1403-1415, 2007.
[6] W. Lee, D. Grosh, and F. Tillman, "Fault Tree Analysis, Methods, and Applications—A Review." *IEEE Trans. Reliability*, vol. R-34, no. 3, pp. 194-203, Aug. 1985.
[7] M. Shooman, *Reliability of Computer Systems and Networks*. Wiley Online Library, 2002.
[8] X. Zang, H. Sun, and K. Trivedi, "A BDD-Based Algorithm for Reliability Evaluation of Phased Mission Systems," *IEEE Trans. Reliability*, vol. 48, no. 1, pp. 50-60, 1999.
[9] M. Bouissou and J. Bon, "A New Formalism that Combines Advantages of Fault-Trees and Markov Models: Boolean Logic Driven Markov Processes," *Reliability Eng. & System Safety*, vol. 82, no. 2, pp. 149-163, 2003.
[10] D. Wooff, M. Goldstein, and F. Coolen, "Bayesian Graphical Models for Software Testing," *IEEE Trans. Software Eng.*, vol. 28, no. 5, pp. 510-525, May 2002.
[11] C. Bai, Q. Hu, M. Xie, and S. Ng, "Software Failure Prediction Based on a Markov Bayesian Network Model," *J. Systems and Software*, vol. 74, no. 3, pp. 275-282, 2005.
[12] S. Bhanja and N. Ranganathan, "Switching Activity Estimation of VLSI Circuits using Bayesian Networks," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 4, pp. 558-567, Aug. 2003.
[13] S. Bhanja, K. Lingasubramanian, and N. Ranganathan, "A Stimulus-Free Graphical Probabilistic Switching Model for Sequential Circuits using Dynamic Bayesian Networks," *ACM Trans. Design Automation of Electronic Systems (TODAES)*, vol. 11, no. 3, pp. 773-796, 2006.
[14] S. Krishnaswamy, G. Viamontes, I. Markov, and J. Hayes, "Probabilistic Transfer Matrices in Symbolic Reliability Analysis of Logic Circuits," *ACM Trans. Design Automation of Electronic Systems (TODAES)*, vol. 13, no 1, pp 1-35, 2008.
[15] C.-C. Yu and J.P. Hayes, "Scalable and Accurate Estimation of Probabilistic Behavior in Sequential Circuits," *Proc. 28th VLSI Test Symp.*, pp. 165-170, 2010.
[16] G. Norman, D. Parker, M. Kwiatkowska, and S. Shukla, "Evaluating the reliability of nand multiplexing with prism," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 10, pp. 1629-1637, Oct. 2005.
[17] D. Bhaduri and S. Shukla, "Nanoprism: A Tool for Evaluating Granularity vs. Reliability Trade-Offs in Nano Architectures," *Proc. 14th ACM Great Lakes Symp. VLSI*, pp. 109-112, 2004.
[18] *IEC 61131-3 Standard (PLC Programming Languages)*, 2nd ed., Int'l Electrotechnical Commission (IEC), 2003.
[19] L. Rabiner and B. Juang, "An Introduction to Hidden Markov Models," *IEEE Acoustics, Speech, and Signal Processing Magazine*, vol. 3, no. 1, pp. 4-16, Jan. 1986.
[20] A. Poritz, "Hidden Markov Models: A Guided Tour," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP'88)*, pp. 7-13.
[21] X. Song, "Algebraic Characteristics of Reversible Gates," *Theory of Computing Systems (Math. Systems Theory)*, vol. 39, no. 2, pp. 311-319, 2006.
[22] A. Dempster et al., "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc., Series B (Methodological)*, vol. 39, no. 1, pp. 1-38, 1977.
[23] J. Bilmes, "A Gentle Tutorial of the EM Algorithm and Its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models," *Int'l Computer Science Inst.*, vol. 4, p. 126, 1998.
[24] X. Song and Y. Wang, "On the Crossing Distribution Problem," *ACM Trans. Design Automation of Electronic Systems (TODAES)*, vol. 4, no. 1, pp. 39-51, 1999.
[25] S. Levinson, L. Rabiner, and M. Sondhi, " An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition," *The Bell System Technical J.*, vol. 62, no. 4, pp. 1035-1074, 1983.
[26] H. Zhang, Y. Jiang, W. Hung, G. Yang, M. Gu, and J. Sun, "New Strategies for Reliability Analysis of Programmable Logic Controllers," *Math. and Computer Modelling*, vol. 55, no. 7-8, pp. 1916-1931, 2011.
[27] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257-286, Feb. 1989.
[28] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: A Tool for Automatic Verification of Probabilistic Systems," *Proc. 12th Int'l Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, H. Hermanns and J. Palsberg, eds., pp. 441-444, 2006.
[29] K. Venkatesh, M. Zhou, and R.J. Caudill, "Comparing Ladder Logic Diagrams and Petri Nets for Sequence Controller Design through a Discrete Manufacturing System," *IEEE Trans. Industrial Electronics*, vol. 41, no. 6, pp. 611-619, Dec. 1994.
[30] G. Dunning, *Introduction to Programmable Logic Controllers*. Delmar Thomson Learning, 2002.
[31] L. Baum, "An Equality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes," *Inequalities*, vol. 3, pp. 1-8, 1972.
[32] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic Model Checking," *Proc. 7th Int'l Conf. Formal Methods for Performance Evaluation*, pp. 220-270, Oct. 2007.

Hehua Zhang received the BS and MS degrees in computer science from Jilin University, Changchun, China in 2001 and 2004, respectively. She received the PhD degree in computer science from Tsinghua University, Beijing, China in 2010. She is currently a lecturer in the School of Software at Tsinghua University. Her current research interests include domain specific modeling, formal verification, and their applications in embedded systems.



Yu Jiang received the BS degree in software engineering from Beijing University of Post and Telecommunication, Beijing, China in 2010. He is currently pursuing the PhD degree in computer science from Tsinghua University, Beijing, China. His current research interests include domain specific modeling, formal verification, and their applications in embedded systems.



William N.N. Hung received the BS and MS degrees in electrical and computer engineering from the University of Texas at Austin in 1994 and 1997, respectively. He received the PhD degree in electrical and computer engineering from Portland State University, Oregon in 2002. From 1997 to 2004, he worked as a senior engineer at Intel Hillsboro, Oregon. From 2004 to 2007, he worked as a senior staff engineer/director at Synplicity, Sunnyvale, California. Since November 2007, he has been working as a senior staff R&D engineer/senior R&D manager at Synopsys, Mountain View, California. His research interests include constraint solving, logic synthesis, physical design, formal methods, combinatorial optimization, nanotechnology, and quantum computing. He has published over 60 papers and has 5 patents.

Xiaoyu Song received the PhD degree from the University of Pisa, Italy in 1991. From 1992 to 1999, he was on the faculty at the University of Montreal, Canada. In 1998, he worked as a senior technical staff on Cadence, San Jose. In 1999, he joined the faculty at Portland State University, Oregon. He is currently a professor with the Department of Electrical and Computer Engineering, Portland State University. His current research interests include formal methods, design automation, embedded system design, and emerging technologies. During 2000-2005, he was named an the Intel Faculty Fellow. He served as an associate editor of the *IEEE Transactions on Circuits and Systems* and the *IEEE Transactions on VLSI Systems*.

Ming Gu received the BS degree in computer science from the National University of Defence Technology, Changsha, China in 1984, and the MS degree in computer science from the Chinese Academy of Science at Shengyang in 1986. Since 1993, she has been working as a lecturer/associate professor/researcher at Tsinghua University, Beijing, China. She is also serving as the vice dean of the School of Software, Tsinghua University. Her research interests include formal methods, middleware technology, and distributed applications.

Jiaguang Sun received the BS degree in automation science from Tsinghua University, Beijing, China in 1970. He is currently a professor at Tsinghua University. He is dedicated to teaching and R&D activities in computer graphics, computer-aided design, formal verification of software, and system architecture. He is currently the director of the School of Information Science and Technology and the School of Software, Tsinghua University. He is also the director of the National Laboratory for Information Science and Technology. He has been a member of the Chinese Academy of Engineering since 1999.