Bayesian-Network-Based Reliability Analysis of PLC Systems

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

***Abstract*—Reliability analysis is an important part of safety critical programmable logic controller (PLC) systems. The complexity of PLC system reliability analysis arises in handling the complex relations among the hardware components and the embedded software. Different embedded software types will lead to different arrangements of hardware executions and different system reliability quantities. In this paper, we propose a novel probabilistic model, called the hybrid relation model (HRM), for the reliability analysis of PLC systems. Its construction is based upon the execution logic of the embedded software and the distribution of the hardware components. We prove the constructed HRM to be a Bayesian network (BN) that captures the execution logic of the embedded software. Then, we map the hardware components to the corresponding HRM nodes and embed the failure probabilities of the hardware components into the well-defined conditional probability distribution tables of the HRM nodes. With the computational mechanism of the BN, the HRM handles the failure probabilities of the hardware components as well as the complex relations caused by the execution logic of the embedded software. Experiment results demonstrate the accuracy of our model.**

***Index Terms*—Bayesian network (BN), hybrid relation model (HRM), programmable logic controller (PLC), reliability analysis.**

I. Introduction

RELIABILITY analysis has become an important part of the system life cycle. This is particularly true for systems performing critical applications such as controlling nuclear power plants and spaceport devices [1], [2]. The reliability of a system is defined as the probability that the system will perform its intended function for a specified time period when operating under stated environmental conditions [3], [4]. The reliability theory deals with the interdisciplinary use of probability, statistics, and stochastic modeling, combined with engineering insights into the scientific understanding of failure mechanisms. The typical task for the reliability analysis is to build a mathematical reliability model representing the system through a set of random variables. Then, distributions of these variables are fully specified to calculate a system-level reliability for the output parameters [5]–[8]. For programmable logic controller (PLC) systems, these random variables are the failure probabilities of the hardware components and of software executions caused by the environment, design weakness, and mishandling.

Traditionally, the reliability analysis of PLC systems is usually realized by combinatorial methods such as fault tree (FT) [9] and reliability block diagram (RBD) [10]. FT 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 using techniques such as binary decision diagrams [11]. Analysts extend the traditional FT by associating a particular Markov process to the leaf node to improve the modeling power [12]. RBD is a graphical depiction of the system components and connectors. It can be used to determine the overall system reliability, when the reliability of each system component is given. Similar work for extending the traditional RBD with the Markov process is presented in [13]. Recently, a more flexible modeling framework, named the Bayesian network (BN), has been applied in system reliability [14]. It is similar to the neural network [15], [16] and is based on the graphical and probabilistic reasoning theory for handling uncertain probabilistic events. System reliability can be expressed as a joint probability function over some random variables and mapped onto a BN. If the qualitative part of the BN follows the logic connection of the system components, the causal dependences of the system are then thought to be understood. Some comparisons between BN and FT in terms of the modeling and analysis capabilities are presented in [17]. Dynamic BN is a generalization of BN and is also used for the reliability analysis [18]–[20]. The generalization is introduced to deal with the temporal correlation between time slices of the system. Those methods are efficient and easy to use and are the most widely used models for reliability analysis.

However, those methods present the designers and analysts with a high-level abstraction of PLC systems, demonstrating the distributions of the system components and events. They do not consider the complex relations among system components caused by the embedded software. For example, different types of embedded software have different times of memory reading and processor writing, which will lead to different overall

Manuscript received January 7, 2012; revised March 29, 2012 and June 29, 2012; accepted October 9, 2012. Date of publication October 18, 2012; date of current version June 6, 2013. This work was supported in part by the National Natural Science Foundation of China Programs under Grants 61202010 and 91218302, in part by the 973 Program of China under Grant 2010CB328003, and in part by the National Key Technologies R&D Program (SQ2012BAJY4052).

Y. Jiang is with the Department of Computer Science and Technology, School of Software, Tsinghua National Laboratory for Information Science and Technology, Key Laboratory for Information System Security, Ministry of Education, China (e-mail: jiangyu198964@gmail.com).

H. Zhang, M. Gu, and J. Sun are with the School of Software, Tsinghua National Laboratory for Information Science and Technology, Key Laboratory for Information System Security, Ministry of Education, China (e-mail: zhangheha@gmail.com; guming@126.com; sunjiaguang@126.com).

X. Song is with the Department of Electrical and Computer Engineering, Portland State University, Portland, OR 97201 USA (e-mail: song@ece.pdx.edu).

X. Jiao is with the International School, Beijing University of Posts and Telecommunications, Beijing 102209, China (e-mail: canbyjiaoxun@163.com).

W. N. N. Hung is with the Synopsys, Inc., Mountain View, CA 94043 USA (e-mail: william_hung@alumni.utexas.net).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2012.2225393

reliability quantities. For those reasons, we propose a novel probabilistic model, called the hybrid relation model (HRM), to handle the distributions of the system components as well as the complex relations caused by the embedded software. The HRM construction is based on the translation of the embedded software. We prove the constructed HRM to be a BN that captures the execution logic of the embedded software. Each node of the HRM is mapped to a corresponding system component, and the failure probability of the mapped component is used to initiate the conditional probability distribution (CPD) table of the node. Then, the HRM handles the failure probability of each system component and the execution logic of the embedded software. In addition, the HRM supports both predictive and diagnostic inferences about the reliability properties of the PLC system with the inference algorithms used in BN.

Generally speaking, the main contributions of our work are as follows.

1) We present an approach to translate the embedded control software of PLC systems into an HRM and prove that the constructed HRM captures the execution logic of the embedded software.
2) We define some CPD tables for the HRM nodes, map the system components to the HRM nodes, and initialize the CPD tables of the nodes with the failure probabilities of the corresponding components.
3) It is the first time that a model including both the execution logic of the embedded software and the hardware components in an automatic method can carry on reliability analysis.

This paper is organized as follows: Section II introduces some background on PLC systems and the BN, Section III presents the proposed HRM construction and initialization, Section IV presents the computation mechanism for the HRM and the reliability characterization of the whole system, Section V presents the experiments on some real industry PLC systems to illustrate the efficiency and validity of our work, and Section VII concludes this paper.

II. Background

A. PLC System

A simple PLC system is composed of a microprocessor, some input devices, and some output actuators. Signals can be received from input devices such as sensors and switches, processed by the microprocessor according to the arrangement of the embedded software, and sent to actuators [21], [22]. It is essentially an industrial control computer, which works by way of a periodic scanning mechanism. Each cycle is composed of three stages, as shown in Fig. 1. The first stage is sampling: The system reads the input data into the corresponding I/O. The second stage is processing: The microprocessor applies the embedded software to the control circuit and refreshes the state of the I/O image. The last stage is actuating: The microprocessor refreshes the output according to the state of the I/O image and actuates the peripherals via the output circuit.

The embedded software is used to control and combine the signals of these system components together. The International Electrotechnical Commission has defined four standard programming languages for PLC [23]: ladder diagram (LD), instruction list (IL), functional block diagram (FBD), and structured text (ST). LD has its roots in the U.S. It is based on the graphical presentation of relay ladder logic. IL is the European counterpart of LD. As a textual language, it resembles assembler. FBD expresses the behavior of a controller as a set of interconnected graphical blocks, like in the electronic circuit diagrams. ST is a very powerful high-level language that is close to Pascal. It is claimed that the four languages are equivalent [23]. There are many existing works translating among the four languages [24]–[26].

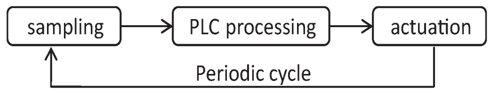


Fig. 1. Three stages for PLC system cycle.

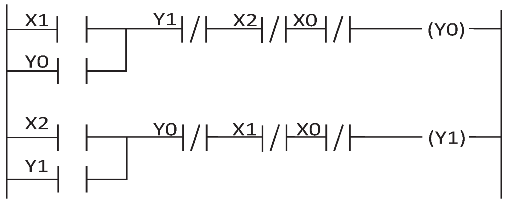


Fig. 2. Simple LD for the motor reversible control system.

In this paper, we focus on LD. Over the past 30 years, LD has been modified in order to keep up with increasing demands of the industry. The LD program appears in many PLC systems currently on the market. The LD program can be subdivided into networks and rungs. The basic principle of the ladder logic is the current flowing through the networks and rungs, from left to right and top to bottom. There are two types of basic instructions in LD. The first type consists of the regular instructions representing the conditions of the ladder rung. It is composed of contacts and logic connections. The second type consists of some special instructions such as timer, counter, and coil. If a path can be traced through the asserted contacts, the rung is true, and the output storage bit will be asserted true. Fig. 2 shows a simple LD program. It is made up of two ladder rungs. The symbol $-||-$ is a normally open contact, representing a primary input. When the value of X_1 is 1, the contact stays in the closed state, and the current flows through the trace. The symbol $-|/|-$ is a normally closed contact. When the value of Y_1 is 0, the contact stays in the closed state, and the current flows through the trace. For more detailed information about the other instructions, check [23].

B. BN

BN [27] is a directed probabilistic graphical model. Each node in the graph represents a random variable, and the arc between two nodes expresses the conditional probabilistic dependence of the two random variables. The formal definition of the BN is the tuple $\langle N, E, P\rangle$.

1) N is the set of nodes: $N = \{n_1, n_2, \ldots, n_n\}$, and n_i is the label of the node.
2) E is the set of arcs: $E = \{e_{ij}|\text{there is an arc from node } n_i \text{ to } n_j\}$, and e_{ij} is the label of the arc.

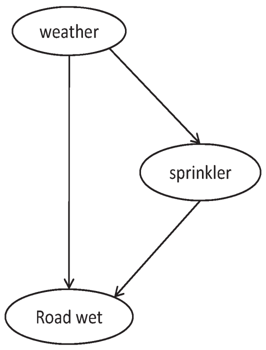


Fig. 3. Simple BN example for road state.

TABLE I
CPD FOR THE STATE OF ROAD: WET OR DRY

weather	sprinkler	road wet = 1	road wet = 0
0	0	0.0	1.0
0	1	0.7	0.3
1	0	0.9	0.1
1	1	1.0	0.0

3) P is the set of CPDs: $P = \{f(n_i|\text{parent}(n_i))\}$. $\text{parent}(n_i)$ denotes the parents of the node n_i, and $f(n_i|\text{parent}(n_i))$ is the conditional distribution of variable n_i given all its parents.

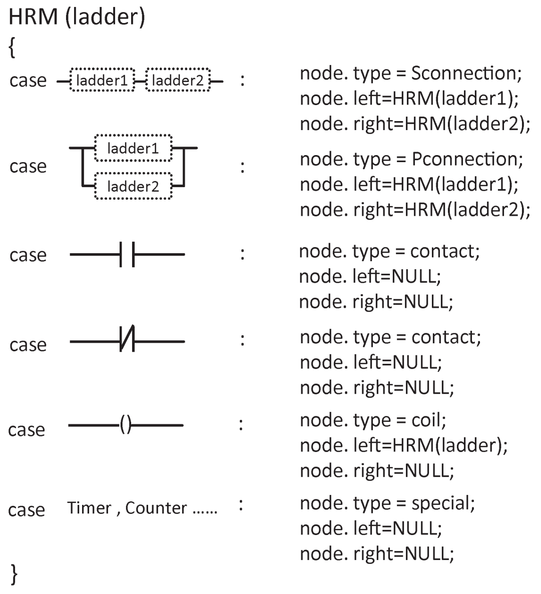
The first two items make up a directed acyclic graph (DAG), which is the qualitative part of the BN. The qualitative part is used to encode the conditional independence statements of a multivariate statistical distribution, representing that a variable is conditionally independent of its nondescendants given its parents. All of these conditional independences can be extracted from a BN structure using the *d-separation* rules.

The third item is the quantitative part of the BN. The conditional independences embedded among the random variables are described through the CPD tables. The CPD tables allow us to calculate the joint probability function in a simplified form (2) compared to the original form (1), where $f(n_1, n_2, \ldots, n_n)$ is the joint distribution of those variables and $f(n_n|n_{n-1} \cdots n_1)$ is the distribution of n_n given all the other variables

$$f(n_1, n_2, \ldots, n_n) = f(n_n|n_{n-1} \cdots n_1) \cdot f(n_{n-1}|n_{n-2} \cdots n_1) \cdots f(n_1) \quad (1)$$

$$f(n_1, n_2, \ldots, n_n) = \prod_{i=1}^{n} f\left(n_i|\text{parent}(n_i)\right). \quad (2)$$

A simple BN example is shown in Fig. 3. It captures the relations among the weather, sprinkler, and road. The weather can be sunny or rainy, the road can be wet or dry, and the sprinkler can be on or off. There are some causal links among the three nodes. If it is rainy, it will make the road wet directly. If it is sunny for a long time, we can make the road wet indirectly by turning on the sprinkler. Those causal correlations among the three objects are expressed with the directed arcs and the CPD table. The first line of the table represents the fact: When the weather is sunny and the sprinkler is closed, the road is wet with the probability of zero (Table I).



```
HRM (ladder)
{
case  ladder1 ladder2     :  node. type = Sconnection;
                             node. left=HRM(ladder1);
                             node. right=HRM(ladder2);
case  ladder1 / ladder2   :  node. type = Pconnection;
                             node. left=HRM(ladder1);
                             node. right=HRM(ladder2);
case                      :  node. type = contact;
                             node. left=NULL;
                             node. right=NULL;
case                      :  node. type = contact;
                             node. left=NULL;
                             node. right=NULL;
case                      :  node. type = coil;
                             node. left=HRM(ladder);
                             node. right=NULL;
case  Timer , Counter ……  :  node. type = special;
                             node. left=NULL;
                             node. right=NULL;
}
```

Fig. 4. Algorithm to translate the LD into a graph structure.

Based on the BN graphical structure and some CPD tables that reflect the reality of the three objects, both predictive and diagnostic inferences can be performed.

III. HRM CONSTRUCTION

This section introduces a probabilistic modeling method of the PLC system embedded with the LD program. An algorithm is proposed to translate the LD program into the HRM, and the constructed HRM is proved to be a BN that captures the underlying dependence model of the execution logic of the LD program. Then, some CPD tables for the HRM nodes are defined and initiated with the failure probabilities of the mapped system components.

A. Qualitative Part Structure Construction

The execution logic of the embedded LD program is ignored by the original component-based FT, RBD, and BN methods. We intend to model the complex relation into the HRM with an automatical method. The key challenge to build the HRM is to organize these contacts, coils, special instructions, and connections in a structured way. Since the microprocessor processes the input signals according to the arrangement of the ladder program from left to right and from top to bottom, we develop an iterative traversal algorithm for the translation. The translation algorithm is shown in Fig. 4. The algorithm makes use of a structure node $\textbf{Struct node } \{\textbf{char}^* \textbf{ type}; \textbf{node}^* \textbf{ left}; \textbf{node}^*\textbf{right};\}$. Based on the structure node, the algorithm builds an undirected graph model. All the contacts, special instructions, and connections are mapped to the nodes in the graph structure.

We will now examine the algorithm in detail. The coil of each ladder rung is ignored and will be processed in the next procedure. In the first case, the *ladder2* is the minimal ladder block that is serially connected to the rest of the ladder logic block. We traverse the contacts of the ladder rung to find the

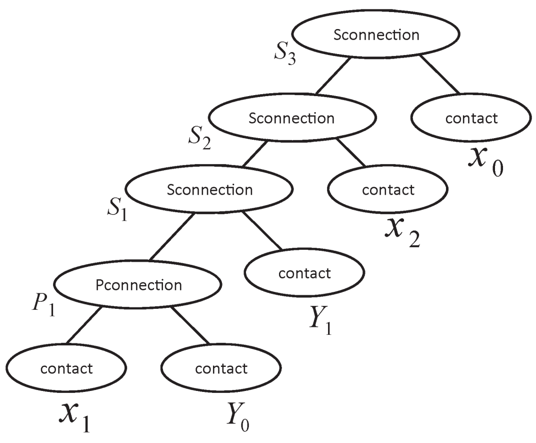


Fig. 5. Graph structure result from the algorithm.

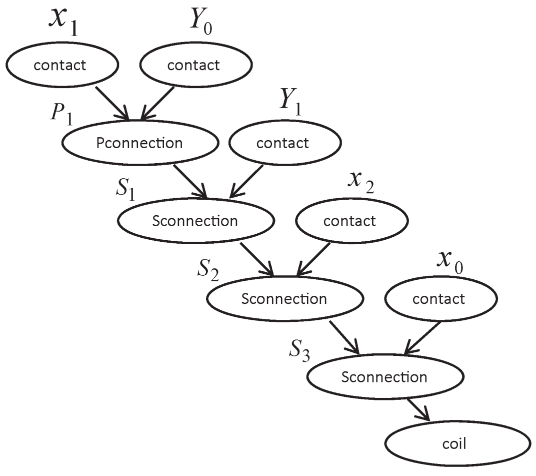


Fig. 6. HRM for the PLC system controlled by LD shown in Fig. 2.

last contact I or the last parallel-connected structure IS, where I or IS is serially connected with the proceeding contacts. The *ladder2* consists of the ladder block I or IS, and the *ladder1* consists of the rest of the ladder rung. The original ladder rung is translated into an *Sconnection* node, with two subgraph structure construction tasks. In the second case, the *ladder1* and *ladder2* are the maximal ladder logic blocks that are in parallel connection, which can be processed similarly to the first case. The atomic instructions of a ladder rung must be one of the last three cases, including the regular instructions such as logic contacts and special instructions such as timer. The result of the translation algorithm for the first ladder rung in Fig. 2 is a graph structure shown in Fig. 5.

When we get the undirected graph structure with the proposed algorithm, we can accomplish the construction process by the following three actions: adding a coil node to the root of the graph structure, rotating the graph structure, and adding all the arcs with direction from the top to bottom. The final graph structure in Fig. 6 is the HRM corresponding to the first ladder rung in Fig. 2. It is obvious that the execution logic of the LD program, processing from the left to right, is captured in this DAG. We present more formal proof process hereinafter. If we apply this translation framework from the first ladder rung to the final ladder rung repeatedly, the execution logic of the LD program, processing from the top to bottom, is also captured.

Then, we prove that the constructed HRM is a minimal representation of the underlying dependence model of the execution logic of the LD program and is a BN. The proof process is based on the description in [28] and [29]. We start by introducing some basic concepts such as conditional independence and d separation. Then, we prove that all dependences among the LD program can be captured in the HRM. The following fundamental definitions have been defined in [28].

Definition 1: Let M be a dependence model, which consists of a set of random variables $V = \{v_1, v_2, \ldots, v_n\}$ and a probability function P over these variables. V_1, V_2, and V_3 are the subsets of V. Then, V_1 and V_2 are said to be conditional independent given V_3, if the joint probability function P over these three subsets satisfies

$$P(V_1|V_2, V_3) = P(V_1|V_3) \quad \text{or} \quad P(V_2|V_1, V_3) = P(V_2|V_3).$$

The conditional independence between V_1 and V_2 given V_3 is denoted by the notation $I(V_1, V_3, V_2)$.

Definition 2: A subset V_1 of V is called a Markov blanket of element v_i, if $I(v_i, V_i; V - V_i - v_i)$ holds. Furthermore, the subset V_i is called a Markov boundary of v_i, if none of its proper subset satisfies the triple independence relation.

Definition 3: Let $d = \{v_{d1}, v_{d2}, \ldots, v_{dn}\}$ be a reordering of the random variables $\{v_1, v_2, \ldots, v_n\}$ of the dependence model M. The boundary strata of M relative to d are an ordered set $\{V_{d1}, V_{d2}, \ldots, V_{dn}\}$, where V_{di} is a Markov boundary of v_{di} with respect to the set $\{v_{d1}, v_{d2}, \ldots, v_{d(i-1)}\}$. The DAG D, created by designating each V_{di} as the parents of the corresponding vertex v_{di}, is called a boundary DAG of M relative to the ordering d.

Definition 4: Let D be a DAG, which consists of a set of nodes $N = \{n_1, n_2, \ldots, n_n\}$ and arcs among them. N_1, N_2, and N_3 are the subsets of N. N_1 and N_2 are said to be d separated given N_3, if there is no path between any node in N_1 and N_2 satisfying the following two properties: 1) Every node on the path with converging arrows is in N_3 or has a descendant in N_3, and 2) every node on the path without converging arrows is outside N_3. The d-separated relation between N_1 and N_2 given N_3 is denoted by the notation $\langle N_1|N_3|N_2\rangle$.

Definition 5: A DAG D is said to be an I-*map* of a dependence model M, if every d-separation condition displayed in D corresponds to a valid conditional independence relation in M. Furthermore, D is a minimal I-*map* of M, if none of the arc can be deleted without destroying its implied dependence model M.

Based on the dependence model M, the DAG D, and those definitions of joint probability functions on the dependence model M and the DAG D, the following definition proved in [28] shows the equivalence between DAG and BN.

Definition 6: If the DAG D is a boundary DAG of the dependence model M relative to an ordering d, D is a minimal I-*map* of M. Furthermore, if the DAG D is a minimum I-*map* of the dependence model M, D is called a BN of M.

Finally, we draw the conclusion that shows the equivalence between the HRM and BN as follows.

Theorem 1: The constructed HRM is a minimal I-map of the underlying execution logic dependence model of the LD program used to control PLC system and is a BN.

Proof: Let $\{v_i\}$ denote the signal flowing through the ladder rung from the left to right, including the signals generated by the input sensors (contacts) and the result generated by the execution of the PLC microprocessor (special instructions and connections between contacts). It is a one-to-one mapping to HRM node $\{n_i\}$. We give a reordering d over the random signal variables $\{v_i\}$. The ordering d satisfies the property: For two random variables v_i and v_j, v_i must appear before v_j if v_i is on the left side of v_j. That means that signal v_i is the input signal of the connection logic execution of the PLC microprocessor while signal v_j is the output of the execution. With this ordering, we derive the Markov boundary of a variable according to the dependence model with the following two principles: 1) If v_j represents the output signal of PLC microprocessor logic execution, its Markov boundary is the signal variables $\{v_i\}$ that are used to generate v_j, and 2) if v_i represents the signal generated by a contact and a special instruction execution, then its Markov boundary is null. Then, the parents of each node variable are its Markov boundary in the HRM. This is an important and useful feature captured by the translation algorithm and the direction arcs on the translated graph structure. By Definition 6, the HRM is a boundary DAG, a minimal I-map of execution logic dependence model M of the ladder program, and, thus, a BN. ■

According to the theorem, the constructed HRM in Fig. 6 is a minimal *I-map* of the underlying dependence model of the first ladder rung shown in Fig. 2. The execution logic of the LD program can be mapped into the HRM nodes through the translation algorithm and the arcs of the translated graph structure. The contacts, special instructions, coils, and connections of the LD program can be mapped to the PLC system components. For example, the contacts can be mapped to input devices such as sensors, the coils can be mapped to actuators such as motors, and the connections and special instructions can be mapped to the execution of the PLC microprocessor. We harmonize the system components and the embedded software through this mapping process.

B. Quantitative Part Structure Construction

As described in Section II, the quantitative part of the BN is a set of CPD tables. Each CPD table is driven by the core distribution function $P = \{f(n_i^t|\text{parent}(n_i^t))\}$. The joint probability distribution can be expressed by the product of each conditional probability. In the HRM, we have four types of nodes: the *coil* node, *contact* node, *special* node, and *connection* node. Each node can be mapped to a corresponding hardware component. We need to incorporate the failure probabilities of the corresponding hardware components into these variables by well-defined CPD tables. Those CPD tables are listed in Tables II–V.

Table II is the CPD table of the *contact* node. Because the *contact* node is mapped to the input devices such as the sensor and switch, we must use the failure probabilities of these components to initialize the table. In the table, x represents the correct input of the PLC system, and ε_s represents the failure probability of the input devices. The *contact* node O_s has four possible values: 1) "10" represents that the correct input should be one but the actual sampling value of the component turns out to be zero; 2) "01" represents that the correct input should be zero but the actual sampling value of the component turns out to be one; 3) "00" represents that the correct input should be zero and the actual sampling value of the component is zero; and 4) "11" represents that the correct input should be one and the actual sampling value of the component is one.

The *special* node mapped to special instructions such as timer and counter can be regarded as an execution of the PLC microprocessor, just like an input sampling of the sensor. It is

TABLE II
CPD FOR CONTACT NODE

$P(O_s\|x)$	$O_s = 10$	$O_s = 01$	$O_s = 00$	$O_s = 11$
1	ε_s	0	0	1 - ε_s
0	0	ε_s	1 - ε_s	0

TABLE III
CPD FOR SPECIAL INSTRUCTION NODE

$P(O_i\|r)$	$O_i = 10$	$O_s = 01$	$O_s = 00$	$O_s = 11$
1	ε_p	0	0	1 - ε_p
0	0	ε_p	1 - ε_p	0

TABLE IV
CPD TABLE FOR CONNECTION NODE THAT REPRESENTS THE SERIAL LOGIC EXECUTION OF PLC MICROPROCESSOR

$P(O_e\|O_1, O_2)$	$O_e = 10$	$O_e = 01$	$O_e = 00$	$O_e = 11$
(00, 00)	0	ε_s	1 - ε_s	0
(00, 01)	0	ε_s	1 - ε_s	0
(00, 10)	0	ε_s	1 - ε_s	0
(00, 11)	0	ε_s	1 - ε_s	0
(01, 00)	0	ε_s	1 - ε_s	0
(01, 01)	0	1 - ε_s	ε_s	0
(01, 10)	0	ε_s	1 - ε_s	0
(01, 11)	0	1 - ε_s	ε_s	0
(10, 00)	0	ε_s	1 - ε_s	0
(10, 01)	0	ε_s	1 - ε_s	0
(10, 10)	1 - ε_s	0	0	ε_s
(10, 11)	1 - ε_s	0	0	ε_s
(11, 00)	0	ε_s	1 - ε_s	0
(11, 01)	0	1 - ε_s	0	ε_s
(11, 10)	1 - ε_s	0	0	ε_s
(11, 11)	ε_s	0	0	1 - ε_s

TABLE V
CPD FOR CONNECTION NODE THAT REPRESENTS THE PARALLEL LOGIC EXECUTION OF PLC MICROPROCESSOR

$P(O_e\|O_1, O_2)$	$O_e = 10$	$O_e = 01$	$O_e = 00$	$O_e = 11$
(00, 00)	0	ε_s	1 - ε_s	0
(00, 01)	0	1 - ε_s	ε_s	0
(00, 10)	1 - ε_s	0	0	ε_s
(00, 11)	ε_s	0	0	1 - ε_s
(01, 00)	0	1 - ε_s	ε_s	0
(01, 01)	0	1 - ε_s	ε_s	0
(01, 10)	ε_s	0	0	1 - ε_s
(01, 11)	ε_s	0	0	1 - ε_s
(10, 00)	1 - ε_s	0	0	ε_s
(10, 01)	ε_s	0	0	1 - ε_s
(10, 10)	1 - ε_s	0	0	ε_s
(10, 11)	ε_s	0	0	1 - ε_s
(11, 00)	ε_s	0	0	1 - ε_s
(11, 01)	ε_s	0	0	1 - ε_s
(11, 10)	ε_s	0	0	1 - ε_s
(11, 11)	ε_s	0	0	1 - ε_s

translated as a leaf node with the proposed algorithm, just like the *contact* node, as shown in Fig. 5. Hence, the CPD table for the *special* node has the same structure as Table II. ε_p is the failure probability of the PLC microprocessor, denoting the probability that the processor generates an error output of the instruction execution.

Then, let us consider the CPD table of the *connection* node. The CPD table for the serial logic connection is presented in Table IV. The table is coincident to those of the *contact* node and *special* node. The connection between atomic instructions can be regarded as parallel and serial logic executions of the PLC microprocessor. Each *connection* node (O_e) has two parent nodes (O_1, O_2). The *connection* node also has four possible values. "10" represents that the correct output of this logic execution should be one but the microprocessor generates actual output of zero due to some uncertain errors. The uncertain errors include the errors inheriting from its parent nodes and the errors caused by the current logic execution of the PLC microprocessor. The other three values can be explained in the same way.

Take Fig. 6 as an example. We map the *contact* node Y_1 to O_1 and the parallel *connection* node P_1 to O_2 and generate the value of the serial *connection* node S_1. The value of (Y_1, P_1) is (11, 01). It means that the correct output of the two nodes should be (1, 0) but the actual output of the two nodes is (1, 1). The error of P_1 may be caused by the sampling error inheriting from X_1 and Y_0 or by the logic execution of the PLC microprocessor. The error will be propagated to the serial *connection* node S_1. When S_1 inherits this error, the actual output will turn out to be one while the correct output should be zero. Furthermore, when the PLC microprocessor happens to be in error at the same time, the wrong output, due to the error inheriting from P_1, will be inverted. The error will be canceled out. Hence, the values of S_1 are "10" with probability of $1 - \varepsilon_p$ and "11" with probability of ε_p. Detailed information about this case is presented in the 14th line of Table IV. The other lines can be explained in the same way.

The CPD table for the parallel *connection* node is presented in Table V. The structure and the probability of each value are defined in the same manner with that of the serial *connection* node. Take Fig. 6 as an example. We map the *contact* node X_1 to O_1 and the *contact* node Y_0 to O_2 and generate the value of the first *connection* node P_1. The value of (X_1, Y_0) is (11, 01). It means that the correct output of the two nodes should be (1, 0) but the actual output of the two nodes is (1, 1). The error of Y_0 may be caused by the sampling error and will be propagated to the *connection* node P_1. When P_1 inherits this error and the PLC microprocessor happens to be in error at the same time, the actual output will turn out to be zero while the correct output should be one. Furthermore, when P_1 inherits this error only, the error will be canceled out. Hence, the values of P_1 are "10" with probability of ε_p and "11" with probability of $1 - \varepsilon_p$. Detailed information about this case is presented in the 14th line of Table V.

Finally, we initiate the CPD table for the *coil* node. It is presented in Table VI. The *coil* node can be mapped to the actuator device of the PLC system. The actuator device is just controlled by the output signal of the PLC system. Hence, the

TABLE VI
CPD TABLE FOR THE COIL NODE

$P(O_c\|O_e)$	$O_c = 10$	$O_c = 01$	$O_c = 00$	$O_c = 11$
00	0	ε_a	1 - ε_a	0
01	0	1	0	0
10	1	0	0	0
11	ε_a	0	0	1 - ε_a

coil node cannot cancel out the errors inheriting from its parent node. When the values of its parent node are "01" and "10," the actuator will be in error with probability of one. When the values of its parent are "00" and "11," the *coil* node will be in error with the probability of ε_a. ε_a is the reliability of the actuator, denoting the probability that the actuator fails to accomplish the task.

We have incorporated the reliability of the PLC input devices, PLC microprocessor, and actuators into these CPD tables. What we need is to change the values of ε_s, ε_p, and ε_a according to the stated operation environment. We can perform predictive and diagnostic inferences to study the behavior of the whole system or a single component.

IV. HRM COMPUTATION

A. Inference Algorithm

Given the HRM graphical model and the corresponding CPD tables, we can evaluate all possible predictive and diagnostic inferences efficiently. A number of algorithms have been invented to solve the inference problems. One of the most popular algorithms is the message passing algorithm, which is an exact inference algorithm. It is based on the local message passing on a junction-tree structure, the nodes of which are subsets of the original random variables. All the approximate inference methods are based on sampling techniques. Based on those algorithms, many software tools such as Netica [30] have been implemented.

Before we perform the inference algorithms, we must first compile the HRM to get the final junction tree. The compiling process to get the nodes of the junction tree named as clique and the connections among the cliques is divided into the following three steps.

1) Graph moralization: We must check every node in the HRM. If the node has more than one parent and the parents have not been connected by arcs, we add undirected edges among those parents to ensure that every parent–child set is a complete graph. Then, we delete the direction of arcs in the HRM to get the moral graph.
2) Graph triangulation: We look for the cycles longer than three nodes in the moral graph and break them into the cycles of three nodes by additional edges. The purpose is to transform the moral graph into a complete graph. We can triangulate the moral graph by more than one way. Some heuristic schemas are adopted to make the moral graph triangulated by adding a minimum number of edges [29], [31].
3) Tree formation: The node of the junction tree named clique is a maximal set of nodes where all nodes in the set are connected. Global consistency is automatically

preserved by constructing the junction tree in the following manner: 1) Between any two cliques C_i and C_j, there is a unique path, and 2) the variables in the set $C_i \cap C_j$ must be contained in all the cliques along the path. A tree can be constructed according to heuristic schemas presented in the previous step to meet the two properties.

Then, inference is performed by maintaining functions related to the joint distributions of all variables in the clique. Let us take an information flow example in two neighboring cliques to understand the key feature of the message passing. Let clique $C_1 = \{N_i\}$, clique $C_2 = \{N_j\}$, and node set $S_{12} = C_1 \cap C_2 = \{N_s\}$ which is the intersection of the two cliques. Then, the probabilities of them are defined as follows:

$$P(C_1) = \prod_{n_i \in C_1} f\left(n_i|\text{parent}(n_i)\right) \tag{3}$$

$$P(C_1, C_2) = \frac{P(C_1)P(C_2)}{P(S_{12})}. \tag{4}$$

Equation (3) is about the conditional distribution of the variable C_1 in the junction tree. It is expressed by the conditional probabilities of the variables in the original HRM. The expressions for variables C_2 and S_{12} are in the same manner. Equation (4) is the joint probability function over the cliques C_1 and C_2. They must agree on the probability of the node set S_{12}. When there is new evidence for the clique C_1, the CPD for the clique C_2 must also be changed to adapt to the agreement on S_{12}. Thus, we need to compute the marginal probability of S_{12} from probability potential of the clique C_1 and use this probability scaling factor to scale the probability potential of C_2. This transmission process is called message passing, and every new piece of evidence is added to the network by this type of local message passing.

B. System Reliability

We have presented the computational mechanism to analyze the reliability characterization of the PLC system corresponding to each independent ladder rung. The LD program is composed of many ladder rungs, which may relate with each other. For example, the output of a ladder rung may work as an input of another ladder rung. As a result, we have to consider the effect of this relation as well as how to combine the independent failure probability of each ladder rung to get the reliability characterization of the whole PLC system.

First, we have to find out how the relation will affect the failure probability of another ladder rung. We take the program in Fig. 2 as an example to illustrate the relation and its effects. As to the motor reversible control program, the output $O(Y_0)$ of the first ladder rung is an input $I(Y_0)$ of the second ladder rung. We must calculate the reliability characterization of $O(Y_1)$ with the consideration of the input $I(Y_0)$. As we can see, when we translate the second ladder rung with the proposed algorithm, the contact $I(Y_0)$ will be translated as a leaf node similar to the other primary inputs. We need to initiate the CPD of this node in the form of Table II. The first method is to set the ε_s with a value of zero and cascade the translated graph structure of the first ladder rung (Fig. 5) to the node corresponding to $I(Y_0)$. The second method is to set the ε_s with the failure probability of the first ladder rung, because the error probability of $I(Y_0)$ is the error probability of $O(Y_0)$ of the first ladder rung. The relation can be captured with either of the two methods. If the LD program is considerably large and has high connectivity between ladder rungs, the size of the cascaded HRM becomes large. This results in large number of CPD tables and complex calculation with limited memory resources. In our work, we prefer to use the second method.

Second, we need to know how to combine the independent failure probability of each ladder rung. After the failure probability of each ladder rung is obtained by applying the inference algorithm, we can get the final reliability characterization function for the whole PLC system as follows.

Theorem 2: The reliability characterization function of a PLC system is

$$f = 1 - \prod_{i=1}^{i \leq n} \left(f(O_i)\right)$$

where $f(O_i)$ is the failure probability of the ith ladder rung and n is the total number of the ladder rungs.

Proof: The formula $(1 - f(O_i))$ denotes the probability that the ith ladder rung is not in failure, and $\prod_{i=1}^{i \leq n}(1 - f(O_i))$ denotes the probability that all the ladder rungs are not in failure. The failure probability of a PLC system is the sum of all the combinations that there is at least one ladder rung in failure. It equals to one minus the probability that none of the ladder rungs is in failure. ■

V. Experiment Result

In this section, we illustrate our method with a small example. Then, we validate our method through more complex industry applications. For those applications, the HRM is constructed with the proposed algorithm, and the conditional probabilities are assigned by the well-defined CPD tables. We use Netica for compiling the junction tree and propagating the probabilities in order to compute the reliability characterization of these systems. We also calculate the reliability characterization of those systems with the original component-based RBD and BN methods. Finally, we devise some random simulations to confirm the correctness of the reliability characterization. The simulations run on a computer with 3.06-GHz CPU and 2 GB of memory. The Monte Carlo simulation framework for the reliability analysis is based on fault injection. We embed the error probabilities of the system components into the Monte Carlo simulator.

A. Small Example

Let us calculate the PLC system that controls the motor to move forward and stop. It consists of some sensors to sample the inputs of buttons, a processor to process those inputs according to the embedded LD program shown in Fig. 2, and the actuator motor. The HRM of the system corresponding to the first ladder rung is shown in Fig. 6. The HRM corresponding to the second rung is similar. We set ε for each component

TABLE VII
CPD FOR NODE P_1

$P(O_e \mid O_1, O_2)$	$O_e = 10$	$O_e = 01$	$O_e = 00$	$O_e = 11$
(00, 00)	0	0.02	1 - 0.02	0
(00, 01)	0	1 - 0.02	0.02	0
(00, 10)	1 - 0.02	0	0	0.02
(00, 11)	0.02	0	0	1 - 0.02
(01, 00)	0	1 - 0.02	0.02	0
(01, 01)	0	1 - 0.02	0.02	0
(01, 10)	0.02	0	0	1 - 0.02
(01, 11)	0.02	0	0	1 - 0.02
(10, 00)	1 - 0.02	0	0	0.02
(10, 01)	0.02	0	0	1 - 0.02
(10, 10)	1 - 0.02	0	0	0.02
(10, 11)	0.02	0	0	1 - 0.02
(11, 00)	0.02	0	0	1 - 0.02
(11, 01)	0.02	0	0	1 - 0.02
(11, 10)	0.02	0	0	1 - 0.02
(11, 11)	0.02	0	0	1 - 0.02

TABLE VIII
RELIABILITY FOR MOTOR CONTROL SYSTEM

processor ε	BN	RBD	HRM	simulation
0.01	0.42%	0.42%	0.53%	0.61%
0.02	0.67%	0.67%	0.83%	0.91%
0.03	0.73%	0.73%	0.95%	1.02%
0.04	0.84%	0.84%	1.13%	1.25%
0.05	1.57%	1.57%	2.12%	2.30%
0.06	2.31%	2.31%	3.51%	3.76%
0.07	3.92%	3.92%	4.67%	4.83%
0.08	4.69%	4.69%	5.27%	5.52%
0.09	5.21%	5.21%	6.47%	6.59%
0.10	5.67%	5.67%	7.13%	7.42%

as follows: 0.05 for the sensor, 0.02 for the processor, and 0.01 for the motor. Then, we use these failure probabilities of components to initiate the quantitative part of the HRM. For example, the CPD table of node P_1 that represents the reliability of the processor output signal is an instance of Table V, shown in Table VII.

The reliability characterization of the motor control system with the HRM-based methods is 0.83%. If we do not consider the execution logic of the embedded LD program, the reliability characterization for hardware-component-based BN and RBD methods is 0.67%, while the failure probability of the system simulation is 0.91%. We change the failure probability of the processor and keep the others the same. The results are presented in Table VIII. As can be observed in those results, in comparison to the original component-based RBD and BN methods, the HRM-based reliability analysis is more close to the simulation results.

B. Complex Applications

The first complex application is an actual industrial PLC system, which was originally published in [32]. The hardware component distribution of the system is shown in Fig. 7. It consists of four pistons $(A,\ B,\ C,$ and $D)$ which are operated by four solenoid valves $(V_1,\ V_2,\ V_3,$ and $V_4)$. Each piston has two corresponding normally open limit sensor contacts. Three push buttons are provided to start the system, stop the system normally, and stop the system immediately in emergency. In a manufacturing facility, such piston systems can be used to

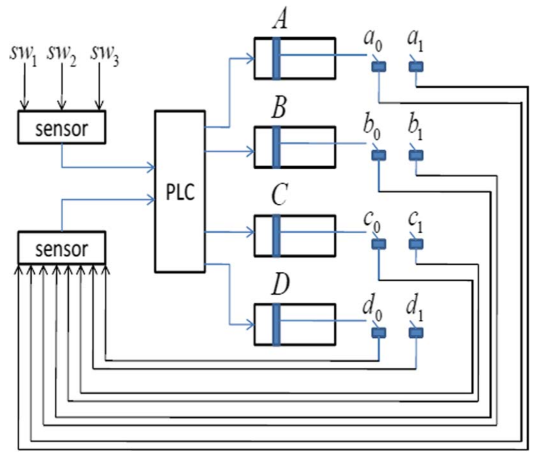


Fig. 7. Component distribution of the industrial automated system.

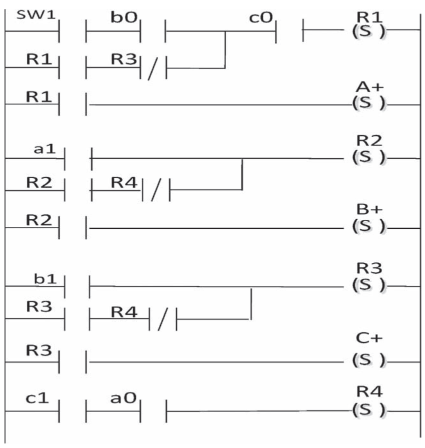


Fig. 8. First diagram used to control the piston system in Fig. 7.

load/unload parts from a machine table and extend/retract a cutting tool spindle.

There are four embedded LD programs used to control the aforementioned hardware component deployment. The four LD programs are shown in Figs. 8–11. The functions of the four programs are presented in [32]. They accomplish different services by actuating those pistons in different sequences. We set the failure probabilities of the system components in Table IX. We can also change the failure probability of the processor and keep the others the same. Then, the failure probabilities of the system corresponding to the four LD programs are listed in Tables X–XIII, respectively.

As can be observed from the simulation results presented in the fifth column of Tables X–XIII, the final failure probability of the piston system is different when it is controlled by different LD programs. With the same PLC processor failure probability, more complex LD programs will lead to higher failure probability. This is due to the fact that a more complex LD program will engender more complex arrangements of the logic executions of the system components. On the other hand, the reliability characterizations obtained by the original component-based BN and RBD methods, as presented in the second and third columns of the four tables, are the same. The values are not close to the simulation results. This is due to the fact that they mainly consider the distribution of the

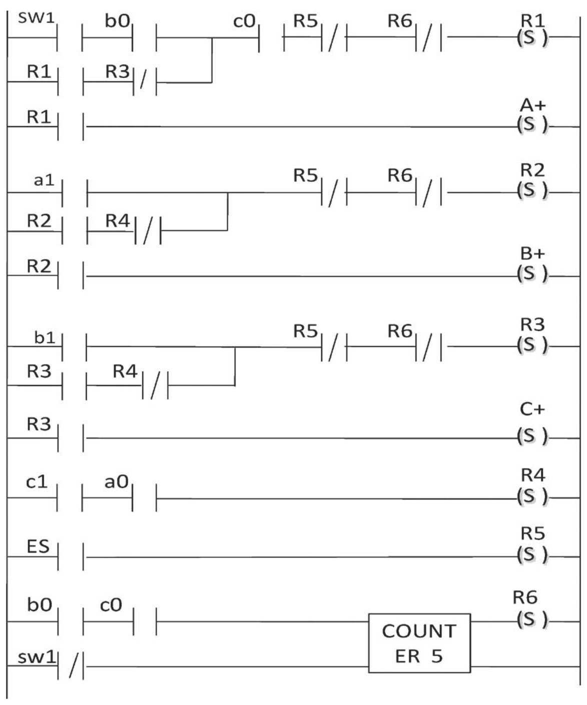


Fig. 9. Second diagram used to control the piston system in Fig. 7.

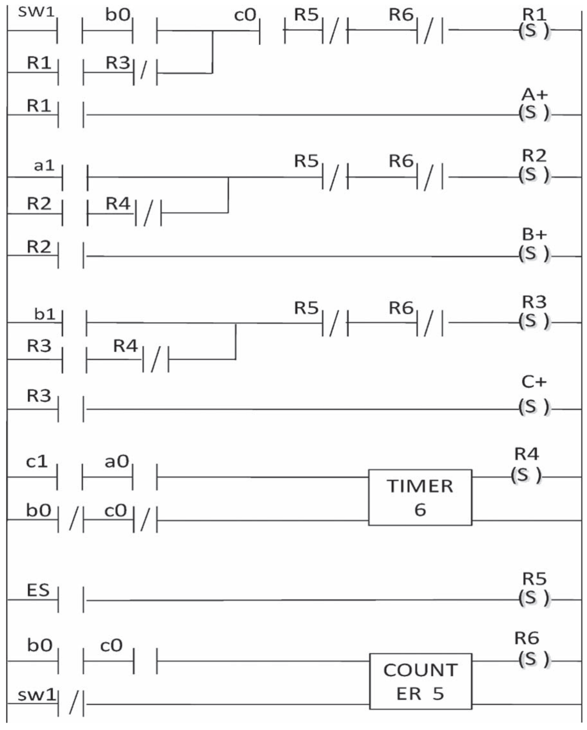


Fig. 10. Third diagram used to control the piston system in Fig. 7.

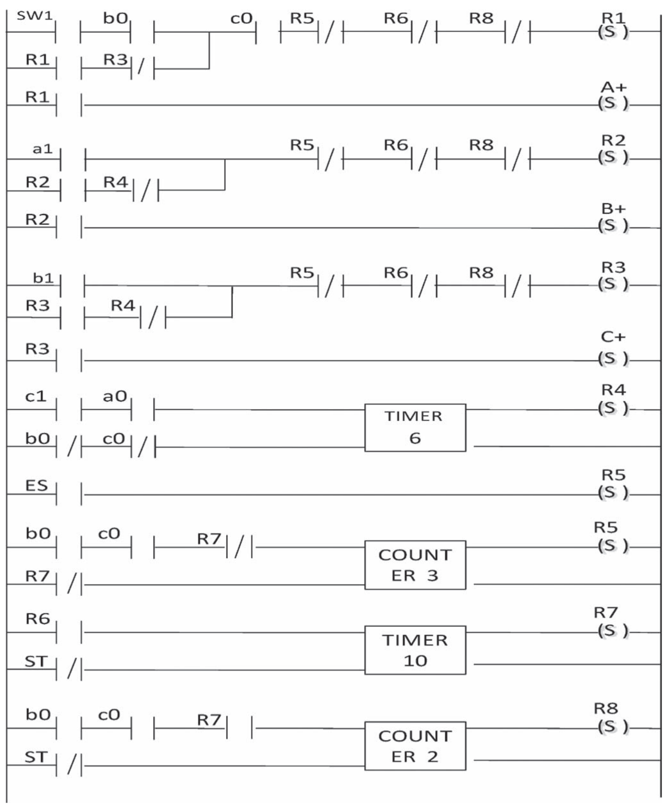


Fig. 11. Fourth diagram used to control the piston system in Fig. 7.

TABLE IX
STATIC FAILURE PROBABILITY OF EACH COMPONENT

component	failure probability
sensors	ε_s 0.05
switches	ε_s 0.05
PLC microprocessor	ε_p 0.05
pistons	ε_a 0.05

TABLE X
RELIABILITY FOR THE PISTON SYSTEM CONTROLLED BY THE LADDER 1

processor ε	BN	RBD	HRM	simulation
0.01	0.46%	0.46%	0.58%	0.61%
0.02	0.57%	0.57%	0.73%	0.82%
0.03	0.72%	0.72%	0.98%	1.05%
0.04	0.94%	0.94%	1.34%	1.41%
0.05	2.10%	2.10%	2.72%	2.85%
0.06	4.61%	4.61%	5.37%	5.64%
0.07	6.12%	6.12%	7.27%	7.59%
0.08	8.03%	8.03%	9.26%	9.61%
0.09	8.71%	8.71%	11.37%	11.72%
0.10	9.24%	9.24%	14.21%	14.81%

system components, the signal dependences among them, and the complex relations caused by the execution logic of the LD program being ignored. The HRM-based method is more accurate. As shown in the fourth column of each table, the error between the HRM results and the simulation results is less than 3%. The results gathered by way of the HRM are more close to the run time station, because the failure probability of the single system component and the complex execution logic of the control program are captured by the HRM and CPD tables. Furthermore, for the four LD programs, the total time for the HRM construction of our algorithm and the evidence propagation in Netica is within 0.1 s.

The other three more complex industry applications are introduced in detail in the papers [33]–[35], including the system component distributions and the embedded LD programs. The detailed LD programs are also presented in the website [36]. The first is a PLC control system for a secondary clarifier scum removal that is installed in the Deer Island Water Pollution Treatment Facility near Boston, MA. The second is a discrete manufacturing PLC system, representing a component sorting,

TABLE XI
RELIABILITY FOR THE PISTON SYSTEM CONTROLLED BY THE LADDER 2

processor ε	BN	RBD	HRM	simulation
0.01	0.46%	0.46%	0.69%	0.73%
0.02	0.57%	0.57%	0.81%	0.89%
0.03	0.72%	0.72%	1.12%	1.20%
0.04	0.94%	0.94%	1.53%	1.62%
0.05	2.10%	2.10%	3.94%	4.11%
0.06	4.61%	4.61%	6.47%	6.78%
0.07	6.12%	6.12%	8.32%	8.63%
0.08	8.03%	8.03%	11.26%	11.92%
0.09	8.71%	8.71%	13.51%	13.89%
0.10	9.24%	9.24%	16.02%	16.67%

TABLE XII
RELIABILITY FOR THE PISTON SYSTEM CONTROLLED BY THE LADDER 3

processor ε	BN	RBD	HRM	simulation
0.01	0.46%	0.46%	0.87%	0.90%
0.02	0.57%	0.57%	1.09%	1.20%
0.03	0.72%	0.72%	1.71%	1.82%
0.04	0.94%	0.94%	2.46%	2.81%
0.05	2.10%	2.10%	4.67%	4.79%
0.06	4.61%	4.61%	7.03%	7.31%
0.07	6.12%	6.12%	9.52%	9.71%
0.08	8.03%	8.03%	11.97%	12.21%
0.09	8.71%	8.71%	14.66%	14.92%
0.10	9.24%	9.24%	17.17%	17.50%

TABLE XIII
RELIABILITY FOR THE PISTON SYSTEM CONTROLLED BY THE LADDER 4

processor ε	BN	RBD	HRM	simulation
0.01	0.46%	0.46%	0.99%	1.06%
0.02	0.57%	0.57%	1.22%	1.30%
0.03	0.72%	0.72%	1.98%	2.09%
0.04	0.94%	0.94%	2.71%	2.82%
0.05	2.10%	2.10%	4.99%	5.12%
0.06	4.61%	4.61%	7.45%	7.52%
0.07	6.12%	6.12%	10.27%	10.59%
0.08	8.03%	8.03%	12.37%	12.62%
0.09	8.71%	8.71%	15.41%	15.77%
0.10	9.24%	9.24%	18.57%	18.92%

TABLE XIV
RELIABILITY FOR THE DOUBLE-DOOR CONTROL SYSTEM

processor ε	BN	RBD	HRM	simulation
0.01	1.53%	1.53%	1.77%	1.83%
0.02	1.66%	1.66%	2.06%	2.11%
0.03	2.13%	2.13%	2.70%	2.76%
0.04	2.94%	2.94%	3.44%	3.51%
0.05	3.87%	3.87%	4.76%	4.83%
0.06	6.12%	6.12%	8.54%	8.62%
0.07	7.67%	7.67%	10.83%	11.02%
0.08	9.13%	9.13%	13.11%	13.42%
0.09	11.32%	11.32%	15.80%	16.21%
0.10	13.11%	13.11%	18.72%	19.23%

assembly, and inspection process. The third is a double-door control PLC system in the Lingshan stage control system in China. We set the failure probability of the system components similar to that of the piston system in Fig. 1. All failure probabilities of the system components are set to be 0.05. Then, the reliability characterizations of the three systems are presented in Table XIV–XVI, respectively. From the three tables, we can see that the HRM is accurate, even with these complex applications, that the error between the simulations results and the HRM-based results is also within 0.3, and that the total time is also within 0.1 s.

TABLE XV
RELIABILITY FOR DISCRETE MANUFACTURING PLC SYSTEM

processor ε	BN	RBD	HRM	simulation
0.01	1.61%	1.61%	1.92%	1.96%
0.02	1.72%	1.72%	2.11%	2.17%
0.03	2.24%	2.24%	2.90%	2.98%
0.04	2.98%	2.98%	3.63%	3.73%
0.05	4.57%	4.57%	6.31%	6.42%
0.06	5.32%	5.32%	8.82%	8.99%
0.07	7.35%	7.35%	11.74%	12.03%
0.08	9.88%	9.88%	13.89%	14.21%
0.09	11.54%	11.54%	16.67%	17.12%
0.10	13.87%	13.87%	19.72%	20.34%

TABLE XVI
RELIABILITY FOR CLARIFIER SCUM REMOVAL SYSTEM

processor ε	BN	RBD	HRM	simulation
0.01	1.57%	1.57%	1.84%	1.87%
0.02	1.69%	1.69%	2.20%	2.25%
0.03	2.71%	2.61%	3.36%	3.42%
0.04	3.12%	0.94%	3.91%	3.98%
0.05	4.59%	4.59%	6.37%	6.47%
0.06	5.41%	5.41%	8.83%	8.96%
0.07	7.62%	7.62%	11.81%	12.10%
0.08	9.89%	9.89%	13.89%	14.25%
0.09	11.31%	11.31%	16.78%	17.21%
0.10	14.19%	14.19%	20.14%	20.58%

VI. EXPERIMENT ANALYSIS

The experiment results demonstrate the accuracy of our model. Now, we give a mathematical time complexity analysis of our model to demonstrate the efficiency and scalability of our model. The time complexity of the translation algorithm is $O(m \cdot n_{\max})$, where m is the number of ladder rungs in the LD program and $n_{\max}$ is the number of contacts in the longest ladder rung. The time complexity of the exact inference in Netica is $O(m \cdot c_n \cdot 4^{|c_{\max}|})$, where c_n is the number of cliques in the compiled junction tree of the original HRM and $|c_{\max}|$ is the number of contacts in the largest clique.

From the aforementioned complexity analysis, we can draw the conclusion that the proposed HRM is more close to the simulation results than the original component-based BN and RBD framework and that the calculation time consumption is not expensive. Moreover, the flexibility of the HRM is also useful, because, for different samples of failures, only the failure probability of the component changes. The constructed HRM of the system will not change, and the CPD table for each node needs to be adapted only once. It makes the HRM-based reliability analysis very efficient for large PLC systems with complex LD programs.

VII. CONCLUSION

In this paper, we have proposed an HRM for the reliability analysis of the PLC system. The model is constructed based on the embedded LD program. The execution logic of the embedded program is mapped to the HRM through the proposed translation algorithm. The model consists of four kinds of nodes and some arcs among those nodes. The HRM is formally proved to be a BN. Then, we have defined some CPD tables for the nodes according to the semantic of each kind of node.

All the nodes can be mapped to some corresponding hardware components through the CPD table initialization. Through these two mapping processes, both the execution logic of the embedded control software and the hardware components are captured. Then, it provides us a convenient way to carry on predictive inferences about the reliability properties of the PLC system.

REFERENCES

[1] M. Villani, M. Tursini, G. Fabri, and L. Castellini, "High reliability permanent magnet brushless motor drive for aircraft application," *IEEE Trans. Ind. Electron.*, vol. 59, no. 5, pp. 2073–2081, May 2012.
[2] J. Wang, M. Chen, X. Wan, and C. Wei, "Ant-colony-optimization-based scheduling algorithm for uplink CDMA nonreal-time data," *IEEE Trans. Veh. Technol.*, vol. 58, no. 1, pp. 231–241, Jan. 2009.
[3] W. Blischke and D. Murthy, *Reliability: Modeling, Prediction, and Optimization*. New York: Wiley, 2011, ser. Wiley Series in Probability and Statistics.
[4] U. S. Dept. Defense, Reliability Modeling and Prediction, Military Standard, Washington, DC, U.S. Dept. Defense, 1981.
[5] X. Yu and A. Khambadkone, "Reliability analysis and cost optimization of parallel inverter system," *IEEE Trans. Ind. Electron.*, vol. 59, no. 10, pp. 3881–3889, Oct. 2011.
[6] F. Chan and H. Calleja, "Reliability estimation of three single-phase topologies in grid-connected PV systems," *IEEE Trans. Ind. Electron.*, vol. 58, no. 7, pp. 2683–2689, Jul. 2011.
[7] G. Petrone, G. Spagnuolo, R. Teodorescu, M. Veerachary, and M. Vitelli, "Reliability issues in photovoltaic power processing systems," *IEEE Trans. Ind. Electron.*, vol. 55, no. 7, pp. 2569–2580, Jul. 2008.
[8] J. Wang, M. Chen, and J. Wang, "Adaptive channel and power allocation of downlink multi-user MC-CDMA systems," *Comput. Elect. Eng.*, vol. 35, no. 5, pp. 622–633, Sep. 2009.
[9] W. Lee, D. Grosh, and F. Tillman, "Fault tree analysis, methods, and applications—A review," *IEEE Trans. Rel.*, vol. R-34, no. 3, pp. 194–203, Aug. 1985.
[10] H. Guo and X. Yang, "A simple reliability block diagram method for safety integrity verification," *Reliab. Eng. Syst. Saf.*, vol. 92, no. 9, pp. 1267–1273, Sep. 2007.
[11] X. Zang, H. Sun, and K. Trivedi, "A BDD-based algorithm for reliability evaluation of phased mission systems," *IEEE Trans. Rel.*, vol. 48, no. 1, pp. 50–60, Mar. 1999.
[12] M. Bouissou and J. Bon, "A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes," *Reliab. Eng. Syst. Saf.*, vol. 82, no. 2, pp. 149–163, Nov. 2003.
[13] S. Distefano and L. Xing, "A new approach to modeling the system reliability: Dynamic reliability block diagrams," in *Proc. IEEE Annu. RAMS*, 2006, pp. 189–195.
[14] C. Bai, Q. Hu, M. Xie, and S. Ng, "Software failure prediction based on a Markov Bayesian network model," *J. Syst. Softw.*, vol. 74, no. 3, pp. 275–282, Feb. 2005.
[15] B. Ayhan, M. Chow, and M. Song, "Multiple discriminant analysis and neural-network-based monolith and partition fault-detection schemes for broken rotor bar in induction motors," *IEEE Trans. Ind. Electron.*, vol. 53, no. 4, pp. 1298–1308, Jun. 2006.
[16] B. Li, M. Chow, Y. Tipsuwan, and J. Hung, "Neural-network-based motor rolling bearing fault diagnosis," *IEEE Trans. Ind. Electron.*, vol. 47, no. 5, pp. 1060–1069, Oct. 2000.
[17] A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla, "Improving the analysis of dependable systems by mapping fault trees into Bayesian networks," *Reliab. Eng. Syst. Saf.*, vol. 71, no. 3, pp. 249–260, Mar. 2001.
[18] "Complex system reliability modelling with dynamic object oriented Bayesian networks (DOOBN)," *Reliab. Eng. Syst. Saf.*, vol. 91, no. 2, pp. 149–162, Feb. 2006.
[19] D. Bruckner and R. Velik, "Behavior learning in dwelling environments with hidden Markov models," *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3653–3660, Nov. 2010.
[20] J. Wang and X. Xie, "Optimal odd-periodic complementary sequences for diffuse wireless optical communications," *Opt. Eng.*, vol. 51, no. 9, p. 095 002-1, Sep. 2012.
[21] W. Bolton, *Programmable Logic Controllers*. Burlington, MA: Newnes, 2009.
[22] G. Dunning and Dunning, *Introduction to Programmable Logic Controllers*. Albany, NY: Delmar, Thomson Learning, 2002.
[23] PLC Programming Languages, IEC 61131-3 Std., Int. Electrotech. Commission (IEC), 2003, 2nd ed.
[24] Y. Yan and H. Zhang, "Compiling ladder diagram into instruction list to comply with IEC 61131-3," *Comput. Ind.*, vol. 61, no. 5, pp. 448–462, Jun. 2010.
[25] R. Sun, J. Zhu, and J. Teng, "Transformation algorithm from PLC instruction list to ladder diagram based on block-growth," *Comput. Knowl. Technol.*, vol. 33, pp. 36–43, 2008.
[26] L. Huang and W. Liu, "Algorithm of transformation from PLC ladder diagram to structured text," in *Proc. IEEE 9th Int. Conf. Electron. Meas. Instrum.*, 2009, pp. 4-778–4-782.
[27] J. Pearl, "Fusion, propagation, and structuring in belief networks * 1," *Artif. Intell.*, vol. 29, no. 3, pp. 241–288, Sep. 1986.
[28] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann, 1988.
[29] R. Cowell, A. Dawid, S. Lauritzen, and D. Spiegelhalter, "Probabilistic networks and expert systems," in *Statistics for Engineering and Information Science*. New York: Springer-Verlag, 1999.
[30] N. Manual, Netica V1.05, Norsys Software Corp., 1997.
[31] U. Kjærulff, Triangulation of Graphs—Algorithms Giving Small Total State Space, Aalborg, Denmark, Univ. Aalborg, Mar. 1990.
[32] 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. Ind. Electron.*, vol. 41, no. 6, pp. 611–619, Dec. 1994.
[33] M. Zhou and E. Twiss, "Design of industrial automated systems via relay ladder logic programming and Petri nets," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 1, pp. 137–150, Feb. 1998.
[34] M. Uzam and A. Jones, "Discrete event control system design using automation Petri nets and their ladder diagram implementation," *Int. J. Adv. Manuf. Technol.*, vol. 14, no. 10, pp. 716–728, 1998.
[35] R. Wang, X. Song, J. Zhu, and M. Gu, "Formal modeling and synthesis of programmable logic controllers," *Comput. Ind.*, vol. 62, no. 1, pp. 23–31, Jan. 2011.
[36] PLC System Ladder Program for the Three Complex Applications. [Online]. Available: https://sites.google.com/site/jiangyu198964

Yu Jiang received the B.S. degree in software engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2010. He is currently working toward the Ph.D. degree in computer science at Tsinghua University, Beijing.

His current research interests include domain specific modeling, formal verification, and their applications in embedded systems.

Hehua Zhang received the B.S. and M.S. degrees in computer science from Jilin University, Changchun, China, in 2001 and 2004, respectively, and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2010.

She is currently a Lecturer with the School of Software, Tsinghua National Laboratory for Information Science and Technology, Tsinghua University. Her current research interests include domain specific modeling, formal verification, and their applications in embedded systems.

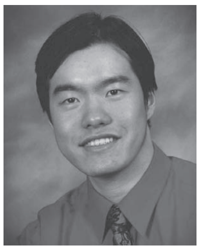
Xiaoyu Song received the Ph.D. degree from the University of Pisa, Pisa, Italy, 1991.

From 1992 to 1999, he was a Faculty Member with the University of Montreal, QC, Canada. In 1998, he was a Senior Technical Staff with Cadence Design Systems, San Jose, CA. Since 1999, he has been with Portland State University, Portland, OR, where he is currently a Professor with the Department of Electrical and Computer Engineering. His current research interests include formal methods, design automation, embedded system design, and emerging technologies.

Dr. Song was a recipient of the Intel Faculty Fellowship Program during 2000–2005. He served as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS and the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.

Xun Jiao is currently working toward the B.S. degree in telecommunication engineering and management in the International School, Beijing University of Posts and Telecommunications, Beijing, China.

His current research interests include wireless communication, wireless sensor network, formal verification, and their applications in embedded systems.



William N. N. Hung received the B.S. and M.S. degrees in electrical and computer engineering from The University of Texas, Austin, in 1994 and 1997, respectively, and the Ph.D. degree in electrical and computer engineering from Portland State University, Portland, OR, in 2002.

From 1997 to 2004, he was a Senior Engineer with Intel Corporation, Hillsboro, OR. From 2004 to 2007, he was a Senior Staff Engineer and a Director with Synplicity, Sunnyvale, CA. Since November 2007, he has been a Senior Staff R&D Engineer and a Senior R&D Manager with Synopsys, Inc., Mountain View, CA. His research interests include constraint solving, logic synthesis, physical design, formal methods, combinatorial optimization, nanotechnology, and quantum computing.

Dr. Hung is currently the Chair of the Quantum Computing Task Force for the Emergent Technologies Technical Committee of the IEEE Computational Intelligence Society. He served as the Session Chair for the Design Automation Conference and World Congress on Computational Intelligence and as the Publications Chair for the Formal Methods in Computer-Aided Design. He served as the Cochair of the Logic and Circuit Track in the Technical Program Committee of the International Conference on Computer Design. He was also a member in the Technical Program Committees of various conferences, including Design, Automation and Test in Europe, Computer-Aided Verification, and Congress on Evolutionary Computation.

Ming Gu received the B.S. degree in computer science from the National University of Defense Technology, Changsha, China, in 1984 and the M.S. degree in computer science from the Chinese Academy of Sciences, Shenyang, China, in 1986.

Since 1993, she has been a Lecturer, an Associate Professor, and a Researcher with Tsinghua University, Beijing, China, where she is also the Vice Dean of the School of Software. Her research interests include formal methods, middleware technology, and distributed applications.

Jiaguang Sun received the B.S. degree in automation science from Tsinghua University, Beijing, China, in 1970.

He is currently a Professor with Tsinghua University, where he is also currently the Director of the School of Information Science and Technology and of the Tsinghua National Laboratory for Information Science and Technology, School of Software. He is dedicated in teaching and R&D activities in computer graphics, computer-aided design, formal verification of software, and system architecture.

Prof. Sun has been a member of the Chinese Academy of Engineering since 1999.