On the uncertainty characterization of programmable logic controllers

Hehua Zhang, Yu Jiang, William N. N. Hung, Guowu Yang, and Ming Gu

***Abstract*—Programmable Logic Controllers (PLCs) are widely used in industry. Reliable PLCs are vital to many critical applications like nuclear power plants control. This paper presents a probabilistic modeling for PLC programs. Three analysis strategies are proposed for probabilistic evaluation of uncertainty characterization. The first method is an input-based analysis which considers the impact of errors from primary inputs. The second one is an action-based analysis. It extends the first method by considering the impact of processing deviations on primary inputs. The third method is an action-traverse analysis. It computes the uncertainty characterization in a single topological traverse through the primary inputs during the program execution. Experiment results demonstrate the effectiveness of our approaches.**

***Index Terms*—Input error, processing deviation, PLC, uncertainty analysis.**

I. Introduction

PRogrammable Logic Controllers (PLCs) are widely used in industry to control machinery on factory, spaceport devices or nuclear power plants. Many PLC applications are safety-critical and the reliability of PLCs are vital. As a result, modeling and verification of PLC programs has been widely studied. Most of the existent methods modeling PLC programs as automata [1], [2], [3], [4] , Petri nets[5] or specific logics. Formal verification techniques [6], [7], [8] like model checking and theorem proving are then proposed for analysis.

Though the existent deterministic analysis of PLC programs are valuable, the uncertain errors caused by noise, environment, or hardware should not be neglected [9]. The casual errors may happen when PLC sampling the inputs, computing the data, or outputting its signals.

In this paper, we present an approach to characterize PLC programs in a probabilistic way. Three analysis strategies are proposed for probabilistic evaluation of uncertainty characterization. The first method is an input-based analysis which considers the impact of errors from primary inputs. The second one is an action-based analysis. It extends the first method by considering the impact of processing deviations on primary inputs. The third method is an action-traverse analysis. It computes the uncertainty characterization in a single topological traverse through the primary inputs during the program execution. Experiment results demonstrate the effectiveness of our approaches.

The paper is organized as follows. We introduce some basic concepts in Section II, including an introduction to PLC and its ladder diagram programming language. The probabilistic modeling of PLC is given in Section III. We explain the input-based analysis algorithm in Section IV. Then, we present the action-based analysis algorithm in Section V, including the introduction of action and the method to transfer a PLC program into action units. The action-traverse analysis is introduced in Section VI, explaining how the units are sorted and processed. Section VII extends the three algorithms to deal with the relationship and combine the uncertainty of ladder steps to establish the uncertainty characterization of an entire PLC program. Section VIII provides the experiment results. Finally, Section IX concludes the paper.

II. BACKGROUND

PLC is essentially an industrial control computer, which works in a periodic scanning mechanism. Each cycle is composed by three stages (see Fig. 1). The first stage is sampling. PLC reads all the input data and state into the corresponding I/O. The second stage is PLC processing. The CPU of PLC applies logical operations to the control circuit which is composed of contacts with the order from left to right and top to bottom. It then refreshes the state of the I/O image area related to the output coil and determines whether to run any special functional instruction (e.g.: $JUMP$). The last stage is Actuating. The CPU refreshes all the outputs according to the state of their I/O image, and actuates the peripheral via the output circuits. Two types of PLC programs are distinguished: (1) without memory and (2) with memory to store state of the system. The former is called a combinatorial PLC program whose output is based on the primary input, while the latter is referred as a sequential PLC program whose output is based on the primary input and the state of the memory. We consider the former kind of PLC programs in this paper.

The ladder diagram is widely used as a graphical programming language for PLCs. There are two types of basic instructions in a ladder diagram language: (1) The instructions representing the conditions on the ladder and are composed of primary inputs and logic connections; and (2) Special instructions located at the right side of the ladder, and determined by the conditional instructions. Fig. 2 shows a sample ladder program with several frequently-used symbolic instructions. It consists of four ladder steps.

Hehua Zhang and Ming Gu are with the School of Software, TNLIST, Tsinghua University, China.

Yu Jiang is with the Dept. of Computer Science and Technology, TNLIST, Tsinghua University, China

William N. N. Hung is with Synopsys Inc., Mountain View, California, USA.

Guowu Yang is with the University of Electronic Science and Technology of China, Sichuan, China.

This research is supported in part by NSFC Program (No.90718039) and 973 Program (No.2010CB328003) of China.

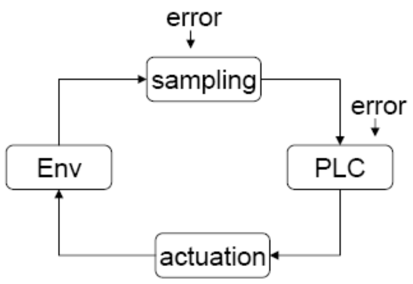


Fig. 1. The four phases of a PLC cycle.

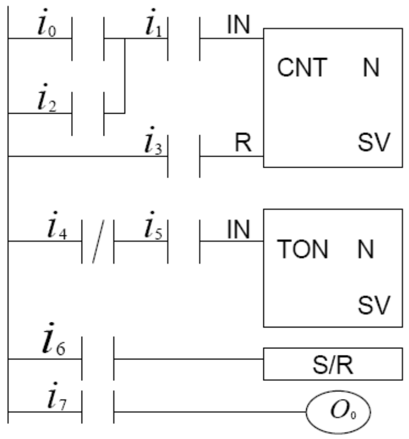


Fig. 2. An example of ladder program.

The symbol $-|\ |-$ is a normally open contact I, said to be the "load" operation ($LD(I)$). When the value of i_0 is 1, the contact stays in the closed state, and the current flows through the contact. The symbol $-|/|-$ is a normally close contact, said to be the "load not" operation ($LD\ NOT(I)$). When the value of i_4 is 0, the contact stays in the closed state, and the current flows through the contact. $-|\ |-|/|-$ represents a serial connection of two kinds of contacts, said to be the "and" operation (AND). When the values of i_4 and i_5 are 0 and 1, respectively, the current can flow through the operation. Similarly, in the first cascade, i_0 and i_2 are connected in parallel, said to be the "or" operation (OR). When at least one value of i_0 and i_2 is 1, the current can flow through the operation. CNT is a counter instruction, where N represents the pre-set value of the counter, and SV is the current count. When the conditional input IN changes from 0 to 1, SV is incremented by 1. If SV is between 0 and N, the output of the counter is 1. When the conditional input R is 1, the counter will be reset ($SV = 0$) and the output is 0. TON is a timer instruction, where N represents the pre-set value of the timer, and SV is the current time. When the conditional input IN is 1, SV is incremented in a fixed step. When SV is equal to N, the output of the timer becomes 1. When IN is 0, the timer will be reset. O_0 is a normal output instruction. When the value of i_7 is 1, the instruction will refresh the corresponding I/O image with 1. S/R is the set-reset instruction. When the conditional input i_6 is 1, S will refresh the corresponding I/O image with 1 and R will refresh the corresponding I/O image with 0. Detailed instructions can be found in [10]. Our work is based on a subset of the instructions, shown in TABLE I, where $JUMP$ is the jumping instruction and EQ, LE, GE are compare instructions.

TABLE I
A SUBSET OF PLC INSTRUCTIONS

Category	Instruction symbol
conditional	$LD, LD\ NOT, AND, AND\ NOT, OR, OR\ NOT$
output	O, S, R
timer	TON
counter	CNT
jump	$JUMP$
compare	EQ, LE, GE

III. MODELING OF PLC PROGRAMS

In this section, we introduce a probabilistic modeling of PLC programs denoted by ladder graphs. We consider the error probability of the primary input sampling and also of each operating unit. An algorithm is presented to model a ladder program as a structured expression of the operation unit.

A. Probabilistic Modeling of Input Sampling

We record the contact j of the ladder as the primary input I_j. Each I_j has the static probability and the sampling probability. Static probability means the probability of a particular input I_j being 1(0) at a given time, denoted by $P_j^1(P_j^0)$, and the scheme can be defined as follows according to [11].

$$\begin{aligned} P_j^1 &= \lim_{N\to\infty} \frac{\sum_{T=1}^{N} I_j(T)}{N} \\ P_j^0 &= 1 - P_j^1 \end{aligned}$$

Sampling probability is the probability of a particular input I_j being $0 \to 1$ ($0 \to 0$, $1 \to 0$, $1 \to 1$), which means that the actual input is 0 $(0, 1, 1)$ but the sampling input turns out to be 1 $(0, 0, 1)$, denoted by P_j^{01} (P_j^{00}, P_j^{10}, P_j^{11}). The scheme can be defined as:

$$P_j^{01} = \lim_{N\to\infty} \frac{\sum_{T=1}^{N} \overline{I_j(T)} I_j^s(T)}{N}$$

$$P_j^{10} = \lim_{N\to\infty} \frac{\sum_{T=1}^{N} I_j(T) \overline{I_j^s(T)}}{N}$$

$$P_j^{00} = \lim_{N\to\infty} \frac{\sum_{T=1}^{N} \overline{I_j(T) I_j^s(T)}}{N}$$

$$P_j^{11} = \lim_{N\to\infty} \frac{\sum_{T=1}^{N} I_j(T) I_j^s(T)}{N}$$

The symbol T represents the execution cycle of the ladder program. $I_j(T)$ represents the original value of the contact j during the cycle T. $I_j^s(T)$ represents the sampling value of the contact at cycle T, which is the actual value of the primary input participating in the processing stage. It denotes the fact that the original input from the environment is $I_j(T)$, and this input will be sampled, while the sampling stage may introduce some error as shown in Fig.1. As the result, the final input passed to the PLC processing stage is the sampled input $I_j^s(T)$.

TABLE II
THE THREE OPERATION UNITS OF LOGICAL INSTRUCTIONS

Ladder	Operation	Failure Pro	Effect Pro
$-\|\|-\|\|-_{(AND)}$	$A(a,b)$	ε_A	E_A
$-\|\|\succ_{(OR)}$	$O(a,b)$	ε_O	E_O
$-\|/\|-_{(LDNOT)}$	$N(a)$	ε_N	E_N

TABLE III
THE TWO OPERATION UNITS OF COMPARE INSTRUCTIONS

Ladder	Operation	Failure Pro	Effect Pro
$a == b_{(CMP?I)}$	$Q(a,b)$	ε_Q	E_Q
$a < b_{(CMP?I)}$	$L(a,b)$	ε_L	E_L

B. Probabilistic Modeling of Ladder Programs

The uncertainty characterization analysis of PLC ladder programs refers to the problem of evaluating the effects of errors caused by the deviation of primary input sampling and deviations of instruction processing. Sampling error happens when the accurate input is 1(0), but the PLC program read 0(1) into the corresponding I/O image. Arithmetic processing error happens when the accurate output of the above conditional and special instruction is 1(0), but the actual output of the instruction turns out to be 0(1). The probability of these two kinds of errors depend on noise, environment, hardware, etc.

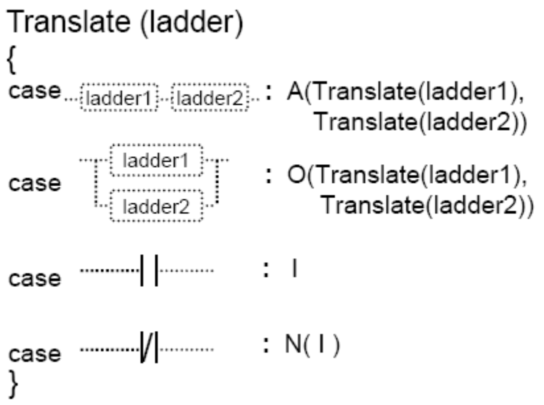
TABLE II and TABLE III together show all the basic operation units we defined. In TABLE II, three basic logical instructions are represented as operation units with one or two operands. In TABLE III, two compare instructions are further defined as operation units. Other instructions can be expressed by these basic operation units. For each unit, we consider two kinds of probabilities: failure probability and effect probability. The failure probability describes the probability when the original result of an operation unit is 1 but the actual result turns out to be 0. It is denoted by ε_A (taking the A operation as an example). On the other hand, the output of the ladder step may also turn out to be wrong when the result of an operation unit is wrong. Effect probability describes the probability of this circumstance, and is denoted by E_A (for the A operation). It can be calculated using boolean difference, symbolic techniques based on binary decision diagrams or simulation [12], [13], [14].

With these basic operation units, we can describe a step of a ladder program as follows.

1) In order to facilitate the processing of errors caused by sampling, the sampling of input is regarded as a special A operation (AND) with the operands the original input $I_j(T)$ and 1. The result of this operation is the actual input $I_j^s(T)$. That is $LD(I_j(T)) = A(I_j(T),1)$, and, the ε of the operation $A(I_j(T),1)$ can be expressed by the sum of P_j^{01} and P_j^{10}. As a result, $\varepsilon_{sampling} \approx P_j^{10} + P_j^{01}$. The effect probability $E_{sampling}$ of this operation is calculated according to the method to calculate E_A.
2) The output instruction at the right side of the step can be expressed as: $O(IN) = A(IN,1)$, $S(IN) = A(IN,1)$, $R(IN) = A(IN,0)$, where IN is the conditional input determined by the primary input. Then, the ε and E of the output instruction can be calculated in the same way as the basic A operation.
3) The comparison instructions are basic operations defined in TABLE III, and the ε and E can be found there. If these units are used in parallel, the result of the units will be connected to the logical instruction of the step by an O operation. If these units are used in series, the result will be connected to the logical instruction by an A operation.
4) As to the timer instruction at the right side of the step, the result of the instruction $TON\ N$ is determined by the conditional input and the current time value. When the current value SV equals to the pre-set value N and the value of the conditional input is 1, the output of the timer is 1. That is $TON\ N = A(Q(N,SV),IN)$. Therefore, when there is a timer, we decompose it into operations A and Q, then the uncertainty analysis of a timer can be done with the method for the units A and Q. On the other hand, if we hope to treat the special instruction TON as a single operation unit to be dealt with directly, we need to package the two kinds of probabilities for A and Q. The processing deviation ε of a timer instruction is determined by an Q operation and an A operation. When only one operation fails, the result will change. Hence, we have $\varepsilon_{TON} \approx \varepsilon_Q \cdot (1-\varepsilon_A) + \varepsilon_A \cdot (1-\varepsilon_Q)$, and $E_{TON} = 1$.
5) As to the counter instruction at the right side of the step, the result of the instruction $CNT\ N$ is determined by the conditional input and the current count value. When the value of the current count SV is between 0 and the pre-set value N, the conditional input changes from 0 to 1, and the value of the reset variable is 0, the output of the counter is 1. That is $CNT\ N = A(A(L(SV,N),L(0,SV))$ $, A(N(R),A(IN_T,N(IN_{T-1}))))$. Consequently, two probabilities of the counter instruction can be done similarly with the timer instruction.
6) When the jump instruction appears at the right side of the step, it can also described by the basic operation units. $JUMP(N)$ means that the ladder will goto step N when the conditional input is 1. Assume that the actual number of the ladder step the program goes to is $N_T^{'}$, if it is equal to the pre-set number N_T, the jump instruction works correctly. So we have $JUMP(N) = A(Q(N_T^{'},N_T),IN)$.

We have shown how to transfer a given instruction into the basic operation units. We then introduce the modeling of ladder program steps by organizing these units in a structured way. Since a PLC processes its input from left to right and from top to bottom, we develop an iterative traversal algorithm for the translation, shown in Fig. 3.

When a ladder step has more than one contact, the structure of the ladder step must be one of the first two cases shown in Fig. 3. We then partition the original ladder step into two sub-steps, say, ladder1 and ladder2). Let us see the first case. We traverse the contacts of the ladder to find the first contact I or the first parallel connected structure SI, which are serially connected with the latter contacts. The ladder1 is then made



Translate (ladder)
{
case ladder1 ladder2 : A(Translate(ladder1), Translate(ladder2))
case ladder1 ladder2 : O(Translate(ladder1), Translate(ladder2))
case : I
case : N(I)
}

Fig. 3. The translating algorithm of ladder steps.

up of the contact I or SI, and also all the contacts preceding I or SI. The ladder2 is composed by all the contacts following I or SI. The step can then be translated into an A operation unit, with $Translate(ladder1)$ and $Translate(ladder2)$ the two operands. Other cases can be processed similarly. At last, a ladder step is translated into a single expression of the basic operation units.

We then need to know the failure probability ε_i and the effect probability E_i of each operation unit. The failure probability is determined by the hardware and the environment and can be set by the testers. The effect probability of the operation unit can be calculated using boolean difference like in [14]. Here we present a different method. Defining the output function of the ladder program on the output action a_i and all the primary inputs that a_i does not cover, the new method to get E_i (equal to $\partial f/\partial a_i$) is defined as follows.

$$\begin{aligned} f_{a_i}(I) &= f(I_1, I_2, \cdots I_j, a_i) \\ f_{a_i=1}(I) &= f(I_1, I_2, \cdots I_j, 1) \\ f_{a_i=0}(I) &= f(I_1, I_2, \cdots I_j, 0) \\ \frac{\partial f}{\partial a_i} &= f_{a_i=1}(I) \oplus f_{a_i=0}(I) \end{aligned}$$

We have transformed the input sampling, output, compare, timer, counter and jump instructions into the operation units and calculated their failure probability and effect probability. Now we can consider the processing deviation of each operation unit. When only one operation unit (i) fails, the probability that the final output will toggle is the product of its effect probability (E_i) and failure probability (ε_i). When multiple operations fail, we have to combine the effect probability of the failed operations to get a joint effect probability, denoted by JE. JE works on a set of operation units and describes the probability when some of the failure operations toggle and the final output toggles too. The probability that the final output toggles (uncertainty characterization) then equals to the product of the failure probabilities of the set of operation units multiplied by their JE. To calculate JE, we introduce the simultaneous effect probability SE. Compared with JE, SE acts on a set of failed operation units, and describes the probability that (1) each failure operation makes the final output toggle once, and (2) after all these operations are processed, the final output toggles compared to the original final output. According to the description of SE, the SE of the set with odd number of failure operation units will cause a toggle of the final output.

Furthermore, two basic assumptions are proposed to calculate the joint effect probability.

Assumption 1: The effect of each failure operation unit at the final output is independent of each other, that is, E_i is independent of each other.

Therefore, when n operation units fail, the JE of them can be expressed by the sum of SE, for all the possible combinations that odd number units have actual effect on the final output:

$$JE = \sum_{|SE|=2i+1} SE \qquad (i \in [0, n-1/2]).$$

Assumption 2: The failure of each operation is independent of each other, that is, ε_i is independent of each other.

As a result, if there are n operation units fail and only k units have effect on the final output, the SE of them can be expressed by the product of their E_i or $1 - E_i$:

$$SE(k) = \prod_{i \in k} E_i \cdot \prod_{i \in n/k} (1 - E_i).$$

We take a small example in Fig. 5 to illustrate the processing of JE and the uncertainty characterization calculation based on the proposed two assumptions. Ignoring the errors caused by sampling and output, Fig. 5 can be expressed by an O operation unit (a_1) and an A operation unit (a_2). When there is only one unit in failure, the probability that the final output toggles (uncertainty characterization) is $f_1 = \varepsilon_1(1-\varepsilon_2)E_1 + \varepsilon_2(1-\varepsilon_1)E_2$. When both operation units fail, by using Assumption 1 and Assumption 2, we get the uncertainty characterization

$$\begin{aligned} f_2 &= \varepsilon_1\varepsilon_2(JE) \\ &= \varepsilon_1\varepsilon_2(\sum_{|SE|=2i+1} SE) \\ &= \varepsilon_1\varepsilon_2(\sum_{|SE|=1} SE) \\ &= \varepsilon_1\varepsilon_2(E_1(1-E_2) + E_2(1-E_1)). \end{aligned}$$

Then, the final uncertainty of the example f is the sum of f_1 and f_2:

$$\begin{aligned} f &= f_1 + f_2 \\ &= \varepsilon_1 E_1 + \varepsilon_2 E_2 - 2\varepsilon_1 E_1 \varepsilon_2 E_2 \\ &= \frac{1}{2} - \frac{1}{2}\prod_{i \in a_i}(1 - 2\varepsilon_i E_i). \end{aligned}$$

IV. Input-Based Analysis

We consider the effect of sampling errors on the output in this section. At each cycle, the PLC needs to sample the input from the environment first. This method is concentrated on sampling, which may affect the response time as well as the correctness of the primary input sampling. On the other side, noise or bad environment may also affect the correctness of sampling. Since input sampling is the entrance of the PLC, we need to find out how this failure will affect the program. In Section III some probabilities about the primary input are provided. Here we will define the uncertainty characterization based on these probabilities. Finally, we will also propose some basic assumptions to simplify the uncertainty characterization function.

A. Computing Uncertainty Characterization

With regard to a step of the ladder program, the original value of all the contacts at cycle T is $(I_1^T, I_2^T \cdots I_j^T)$, denoted by $I(T)$. The original result of the program for the original input is $O(I_{(T)})$. The sampling primary input of these contacts is $(I_1^t, I_2^t \cdots I_j^t)$, denoted by $I(t)$, and the actual output for the actual sampling input is then $O(I_{(t)})$. As mentioned in Section III, the uncertainty characterization analysis of a PLC ladder program refers to evaluating the effects of errors caused by both the deviation of primary input sampling and the deviations in instruction processing. Therefore, the uncertainty characterization function of a step can be expressed as:

$$\begin{aligned} f &= [(O(I_{(T)}) = 0) \wedge (O(I_{(t)}) = 1)] \vee \\ &\quad [(O(I_{(T)}) = 1) \wedge (O(I_{(t)}) = 0)] \\ &= O(I_{(T)}) \oplus O(I_{(t)}). \end{aligned}$$

To simplify the calculation, we propose a basic assumption on the probability of the primary input, and then prove two properties based on the assumption. The properties can facilitate the calculation of f.

Assumption 3: Each primary input I_j is independent of each other and the sampling probability satisfies $P_j^{01} = P_j^{10}$.

Since $P_{j(t)}^1 = P_j^{01} + P_j^{11}$, and $P_{j(T)}^1 = P_j^{10} + P_j^{11}$, we can deduce that $P_{j(t)}^1 = P_{j(T)}^1 = P_j^1$.

Lemma 1: If the input I_j ($j \in N^+$) is independent and o = $I_1 \cdot I_2 \cdots I_i \cdot \overline{I}_{i+1} \cdots \overline{I}_j$, then $P(o = 1) = P_1^1 \cdot P_2^1 \cdots P_i^1 \cdot P_{i+1}^0 \cdots P_j^0$.

Proof: Since the inputs are independent, the proof of the lemma is obvious according to the property for the independent events that occur simultaneously. Q.E.D.

Lemma 2: If $A \wedge B = 0$ and $o = A \vee B$, then $P(o = 1) = P(A) + P(B)$.

Proof: Since $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$ and $A \wedge B = 0$, the lemma is proved. Q.E.D.

When we apply these two properties to the characterization function f, it can be expanded into the expression of the static probability and the sampling probability of primary inputs.

B. Case Study

We illustrate the input-based analysis algorithm by a PLC ladder program shown in Fig. 4. This is a motor reversible control program.

The first step of the ladder has five primary input contacts: X_0 (denoting the power input,with the input point I_1), X_1 (the clockwise rotation input I_2), X_2 (the counter-clockwise rotation input I_3), Y_0 (keeping clockwise rotation I_4), Y_0 (the counter-clockwise rotation mutex input I_4). The output of the first step is then computed as:

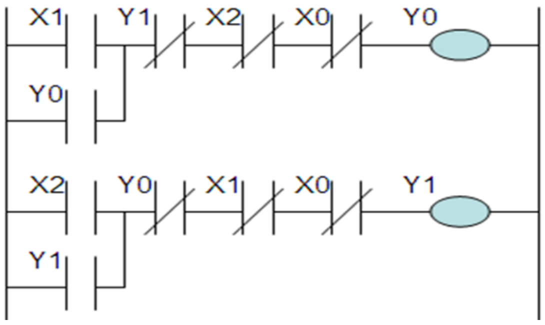


Fig. 4. A motor reversible control program.

$$\begin{aligned} O &= (X_1 \vee Y_0) \wedge \overline{Y}_1 \wedge \overline{X}_2 \wedge \overline{X}_0 \\ &= (I_2 \cdot \overline{I}_5 \cdot \overline{I}_3 \cdot \overline{I}_1) \vee (I_4 \cdot \overline{I}_5 \cdot \overline{I}_3 \cdot \overline{I}_1). \end{aligned}$$

By applying the two properties, the uncertainty characterization function f is finally expanded to the sum of twelve expressions which is the product of two kinds of probabilities:

$$\begin{aligned} f &= O(I_{(T)}) \oplus O(I_{(t)}) \\ &= [(I_2^T \cdot \overline{I}_5^T \cdot \overline{I}_3^T \cdot \overline{I}_1^T) \vee (I_4^T \cdot \overline{I}_5^T \cdot \overline{I}_3^T \cdot \overline{I}_1^T)] \\ &\quad \oplus [(I_2^t \cdot \overline{I}_5^t \cdot \overline{I}_3^t \cdot \overline{I}_1^t) \vee (I_4^t \cdot \overline{I}_5^t \cdot \overline{I}_3^t \cdot \overline{I}_1^t)] \\ &= P_1^{01} \cdot P_2^{10} \cdot P_3^0 \cdot P_4^0 \cdot P_5^0 + P_1^{01} \cdot P_2^{00} \cdot P_3^0 \cdot P_4^{10} \cdot P_5^0 + \\ &\quad \cdots + P_1^{00} \cdot P_2^{10} \cdot P_3^{01} \cdot P_4^0 \cdot P_5^0. \end{aligned}$$

V. ACTION-BASED ANALYSIS

After loading the sampled input values, PLC will enter the processing stage. The CPU will process the input values from left to right and from top to bottom according to the control logic of the ladder program. As shown in Fig.1, there may also be some error caused by hardware. The output of the processing operation unit may toggle because of the errors. When the effect of error introduced in the processing stage is non-ignorable, the method introduced in Section IV is insufficient. On this occasion, we should process the effect of the basic operation units introduced in Section III. After that, the expression of the structured operation unit can be calculated, through the translating algorithm introduced in Section III.

A. Calculating Uncertainty Characterization

Given a step of a ladder program, it can be expressed by the basic operation units. We notate the units of a step as follows. The output is denoted by O; a denotes the set of all operation units (a_i); S is the set of all the nonempty subsets of a; F is an element of the set S, with all each element a_i in F denotes a failed operation unit; 2^F is the power set of F; E is an element of 2^F, with the operations in E have actual effect on the final output.

To illustrate the definitions, we correlate them with the example in Fig. 5. O is the output Y; a is the operation units set $\{a_1,a_2\}$; S is the set $\{\{a_1\},\{a_2\},\{a_1,a_2\}\}$, which means all the possible combinations of operation units that would be in error; F is an element of S, i.e., $\{a_1,a_2\}$, which means both units failed. As a result, 2^F equals $\{\{a_1\},\{a_2\},\{a_1,a_2\},\emptyset\}$, which represents all the possible combinations of SE when two units have failed. E is an element of 2^F, i.e., $\{a_1,a_2\}$, which means that the failed operation units a_1 and a_2 have actual effect on the final output.

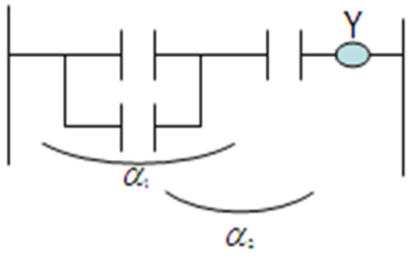


Fig. 5. An example of operation units of a ladder step.

For each combination of the failing operation units F, the uncertainty characterization function $f_{(F)}$ is the product of $W_{(F)}$ (the probability that these units fail) and $JE_{(F)}$ (the joint effect probability for these units). According to Assumption 3, $W_{(F)}$ can be expressed as the product of ε_i (where $a_i \in F$) and $1-\varepsilon_i$ (where $a_i \notin F$). According to Assumption 2, $JE_{(F)}$ can be expressed as the sum of SE, for all possible combinations of odd number units with actual effect on the final output. Consequently, the uncertainty of the output is the sum of all possible combinations. Theorem 1 describes the uncertainty characterization of a step.

Theorem 1: The uncertainty characterization function of a ladder step is:

$$f = \frac{1}{2} - \frac{1}{2}\prod_{a_i\in a}(1-2\varepsilon_i E_i).$$

Proof : From the descriptions, We can deduce the following three formulas:

$$W_{(F)} = \prod_{a_i\in F}\varepsilon_i \prod_{a_i\in a/F}(1-\varepsilon_i) \quad (1)$$

$$JE_{(F)} = \sum_{|E|=2i+1}^{E\in 2^F} SE_{(E)} \quad (2)$$

$$f = \sum_{F\in S} JE_{(F)}\cdot W_{(F)}. \quad (3)$$

We now further analyze formula (2). In terms of Assumption 3, given a random E, we have:

$$SE_{(E)} = \prod_{a_i\in E}E_i \prod_{i\in F/E}(1-E_i).$$

To calculate formula (2), we refer to the method presented in [15] and construct a formula for the constraint $|E| = 2i+1$:

$$se_{(E)} = \prod_{a_i\in E}(-E_i) \prod_{a_i\in F/E}(1-E_i)$$

, satisfying that when $|E|$ is odd, $SE_{(E)} = \frac{1}{2}(SE_{(E)} - se_{(E)})$, and when $|E|$ is even, $\frac{1}{2}(SE_{(E)} - se_{(E)}) = 0$. Therefore, we can get

$$\begin{aligned} JE_{(F)} &= \sum_{|E|=2i+1}^{E\in 2^F} SE_{(E)} = \sum_{E\in 2^F}\frac{1}{2}(SE_{(E)} - se_{(e)}) \\ &= \frac{1}{2}(\sum_{E\in 2^F}\prod_{a_i\in E}E_i\prod_{a_i\in F/E}(1-E_i) - \\ &\quad \sum_{E\in 2^F}\prod_{a_i\in E}(-E_i)\prod_{a_i\in F/E}(1-E_i)). \end{aligned}$$

On the other hand, since 2^F is the power set of F, we can deduce the following results from combinatorics:

$$\begin{aligned} \sum_{E\in 2^F}\prod_{a_i\in E}E_i\prod_{a_i\in F/E}(1-E_i) &= \prod_{a_i\in F}(E_i + (1-E_i)) \\ &= 1 \\ \sum_{E\in 2^F}\prod_{a_i\in E}(-E_i)\prod_{i\in F/E}(1-E_i) &= \prod_{a_i\in F}(-E_i + (1-E_i)) \\ &= \prod_{a_i\in F}(1-2E_i). \end{aligned}$$

Hence, $JE_{(F)}$ can be simplified as:

$$JE_{(F)} = \frac{1}{2}(1 - \prod_{a_i\in F}(1-2E_i)). \quad (4)$$

We then expand formula (3) with formula (1) and (4) and get:

$$\begin{aligned} f &= \sum_{F\in S} JE_{(F)}\cdot W_{(F)} \\ &= \frac{1}{2}\sum_{F\in S}\prod_{a_i\in F}\varepsilon_i\prod_{a_i\in a/F}(1-\varepsilon_i) - \\ &\quad \frac{1}{2}\sum_{F\in S}\prod_{a_i\in F}\varepsilon_i(1-2E_i)\prod_{a_i\in a/F}(1-\varepsilon_i). \quad (5) \end{aligned}$$

Since S is the set of all nonempty subsets of operation units a, we deduce the further results from combinatorial mathematics:

$$\begin{aligned} &\sum_{F\in S}\prod_{a_i\in F}\varepsilon_i\prod_{a_i\in a/F}(1-\varepsilon_i) + \prod_{a_i\in a}(1-\varepsilon_i) \\ &= \prod_{a_i\in a}(\varepsilon_i + 1 - \varepsilon_i) = 1, \quad (6) \end{aligned}$$

$$\begin{aligned} &\sum_{F\in S}\prod_{a_i\in F}\varepsilon_i(1-2E_i)\prod_{a_i\in a/F}(1-\varepsilon_i) + \prod_{a_i\in a}(1-\varepsilon_i) \\ &= \prod_{a_i\in a}(\varepsilon_i(1-2E_i) + (1-\varepsilon_i)) \\ &= \prod_{a_i\in a}(1-2\varepsilon_i E_i). \quad (7) \end{aligned}$$

Applying formula (6) and (7) to simplify formula (5), the result is just the theorem which represents the uncertainty characterization function of the final output. Q.E.D.

The uncertainty characterization of the ladder step which has been transformed to operation units is a closed-form expression ,where ε_i is determined by noise, hardware, environment etc., while E_i can be calculated by boolean differences, binary decision diagrams, or simulation. It is efficient to calculate the uncertainty characterization with Theorem 1.

However, it is not accurate enough with the applying of Assumption 1, since the effect probability of each failing unit is not independent from each other in the real world. Hence, the result is more accurate when there is a small number of failing operation units.

B. Case Study

We take the PLC ladder program in Fig. 4 to illustrate the action-based analysis algorithm. The algorithm in Fig. 3 is applied to translate the first step of the ladder and we get the following expression composed of operation units. $O = A(A(A(O(x_1, Y_0),\ N(Y_1)), N(x_2)), N(x_0))$. Then, we denote each operation unit as follows. $O(x_1, Y_0) = a_1, N(Y_1) = a_2, A(a_1, a_2) = a_3, N(x_2) = a_4, A(a_3, a_4) = a_5, N(x_0) = a_6, A(a_5, a_6) = a_7$. Assuming that the ε_i of each unit is 0.1, we calculate the E_i using BDD and get $E_1 = E_2 = \frac{1}{8}, E_3 = E_4 = \frac{1}{4}, E_5 = E_6 = \frac{1}{2}, E_7 = 1$ Finally, the uncertainty characterization of the step is:

$$\begin{aligned} f &= \frac{1}{2} - \frac{1}{2}\prod_{a_i \in a}(1 - 2\varepsilon_i E_i) \\ &= 0.4401. \end{aligned}$$

If we transform also the sampling of primary input and the output procedure into A operation units, there would be six specific A units. The value of ε_i and E_i can be calculated by the method mentioned in Section III-B. When these specific A units are combined with the seven units $a_1, \ldots, a_7$, the uncertainty characterization of the step will be more accurate for a small amount of failing units.

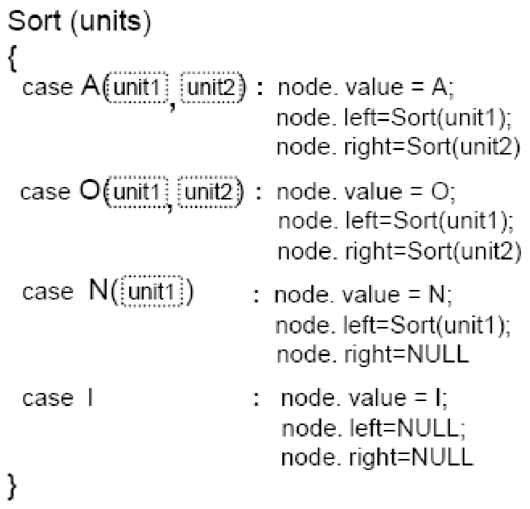
VI. ACTION-TRAVERSE ANALYSIS

We have translated a step of a ladder program into operation units in a structured way, and provided a closed-form expression for the uncertainty characterization. In this section, we improve the accuracy of the action-based analysis for multiple failing units. First, we express the ladder step with operations in the same way we did for action-based analysis. Then, a topological sort is carried out on these units according to that the ladder program processes the input from left to right and from top to bottom. Finally, for each unit, we combine the failure probability of it with the toggle probability it inherits from the preceding unit, to get the toggle probability that it will transmit to its following unit. We process the units in the topological sort one by one, so at last we will get the uncertainty characterization of the final output.

A. Action Traversal

The translating algorithm presented in Section III is taken to translate a ladder step of PLC into operation units. For example, the first step of the motor reversible control ladder program in Fig. 4 is translated into the expression $A(A(A(O(I_2, I_4),\ N(I_5)), N(I_3)), N(I_1))$. We then take an algorithm to decompose the expression and sort it. After that, the operation units can be processed one by one, and we get the final probability of the output at last.

The decomposing and sorting algorithm presented in Fig. VI-A makes use of a structure node with the data structure Struct node {char* value; node* left; node* right; }. Based on it, the algorithm will decompose the expression of the operation units and produce a binary tree. The depth degree of the node in the tree denotes its sequence. The bigger the degree, the further in front the node is. The node with bigger degree then should be processed earlier than the none with smaller degree. In the algorithm, unit1 and unit2 represent the operands of the operation unit being decomposed. The operand is a primary input or the result of a set of operation units.



```
Sort (units)
{
  case A(unit1, unit2) :  node. value = A;
                          node. left=Sort(unit1);
                          node. right=Sort(unit2)

  case O(unit1, unit2) :  node. value = O;
                          node. left=Sort(unit1);
                          node. right=Sort(unit2)

  case  N(unit1)       :  node. value = N;
                          node. left=Sort(unit1);
                          node. right=NULL

  case  I              :  node. value = I;
                          node. left=NULL;
                          node. right=NULL
}
```

Fig. 6. The sorting algorithm.

After the topological sort is decided, the units will be processed one by one. For a single unit, there are two kinds of error sources : (1) the toggle probability inheriting from the previous operation unit; and (2) the failure probability(ε_i) of the unit itself. Both the two error sources should be dealt with to get the toggle probability it transmits to the next unit. The toggle probabilities can also be divided into two cases(1) the original result of the unit is 1 but the actual result is 0, denoted by $P_\varepsilon(a_{1\to 0})$; and (2) the original result of the unit is 0 but the actual result is 1, denoted by $P_\varepsilon(a_{0\to 1})$.

As to each kind of operation unit presented in TABLE II and TABLE III, we need to process the first kind of error source to get the toggle probability it transmits to the next unit.

Considering the operation unit $O(I, J)$ at first. It possesses two operands, each of the operands can be a previous unit a_i, or a primary inputs I_i. As to the primary inputs, the inherited toggle probability need not to be considered. Without loss of generality, we consider the case of (a_i, a_j). The two operation units (a_i, a_j) will propagate the toggle probability $P_\varepsilon(I_{1\to 0})$,$P_\varepsilon(I_{0\to 1})$ of a_i and $P_\varepsilon(J_{1\to 0})$,$P_\varepsilon(J_{0\to 1})$ of a_j to the unit O. These probabilities should be combined for the unit O to get the $P(O_{1\to 0})$,$P(O_{0\to 1})$ it propagates to the next operation unit. If the result of (a_i, a_j) is (1,0), the original result of unit $O(a, b)$ is 1. However, if a_i changes from 1 to 0, and a_j remains 0, the actual result of unit $O(a, b)$ will be 0. Similarly, when the result of (a_i, a_j) is (0,0), the original result of unit $O(I, J)$ is 0. When a_i or a_j changes from 0 to 1, the actual output will be 1. If we know the probability of all the combinations for input(a_i, a_j), and the probability that $a_i(a_j)$ changes from 0 to 1 or 1 to 0, denoted by $P_\varepsilon(I_{0\to1})$, $P_\varepsilon(I_{1\to0})$, we can get the toggle probability that caused by the previous operation unit and transferred to the next unit. The semantics of transmission for operation units $A(I,J)$ and $N(I)$ are similar to the description of $O(I,J)$. The transmission formula for the unit $O(I,J)$, $A(I,J)$ and $N(I)$ can then be presented as follows.

We introduce some definitions first. The probabilities that the original output of the operation unit equals to 1 and 0 are denoted by $P(1)$ and $P(0)$, respectively. The probabilities that (I,J) equals to $(1,1)$, $(0,1)$, $(1,0)$ and $(0,0)$ are denoted by P_{11}, P_{01}, P_{10} and P_{00}, respectively. The $1\to0$ transmission probabilities from the previous operation units for the O, A and N units are denoted by $P_O(1\to0)$, $P_A(1\to0)$ and $P_N(1\to0)$, respectively. The $0\to1$ transmission probabilities from the previous operation units are then denoted by $P_O(0\to1)$, $P_A(0\to1)$, and $P_N(0\to1)$, respectively.

The following lemmas can then be deduced.

Lemma 3 ($O(I,J)$): $P(1) = P_{11} + P_{10} + P_{01}$, $P_O(1\to0) = P^{(1,1)}_{(1\to0)} + P^{(0,1)}_{(1\to0)} + P^{(1,0)}_{(1\to0)}$; $P(0) = P_{00}$, $P_O(0\to1) = P^{(0,0)}_{(0\to1)}$.

Proof : According to the semantic of the operation unit $O(I,J)$, the two transmission probabilities and can be defined as follows.

$$P^{(I,J)}_{(1\to0)} = \begin{cases} P_{11}P_\varepsilon(I_{1\to0})\cdot P_\varepsilon(J_{1\to0}) & (I,J)=(1,1)\\ P_{01}(1-P_\varepsilon(I_{0\to1}))\cdot P_\varepsilon(J_{1\to0}) & (I,J)=(0,1)\\ P_{10}P_\varepsilon(I_{1\to0})\cdot(1-P_\varepsilon(J_{0\to1})) & (I,J)=(1,0)\end{cases}$$

$$P^{(I,J)}_{(0\to1)} = \begin{cases} P_{00}(P_\varepsilon(I_{0\to1})+P_\varepsilon(J_{0\to1})- \\ P_\varepsilon(I_{0\to1})\cdot P_\varepsilon(J_{0\to1})) & (I,J)=(0,0)\end{cases}$$

By summing, the lemma is easily proved. Q.E.D.

Lemma 4 ($A(I,J)$): $P(1) = P_{11}$, $P_A(1\to0) = P^{(1,1)}_{(1\to0)}$; $P(0) = P_{00}+P_{01}+P_{10}$, $P_A(0\to1) = P^{(0,0)}_{(0\to1)} + P^{(0,1)}_{(0\to1)} + P^{(1,0)}_{(0\to1)}$.

Proof : By the semantic of the operation unit $A(I,J)$, the two kinds of transmission probabilities can be defined as:

$$P^{(I,J)}_{(0\to1)} = \begin{cases} P_{00}P_\varepsilon(I_{0\to1})\cdot P_\varepsilon(J_{0\to1}) & (I,J)=(0,0)\\ P_{01}P_\varepsilon(I_{0\to1})(1-P_\varepsilon(J_{1\to0})) & (I,J)=(0,1)\\ P_{10}(1-P_\varepsilon(I_{1\to0}))\cdot P_\varepsilon(J_{0\to1}) & (I,J)=(1,0)\end{cases}$$

$$P^{(I,J)}_{(1\to0)} = \begin{cases} P_{11}(P_\varepsilon(I_{1\to0})+P_\varepsilon(J_{1\to0})- \\ P_\varepsilon(I_{1\to0})\cdot P_\varepsilon(J_{1\to0})) & (I,J)=(1,1)\end{cases}$$

It is easy to know the sum of them are the two kinds of transmission probabilities from the previous unit. Q.E.D.

Lemma 5 ($N(I)$): $P(0) = P_1$, $P_N(0\to1) = P^1_{(0\to1)}$; $P(1) = P_0$, $P_N(1\to0) = P^0_{(1\to0)}$.

Proof : In terms of the semantic of the operation unit $N(I)$, the two kinds of transmission probabilities can be defined as follows.

$$P^I_{(0\to1)} = \begin{cases} P_1P_\varepsilon(I_{1\to0}) & I=1\end{cases}$$

$$P^I_{(1\to0)} = \begin{cases} P_0P_\varepsilon(I_{0\to1}) & I=0\end{cases}$$

Each expression possesses only one ramus, so the result is obviously proven. Q.E.D.

After the first kind of error source is processed so that $P_a(0\to1)$ and $P_a(1\to0)$ ($[a\in A,O,N]$) are obtained, the probability of the second error source (failure probability ε) should be combined to get the final toggle probability $P_\varepsilon(0\to1)$. For example, if the original result of the current unit is 0, and the $P_a(0\to1)$ caused by the previous unit happens but the current unit is not in failure, the output will be changed from 0 to 1.

The calculating of the toggle probabilities for the operation units $Q(a,b)$ and $L(a,b)$ ($P_Q(1) = P_{a=b}, P_L(1) = P_{a<b}$) are different with the former presented three operation units. That is owing to (1) the operands are integers stored in the memory or register; (2) the two operation units are used to describe the instruction $(CNT, TON, JUMP)$ at the right side of the step. According to the two reasons, we assume that they don't have the toggle probability inheriting from the pervious operation unit. For example, if a equals b, while the operation unit $Q(a,b)$ is in failure, the output will toggle from the original value 1 to 0. Therefore, the final toggle probability is defined as follows.

Theorem 2: The final toggle probability that the current operation unit a ($a\in\{A,N,O,E,L\}$) transmits to the next operation unit, is

$$P_\varepsilon(0\to1) = \begin{cases} (1-\varepsilon)(P_a(0\to1)/P(0))+ \\ \varepsilon(1-P_a(0\to1)/P(0)) & a\in\{O,A,N\}\\ \\ \varepsilon\cdot(1-P_a(1)) & a\in\{Q,L\},\end{cases}$$

$$P_\varepsilon(1\to0) = \begin{cases} (1-\varepsilon)(P_a(1\to0)/P(1))+ \\ \varepsilon(1-P_a(1\to0)/P(1)) & a\in\{O,A,N\}\\ \\ \varepsilon\cdot P_a(1) & a\in\{Q,L\}.\end{cases}$$

Proof : Based on Lemma 3, 4, 5 and the text descriptions, the theorem is obvious. Q.E.D.

The left consideration is the toggle probabilities of the sampling and the special instructions CNT, TON and $JUMP$. According to the probabilistic modeling of ladder programs introduced in Section III, the sampling and the special instructions can be expressed by the basic operation units. For example $LD(I_j(T)) = A(I_j(T),1)$ and $TON\ N = A(E(N,SV),IN)$. As a result, a ladder step can be described by the five basic operation units. Then we can use the proved theorem to process the error source of each basic unit, and get the toggle probability of the final output.

On the other hand, if we hope to recognize a special instruction as an independent operation unit to be processed directly, the special instruction should be expressed by the basic operation units first, and then the toggle probabilities of these basic operation units should be packaged.

We take the instruction $TON\ N$ as an example to show how to package. Since $TON\ N = A(Q(N,SV),IN)$, the result of the unit $Q(N,SV)$ should be gotten first. The probabilities of the value of (Q,I) can be defined as follows.

$$P\{(Q,I)=(x,y)\} = \begin{cases} P_{SV<N}\cdot P_I^0 & (x,y)=(0,0)\\ P_{SV<N}\cdot P_I^1 & (x,y)=(0,1)\\ P_{SV=N}\cdot P_I^0 & (x,y)=(1,0)\\ P_{SV=N}\cdot P_I^1 & (x,y)=(1,1)\end{cases}$$

Then, we need to use an A operation unit to package the result of the operation unit $Q(N, SV)$ and the conditional input IN. The transmission probability of the unit $TON\ N$ is as follows.

Lemma 6 (TONN): $P(1) = P_{11}$, $P_T(1 \to 0) = P^{(1,1)}_{(1\to 0)}$; $P(0) = P_{00} + P_{01} + P_{10}$, $P_T(0 \to 1) = P^{(0,0)}_{(0\to 1)} + P^{(0,1)}_{(0\to 1)} + P^{(1,0)}_{(0\to 01)}$.

Proof : According to the semantic of the operation units $A(I, J)$ and $Q(a, b)$, the transmission probabilities can be defined as follows.

$$P^{(Q,I)}_{(0\to 1)} = \begin{cases} (P_{SV<N} \cdot P^0_I) \cdot (\varepsilon \cdot P_{SV<N}) & \\ \cdot (P_\varepsilon(I_{0\to 1})) & (Q, I) = (0, 0) \\ (P_{SV<N} \cdot P^1_I) \cdot (\varepsilon \cdot P_{SV<N}) & \\ \cdot ((1 - P_\varepsilon(I_{1\to 0}))) & (Q, I) = (0, 1) \\ (P_{SV=N} \cdot P^1_I) \cdot (\varepsilon \cdot P_{SV=N} + & \\ P_\varepsilon(I_{1\to 0}) - (\varepsilon \cdot P_{SV=N}) & \\ \cdot P_\varepsilon(I_{1\to 0})) & (Q, I) = (1, 0) \end{cases}$$

$$P^{(Q,I)}_{(1\to 0)} = \begin{cases} (P_{SV=N} \cdot P^1_I) \cdot (\varepsilon \cdot P_{SV=N} + & \\ P_\varepsilon(I_{1\to 0}) - (\varepsilon \cdot P_{SV=N}) & \\ \cdot P_\varepsilon(I_{1\to 0})) & (Q, I) = (1, 1) \end{cases}$$

By calculating the summation, the lemma can be easily proved. Q.E.D.

By replacing the symbol $P_a(0 \to 1)$ in Theorem 2 with $P_T(0 \to 1)$ and the symbol $P_a(1 \to 0)$ with $P_T(1 \to 0)$, the final toggle probability of the instruction $TON\ N$ can be obtained. The result for CNT and $JUMP$ can be obtained similarly.

B. Computing Uncertainty Characterization

Given a step of a ladder program, it is first presented by the five basic units and given a topological sort of all the units from left to right and from top to bottom. Then, Lemma 3, 4, and 5 are applied to process each unit. When the last unit is reached, we get $P_\varepsilon(1 \to 0)$ and $P_\varepsilon(0 \to 1)$. BDD is also used to get the probability of the value for the output (P^1_O, P^0_O). Finally, the characterization function of the step can be defined as follows.

Theorem 3: The uncertainty characterization function of a ladder step is:

$$f = P^0_O P_\varepsilon(0 \to 1) + P^1_O P_\varepsilon(1 \to 0).$$

Proof: Based on Theorem 2 and the explanations, the result is obvious. Q.E.D.

C. Case Study

We take the first step in the program shown in Fig. 4 to illustrate the algorithm. It consists of an O unit, three N units, and three A units. The topological sort of these units are then given as: $O(x_1, Y_0) = a_1, N(Y_1) = a_2, A(a_1, a_2) = a_3, N(x_2) = a_4, A(a_3, a_4) = a_5, N(x_0) = a_6, A(a_5, a_6) = a_7$. Assuming that the ε_i of each unit is 0.1 and the probability of each primary input is $P_0 = P_1 = 1/2$, the units can be processed one by one. When we reach the unit a_7, we get $P_\varepsilon(1 \to 0) = 0.4392$ and $P_\varepsilon(0 \to 1) = 0.164$. At the same time, we get $P^0_O = 3/32$ and $P^1_O = 29/32$ by the BDD method. Finally, the uncertainty characterization of the step is:

$$\begin{aligned} f &= P^0_O P_\varepsilon(0 \to 1) + P^1_O P_\varepsilon(1 \to 0) \\ &= 3/32 * 0.164 + 29/32 * 0.4392 \\ &= 0.4134. \end{aligned}$$

VII. Uncertainty Characterization of PLC

We have presented the algorithms to analyze the uncertainty characterization of an independent ladder step. A PLC ladder program is composed of many steps, which may relate with each other. For example, the output of a step works as one of the input of the next step. As a result, we have to consider the relationship among them as well as how to combine independent characterization of these steps to get the uncertainty characterization of a complete PLC ladder program.

First, We have to find out how the relationship will affect the uncertainty characterization of a ladder step. We take the program in Fig. 4 to illustrate the relationship among steps and their effects. As to the motor reversible control program, the first kind of relationship is that the output $O(Y_0)$ of the first ladder step is one of the input $I(Y_0)$ of the second ladder step. The second kind of the relationship is that the output $O(Y_1)$ of the second ladder step is one of the input $I(Y_1)$ of the first ladder step. Since the PLC processes the inputs from up to down, the two relationships cause different effects. As a result, we develop different methods to deal with the two relationships.

As to the first relationship, we will calculate the uncertainty characterization of $O(Y_1)$ with the consideration of the input $I(Y_0)$. The probability of input $I(Y_0)$ can not be set as another primary input such as X_2, because it is determined by the first ladder step. Therefore, the static probability of $I(Y_0)$ is the static probability of $O(Y_0)$ of the first ladder step which can be calculated by BDD. The sampling error probability of $I(Y_0)$ is the error probability of $O(Y_0)$ of the first ladder step, which can be calculated by the action-traverse analysis method.

The processing of the second relationship can be more complex. That is how to calculate the uncertainty characterization of $O(Y_0)$ in consideration of the input $I(Y_1)$. At the first cycle of the PLC program, $I(Y_1)$ has an initial value 0. We set the probability of the other primary inputs and use the action-traverse analysis method to calculate the static and error probability of output $O(Y_0)$. Then the first relationship in step 2 will be embedded to calculate the static and probability of the output $O(Y_1)$. After that, we go back to step 1, setting the static probability of $I(Y_1)$ as the static probability of the output $O(Y_1)$, the sampling error probability of $I(Y_1)$ as the error probability of $O(Y_1)$. Therefore, we get the real static and error probability of the output $O(Y_0)$. The more iterations we make, the more realistic the result is.

Second, we need to know how to combine the independent characterization of these steps. After the value of $f(O_1)$ and $f(O_n)$ are obtained by applying the action-traverse analysis method, We can get the final uncertainty characterization function for a whole PLC program.

Theorem 4: The uncertainty characterization function of a PLC ladder program is:

$$f = 1 - \prod_{i=1}^{i \leq n} (1 - f(O_n)).$$

Proof : The uncertainty of a PLC program is the sum of all the combinations that there is at least one ladder step in failure. The sum equals to 1 minus the probability that none of the ladder steps is in failure. Q.E.D.

VIII. EVALUATION EXPERIMENTS

We translate the PLC ladder program instructions one by one to the basic operation units. So, the complexity of our translation procedure and the unit processing procedure is linear to the size of a PLC ladder program. We choose five examples to show how these algorithms work. Variants of these five examples are widely used in industrial and consumer electronics. We set the static probability of each primary input (PI) 0.5, the transition probability of primary input 0.1, and the deviation probability of each unit 0.1. The uncertainty characterization values of the five examples are given as follows: Algorithm1 (A1) only considers the primary input, Algorithm2 (A2) considers the primary input and the units, Algorithm3 (A3) considers both of them as well as the related input (RI).

The first application is a ladder program embedded in the ticketed gate entrance for parking lots. When a car arrives at a ticketed gate entrance, the driver can press a button to get a ticket. While the ticket is being issued, it will send a primary input signal to the PLC. The PLC will generate an output to lift the barrier so that the car can enter the parking lot. The lifting of the barrier will trigger a timer (TON) to count down. When the timer reaches zero, the PLC will output a signal to lower the barrier so that other cars cannot enter the parking lot. When the actual operating environment is approximated to the probability we specified, the characterized uncertainty values for A1, A2, and A3 of this ladder program are 0.3367, 0.4326 and 0.3724, respectively.

The second application is a ladder program embedded in the automatic door for any building. Safety is an important concern because the automatic door should be designed to avoid trapping anybody. Sensors are installed near the door to detect people, and to detect the position of the door. Timers are also included in the ladder program to filter noise signals from the sensors. When a person approaches the door, the sensor will be triggered, which will send a signal to the primary input of the PLC. The PLC will generate an output to open the door. When the sensor detects that door is fully opened, the PLC will send a signal to stop opening the door. The door will remain opened until the sensor signals near the door (to detect peoples movement) disappear. It will then trigger a counter to count down. If nobody is near the door anymore, the counter will eventually reach zero, which will trigger the PLC to send a signal to close the door. The "close" signal will be turned off when the door is fully closed. If anybody approaches the door while the door is closing, the PLC will open the door to allow the person to pass through. The uncertainty characterization values of this ladder program for A1, A2, and A3 are 0.5924, 0.7991, and 0.7061 respectively.

The third application is a ladder program embedded in a washing machine, whose variants are widely used in our daily lives. A primary input to the ladder program will initiate the washing sequence. The washing sequence is a series of output signals, each will control the motor of the washing machine to rotate at a different speed. The ladder program uses timers to pace the washing and move from one speed to the next speed. The motor will stop at the end of the washing sequence. The uncertainty characterization of this ladder program for A1, A2, and A3 are 0.5924, 0.6448, and 0.6197, respectively.

The fourth application is a ladder program embedded in a pulse driven electric motor, which is used in assembly lines or other industrial processes. Depending on primary inputs, the PLC can control the motor to rotate forwards, backwards, or to stop. The uncertainty characterization of this ladder program for A1, A2, and A3 are 0.4731, 0.6559, and 0.5198, respectively.

The fifth application is a ladder program embedded in the responder of an entertainment contest. All contest participants will have a response button. The PLC will ensure that the winner is the first to respond and no other participant presses the button before the winner. The uncertainty characterization of this ladder program for A1, A2, and A3 are 0.4657, 0.6928, and 0.5642, respectively.

IX. Conclusion

In this paper, we proposed a new method to model a PLC ladder program and three algorithms to characterize uncertainty for the PLC ladder program. The first algorithm considers errors at the input sampling stage. We characterize the uncertainty of the ladder program by sampling probability and static probability of the primary input. The second method considers also the impact of the processing deviation on the primary input. We characterize the uncertainty by the failure probability ε of the unit and the probability that the output of the step would be wrong when the unit turns out to be wrong. The third method is based on topological sort. We transfer a ladder step into many operation units and sort them topologically. We then process the units one by one, and obtain the final uncertainty characterization of the last unit. The first algorithm is the simplest, while the last algorithm considers more possibilities. They can be chosen according to the possibilities and the values of uncertainties in a real application. The experiments demonstrate the application of the three algorithms.

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," in *Proc of Systems, Man and Cybernetics, IEEE Conference on*, Nashvill, TN, USA, October 2000, pp. 2449–2454.

[2] H.X.Willems, "Compact timed automata for PLC programs," University of Nijmegen, Computing Science Institute," Technical Report CSI-R9925, 1999.

[3] A. Mader and H. Wupper, "Timed automaton models for simple programmable logic controllers," in *Proc of Euromicro Conference on Real-Time Systems*, York, UK, June 1999.

[4] N. Bauer, S. Engell, R. Huuck, S. Lohmann, B. Lukoschus, M. Remelhe, and O. Stursberg, *Verification of PLC Programs Given as Sequential Function Charts*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004, vol. 3147/2004, ch. Verification, pp. 517–540.

[5] H.-M. Hanisch, J. Thieme, A. Luder, and O. Wienhold, "Modeling of PLC behaviour by means of timed net condition/event systems," in *IEEE Int. Symposium on Emerging Technologies and Factory Automation (EFTA '97)*, 1997, pp. 361–369.

[6] M. B. Younis and G. Frey, "Formalization of existing PLC programs: A survey," in *Proc. Computational Engineering in Systems Applications (CESA)*, 2003.

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

[8] M. Rausch and B. H. Krogh, "Formal verification of PLC programs," in *Proc. American Control Conference*, 1998.

[9] T. L. Johnson, "Improving automation software dependability: A role for formal methods?" *Control Engineering Practice*, vol. 15, no. 11, pp. 1403 – 1415, 2007.

[10] *Programmable Controllers - Programming Languages, IEC 61131-3. Ed. 2.0.*, IEC(International Electrotechnical Commission) Std., Rev. 2.0, 2003.

[11] A. K. J.Monteiro, S.Devadas and J.K.White, "Estimation of average switching activity in combinational logic circuits using symbolic simulation," *IEEE Transactions on CAD*, vol. 16, no. 1, pp. 668 – 670, 1997.

[12] K. Parker and E. Mccluskey, "Analysis of logic circuits with faults using input signal probabilities," *IEEE Transactions on computer*, p. 321, jun. 1995.

[13] K. Parker and E. McCluskey, "Probabilistic treatment of general combinational networks," *IEEE Transactions on CAD*, vol. C-24, no. 6, pp. 121–127, 1975.

[14] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco., "Estimate of signal probability in combinational logic networks," in *First European Test Conference*, 1989, pp. 132–138.

[15] M. R. Choudhury and K. Mohanram, "Reliability analysis of logic circuits," *IEEE transactions on CAD*, vol. 28, no. 1, 2009.