INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 10, NO. 1, MARCH 2005, 94-106

Reconfiguration of Ladder Logic Diagrams with Dynamic Input Sets

Jing LIU and Houshang DARABI

Abstract - **This paper addresses the design process of programmable logic controller (PLC) ladder logic diagram (LLD) when the PLC input channels are subject to failure/repair. Given a traditional LLD and its feasible behavior, a finite automata (FA) supervisory control model of the LLD is developed. Partial observation theory is used to extend the generated FA model to a mega-controller. The mega-controller can modify (reconfigure) the control strategies when the FA events are partially observed. The mega-controller model is then converted to an LLD that is tolerable with respect to the availability of the PLC input channels. The developed framework can be used in industrial systems that PLCs have many input channels and these input data to PLC could be unavailable temporarily due to communication network congestion, sensor failure, disconnected input devices, etc. We illustrate the proposed theory by an example.**

***Index Terms*—Control Reconfiguration, Finite Automata Analysis, and Ladder Logic Diagram.**

1. INTRODUCTION

Programmable Logic Controllers (PLC) are widely used in shop floor automation, and Ladder Logic Diagram (LLD) is the most popular programming language of PLCs. An LLD based program continuously reads the state of the system that it is controlling. The system state is determined by the plant information that the (LLD based) controller receives through its input channels. In the current design process of LLD, it is assumed that these channels are always available and that they can properly report their corresponding information to the controller. In practice, this assumption is not necessarily true. In fact, when the number of plant automated components increases, it is likely that the information regarding the states of these components is partially reported to the controller. In this case we say that the controller has a partial observation of its plant (i.e. controlled system). Reasons for incomplete observation include but not limited to: sensor failure, communication line congestions and delays, long scan time, low analog to digital conversion precision of the PLC input module in collecting the input signals, power outage in the input module layer, excessive input channels, and the failure of human in triggering PLC online commands on time. This paper intends to extend the traditional existing Ladder Logic Diagram (LLD) to a *reconfigurable* LLD which is capable of adapting its logic on the fly according to observable plant information. To achieve this goal, a formal method for observation analysis of LLD is essential.

Theoretically speaking, the availability of observation means can be considered as a partial observation problem. For the systems modeled by Finite automata (FA), partial observation problem was investigated by [2, 3] originally. These works determined the conditions under which a given set of observation means can be used for a proper control. However, these works had a fixed view of the observation means and the proposed algorithms could not be applied in situations that the availability of means was changing. Reference [4] presented a switching theory to dynamically reconfigure a DES controller under dynamic partial observation. Reference [5] gave a broader view of this problem by developing new dynamic observation theory-finite time observation policies, and reconfiguration strategies. This work introduced more tolerable control policies compared to the other works in the literature. The results derived in [5] provide the theoretical guidelines of the research presented in the current paper. However, the theory of [5] cannot be directly applied to the LLD design. The following comparison of LLD and finite automata controllers presents the main reason behind this claim:

1. The FA controller is a permissive controller, i.e., it allows events to happen but it does not force them. However, LLD controller forces the output to activate or de-activate.
2. For the FA controller, in one unit of time, at most one event can occur. But in LLD, in one scan time (one unit of time), multiple input and output changes may happen.
3. Usually, there is not a one-to-one relationship between FA controller events and PLC I/O signal changes. Therefore, there might be a need to translate PLC input signals to FA controller events and the controller feedback to PLC output signals.
4. The feedback of the FA controller applies to the events themselves, but the LLD is translated to the control actions only on the outputs, not the inputs. The outputs can affect the plant behavior which in return determines the value of the inputs.

Studies have been carried out on the analysis and verification of LLDs [6-12]. The most popular approach proposed in these works is to develop a controller with one of the formal controller construction techniques, such as Finite Automata [1, 13] and Petri nets [14-22], and convert the resulting model to LLD. The constructed controller is first analyzed and then a conversion procedure generates an equivalent LLD of the controller. Therefore, the final LLD is guaranteed to carry the same control properties as of the constructed model. The FA controller construction technique can generate a maximally permissive controller from a DES plant and control specifications. The conversion from an FA controller to LLD was addressed by [6, 22], which is basically the conversion of a state-transition diagram to LLD [6, 23-24]. While this

Manuscript received December 1, 2004, revised February 15, 2005.
The authors are with the Department of Mechanical and Industrial Engineering, 842 W. Taylor Street, University of Illinois at Chicago, IL 60607, USA (e-mail: jliu4@uic.edu and hdarabi@uic.edu).

conversion is algorithmically true, the resulting LLD is usually not implementable, i.e., it cannot be directly run on PLC. The work [25] converts an FA controller to a ladder logic diagram while assuring its implementability as a PLC program. However, these approaches have been rarely used by practitioners [26]. In fact none of these researches has considered a real-world model of LLD and they are all limited to a simplified LLD. This has prevented the industry from using their techniques.

This paper intends to integrate the reconfigurability into the design of LLDs. It presents the observation analysis theory that can systematically and comprehensively analyze the situations caused by the unavailability of input information, and proposes a proactive measure to prevent out-of-control states. Conversion algorithms are proposed in order to bridge the gap between LLD and FA controllers as mentioned above.

Figure 1 shows the outline of the steps introduced in this paper: (1) a given LLD is first converted to a finite automaton with and a special feedback function Φ'; (2) the feedback function Φ' is then translated to a regular (Ramadge-Wonham) feedback function, say Φ; (3) the observation analysis is performed on the FA controller and the result is a new controller called mega-controller; (4) the implementability of the mega-controller is verified by solving an Integer Linear Programming (ILP) model and an implementable mega-controller, called extended mega-controller is generated; (5) the extended mega-controller is converted to an LLD, called reconfigurable LLD, that integrates the reconfigurability to the original LLD. The paper is organized as follows. Section 2 reviews the theory concerning supervisory control based on finite automata. It also reviews the existing works on observation analysis and control reconfiguration of discrete event systems (DES). Section 3 presents the translation from a LLD to a finite automaton and the generation of feedback on outputs. Section 4 addresses the translation from feedback on outputs to feedback on events. Section 5 discusses the observation analysis on the FA controller. Section 6 discusses the reconfigurable LLD implementation. Section 7 presents an illustrative example. Concluding remarks and potential future research issues are given in Section 8.

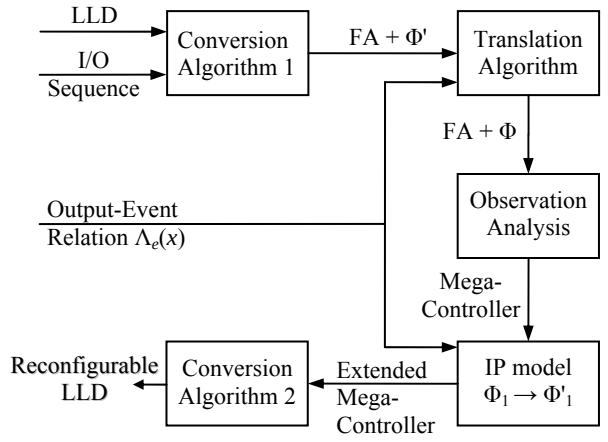


Fig. 1. Methodology Flow Chart.

2. PRELIMINARIES

According to [1], a DES can be modeled by a finite automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is the set of states, Σ is the finite set of events (which can be partitioned into two disjoint subsets, controllable events set Σ_c, and uncontrollable events set Σ_{uc}), $\delta : Q \times \Sigma \rightarrow Q$ is the transition functions, q_0 is the initial state, and $Q_m \subseteq Q$ is the set of marked states. Suppose that Σ^* is the set of finite length strings of Σ. The extension of δ on Σ^* is defined as $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$ where $s\sigma \in \Sigma^*$. G is said to be blocking if $\overline{L_m} \subset L$, and nonblocking if $\overline{L_m} = L$, where $L = \{s \in \Sigma^* : \delta(q_0, s) \text{ is defined}\}$ is the language generated by G and $\overline{L_m} = \{s : \exists t \in \Sigma^* \text{ such that } st \in L \text{ and } \delta(q_0, st) \in Q_m\}$. G models the uncontrolled plant. The supervisor **S** (or controller) interacts with G in a closed loop manner (Fig. 2), and it assures that the plant does not violate a given set of control specifications. Observation means report the occurrence of events in G to **S.** Mathematically, $\mathbf{S} = (S, \Phi)$, where $S = (X, \Sigma, f, x_0, X_m)$ is a deterministic automaton with state set X, initial state x_0, a marked subset $X_m \subseteq X$, and transition function $f : X \times \Sigma \rightarrow X$; $\Phi : X \times \Sigma \rightarrow \{1, 0, dc\}$ is the feedback function. Similar to the extension of δ on the strings of Σ^*, function f can also be extended on these strings. $\Phi(x, \sigma) = 1$ (0) indicates that the control action at state x is to enable (disable) event σ. "dc" is an abbreviation for "don't care", which implies that the enabling or disabling of σ at x doesn't affect the behavior of G. $\Gamma : X \rightarrow 2^{\Sigma}$ is the active event function, and is defined as $\Gamma(x) = \{e : f(x, e) \text{ is defined}\}$ for all $x \in X$. In other words, $\Gamma(x)$ includes all the events that are enabled by **S** at state x. The controlled plant language, so called coupled language, is shown by $K = L(\mathbf{S}/G)$.

Assume that for every event there is an observation means to report its occurrence to the controller, repair and failure of the observation means are modeled by two sets of events, R and B, respectively. Suppose that $b_\sigma \in B$, $r_\sigma \in R$ are the breakdown and repair events of the

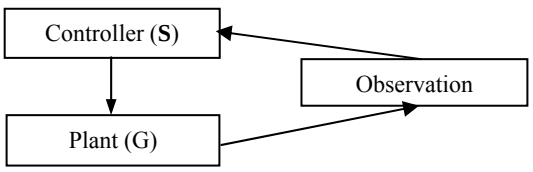


Fig. 2. Control Loop.

observation means that reports the occurrence of event $\sigma \in \Sigma$. We use the notation $y = \bar{X}\_SS$ to show a state of a mega-controller. In this notation $\bar{X} \subseteq X$ and $SS = (ss_{\sigma_1}, ss_{\sigma_2}, \cdots, ss_{\sigma_i}, \cdots, ss_{\sigma_n})$. Here ss_{σ_i} is the observation means status of event σ_i in $\Sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_i, \cdots, \sigma_n\}$, and ss_{σ_i} = 0(1) if the observation means reporting event σ_i is failed (working).

Given the maximally permissive controller $\mathbf{S} = (S, \Phi)$ (recall that $S = (X, \Sigma, f, x_0, X_m)$), the mega-controller automaton $\Theta = (Y, E, g, y_0, Y_m)$ can be constructed as follows:

1- Let $E = \Sigma \bigcup B \bigcup R$, $y = \bar{X}\_SS$, $y_0 = x_0\_11\cdots1$, $Y = \{\}$ and $A = \{y_0\}$. Let $y = y_0$. Define $Y_m = \{y_0\}$.

2- For every $\sigma \in \Sigma$, define the feedback function as:

$$\tilde{\Phi}(y,\sigma) = \begin{cases} 0, & \text{if } \exists\, x \in \bar{X} : \Phi(x,\sigma) = 0; \\ 1, & \text{if } \exists\, x \in \bar{X} : \Phi(x,\sigma) = 1 \\ & \text{and } \nexists\, x' \in \bar{X} : \Phi(x',\sigma) = 0; \\ dc, & \text{otherwise.} \end{cases}$$

$$\tilde{\Phi}(y,b_\sigma) = \begin{cases} 1, & \text{if } ss_\sigma = 1; \\ dc, & \text{otherwise.} \end{cases} \text{ and}$$

$$\tilde{\Phi}(y,r_\sigma) = \begin{cases} 1, & \text{if } ss_\sigma = 0; \\ dc, & \text{otherwise.} \end{cases}.$$

3- For every $\sigma \in \bigcup_{x \in \bar{X}} \Gamma(x)$ such that $ss_\sigma = 1$,

3-1 Create the relationship $g(y,\sigma) = y'$ where y' is calculated by the following steps:

3-1-1 $\bar{X}_1 = \{ f(x,\sigma) \mid x \in \bar{X}, f(x,\sigma) \text{ is defined.} \}$;

3-1-2 $\bar{X}_2 = \Big\{ f(x,t) \mid x \in \bar{X}_1,$

$t = \{e \mid e \in \Sigma \text{ and } ss_e = 0\}^*, f(x,t) \text{ is defined.} \Big\}$;

3-1-3 $\bar{X}' = \bar{X}_1 \bigcup \bar{X}_2$, $SS' = SS$ and $y' := \bar{X}'\_SS'$;

3-2 If y' is neither in *Y* nor in *A*, add it to *A*.

4- For every $\sigma \in \Sigma$ such that $ss_\sigma = 0$,

4-1 Create the relationship $g(y,b_\sigma) = y'$ where $y' = \bar{X}'\_SS'$, $ss_{\sigma'} = \begin{cases} 0, & \text{if } \sigma' = \sigma \\ ss_\sigma, & \text{otherwise} \end{cases}$, and

$\bar{X}' = \Big\{ f(x,t) \mid x \in \bar{X},$

$\{t = \{\sigma' \mid \sigma' \in \Sigma, ss_{\sigma'} = 0 \text{ or } \sigma' = \sigma\}^*\Big\}$.

4-2 If y' is neither in *Y* nor in *A*, add it to *A*.

5- For every $\sigma \in \Sigma$ such that $ss_\sigma = 1$,

5-1 Create the relationship $g(y,\sigma) = y'$ where $g(y,r_\sigma) = y'$, $y' := \bar{X}\_SS'$ and $ss_{\sigma'} = \begin{cases} 1, & \text{if } \sigma' = \sigma; \\ ss_\sigma, & \text{otherwise} \end{cases}$. If y' is not in *Y*, add that to *Y*;

5-2 If y' is neither in *Y* nor in *A*, add it to *A*.

6- Remove *y* from *A*. Add *y* to *Y*. If *A* is empty then go to step 7, otherwise select one of the elements of *A*, call it *y* and go to step 2.

7- Revise the generated automaton by performing a trimming process (see [1] for the details of the trimming process) and consider $\Theta = (Y, E, g, y_0, Y_m)$ to be the revised automaton.

In short, in the trimming process all the states in *Y* that are dangling (no string of events exists from them to y_0) are removed. The removal is done by revising the feedback function Φ' (disabling appropriate controllable events) in non-dangling states of *Y* such that the strings that lead to dangling states are disabled.

3. CONVERSION FROM LLD TO FA

The objective of this section is to introduce a method for converting a given LLD to a finite automaton while preserving the behavioral properties of the LLD. This conversion is labeled by Algorithm 1 in Fig. 1. The resulting FA is used to design a fault tolerant control policy that increases the amount of time that the PLC, with partial observation of the plant events, can properly control the plant. We assume that the following information is known before the conversion proceeds: a) PLC outputs; b) PLC inputs, and c) The complete sequence of possible input values and their corresponding output activation/deactivation commands generated by PLC per scan.

To convert the LLD to an FA model, we need to capture the behavior of the LLD. The PLC communicates with the plant through input and output channels. The change of input and output status reflects the control activities of the PLC, and also reflects the state of the system. For small systems with less than one hundred of rungs of LLD, the input signals are manually entered into the LLD in their order of occurrence and the resulting outputs are recorded. For the complex systems, with the aid of a PLC simulator (emulator), the input and output status is written into a database at each scan.

We assume that the inputs and outputs of PLC are Boolean. Define $I = \{i_1, i_2, \cdots, i_k, \cdots, i_m\}$ as the set of inputs, where i_k is the k^{th} input, and m is the total number of inputs. Similarly let $O = \{o_1, o_2, \cdots, o_k, \cdots, o_n\}$ be the set of outputs, where o_k is the k^{th} output, and n is the total number of outputs. Also define V_j^{IO} as the input and output status vector at the j^{th} scan with size of $(m+n)$, $j \in [0,N]$ and N is the total number of scans recorded. We assume that the selected N scans cover all the control paths that are executed by the LLD when it controls the plant. In fact

V_j^{IO} is a binary vector in which each digit shows the status of an input or output. Let V_j^I (V_j^O) show the input (output) status vector at the j^{th} scan with size of m (n). V_j^I (V_j^O) is a binary vector in which each digit shows the status of an input (output).

The converted FA is shown by $S=(X,\Sigma,f,x_0,X_m)$. The input's status changes (from on to off, or from off to on) are considered as events, and thus the possible events are $\{i_1^+,i_1^-,i_2^+,i_2^-,\cdots,i_k^+,i_k^-,\cdots,i_m^+,i_m^-\}$ where i_k^+ is the off-to-on (from 0 to 1) event of the k^{th} input and i_k^- is the on-to-off (from 1 to 0) event of the k^{th} input. Not the events listed above may be included in the event set Σ. Define $(V)_k$ as the k^{th} element of vector V. Before discussing the conversion algorithm, we introduce the following two functions. *Binary representation of a vector* V, shown by $\beta(V)$, is a binary number where content of its k^{th} position bit equals to the value of the k^{th} element of V. *Indexes of non-zero elements of a vector V* shown by the set $Ind^{\bar{0}}(V)$.

Conversion Algorithm 1: Conversion from LLD to FA

Begin

1. Let $x_0=\beta(V_0^{IO})$, $\Sigma=\varnothing$, $X=\{x_0\}$, $X_m=\varnothing$, and $j=1$.
2. Compare V_j^{IO} with V_{j-1}^{IO},
   a) If $V_j^{IO}=V_{j-1}^{IO}$, go to step 4.
   b) If V_j^I is different from V_{j-1}^I, add $\beta(V_j^{IO})$ to X, and let $\Delta V_j^I=V_j^I-V_{j-1}^I$,
      For $k=1$ to m
         If $(\Delta V_j^I)_k=1$, add i_k^+ to Σ, let $f\left(\beta(V_{j-1}^{IO}),i_k^+\right)=\beta(V_j^{IO})$;
         If $(\Delta V_j^I)_k=-1$, add i_k^- to Σ, let $f\left(\beta(V_{j-1}^{IO}),i_k^-\right)=\beta(V_j^{IO})$.
      End For
   c) Otherwise, add ω to Σ, $\beta(V_j^{IO})$ to X, let $f\left(\beta(V_{j-1}^{IO}),\omega\right)=\beta(V_j^{IO})$.
3. The feedback function is defined as

$$\Phi'\left(\beta(V_j^{IO}),o_k\right)=\begin{cases}1 & \text{if } (V_j^O)_k=1\\ 0 & \text{if } (V_j^O)_k=0\end{cases}\qquad \forall k\in[1,n].$$

4. Let $j=j+1$. If $j\le N$ go to step 2, otherwise go to step 5.
5. Let $X_m=X$.

End.

From the algorithm steps it is trivial that the maximum number of FA states generated by the algorithm is $2^{(m+n)}$, that is $|X|\le 2^{(m+n)}$.

Proposition 1. If the selected scans cover all possible control paths that the PLC LLD can experience, the converted FA with the feedback function Φ' and the LLD both generate the same sequences of input and output vectors.

Proof: Since the selected scans cover all possible control paths that the PLC LLD can experience, by the algorithm construction, starting from the initial state, for every change in the input vector, both the LLD and FA yield the same output vectors. Also event e exists in the LLD control path if and only if it exists in the FA control path (recall that event e is a member of $\{i_1^+,i_1^-,i_2^+,i_2^-,\cdots,i_k^+,i_k^-,\cdots,i_m^+,i_m^-,\omega\}$ that is generated in the FA based on its existence in the LLD). Taking into account that event ω represents all the changes in the outputs that are accompanied by no input changes in the same scan, and given the fact that the PLC output information are included in the state information of the FA model, the two LLD and FA controllers follow the same sequences of input and output vectors.■

In the following we discuss some the issues related to the above algorithm. First issue is the concept of *virtual event.* We have created a virtual event, shown by ω, to model the changes of the internal relays which cause the output changes (while no input changes occur). Because the change of internal relay status is always observable by the PLC, we can actually ignore the event ω in the observation analysis. The second issue is the marked state set of the generated finite automaton. Without loss of generality we set $X_m=\{x_0\}$, i.e. we assume a cyclic behavior for the controlled plant. The third issue is the feedback function $\Phi':X\times O\to\{1,0\}$ that is applied to the output set O. This function does not satisfy the requirements of supervisory control theory as it is not defined based on the FA events (that are PLC input changes and event ω). In fact, we need a feedback function in form of $\Phi:X\times\Sigma\to\{1,0,dc\}$ where $\Sigma\subseteq\{i_1^+,i_1^-,i_2^+,i_2^-,\cdots,i_k^+,i_k^-,\cdots,i_m^+,i_m^-,\omega\}$. In the next section, we translate Φ' into Φ to resolve this problem.

4. TRANSLATION FROM $\Phi':X\times O\to\{1,0\}$ TO $\Phi:X\times\Sigma\to\{1,0,dc\}$

The translation algorithm of this section explains the details of the box labeled by "Translation Algorithm" in Fig. 1. In order to translate Φ' to Φ, we need to find the relationship between the output set O and the event set Σ. The outputs and events (inputs) are not one to one related because multiple outputs may have effect on the occurrence of one event, and similarly, one output may affect the occurrence of multiple events. Furthermore, the relationships between events and outputs are state-dependent. We assume that such relationships are available and can be defined. To mathematically model an event-

output relationship, we define o_k^+ (o_k^-) to denote output o_k is on (off). Using this notation and logical operators "AND" and "OR" one can show the relationships. For example, suppose that if o_1 is on, or o_2 is on and o_3 is off, then event i_1^+ is possible to happen. Therefore one relationship between event i_1^+ and outputs o_1, o_2 and o_3 can be written as $\left(o_1^+\right)$ OR $\left(o_2^+ \text{ AND } o_3^-\right)$.

We use a matrix to show the event-output relationships in every state. For state *x,* each row of its event-output relationship matrix shows the "AND" relation among the outputs, and the relation among different rows is of "OR" type. This means that if all the output conditions in at least one row of this matrix hold then the event can happen. We use $\Lambda_e(x)$ to show the event-output relationship matrix for event *e* at state *x*. The size of $\Lambda_e(x)$ is $r_e(x)\times n$ where *n* is the number of outputs, and $r_e(x)$ is the number of rows of $\Lambda_e(x)$. $\left[\Lambda_e(x)\right]_{a,b}$ corresponds to the status of o_b in the a^{th} row vector, and its content is 0, 1 or *dc* for o_b off, o_b on, or o_b does not contribute to the occurrence of event *e* respectively.

In the traditional supervisory control theory it is assumed that the controllability of an event is fixed. However for the supervisory controller generated from LLD this might not be true. This means that for a given event, one can disable the event in some occasions but not others. Accordingly, we propose a concept called *dynamic controllability* to describe the change in the event controllability over time. An event is controllable as long as there is at least one combination of outputs that can cause that event to be disabled. If no such combination exists then the event becomes uncontrollable. Since for every event, the output conditions change from one state to another, the controllability of events is fixed in every state but it could change when the controller state changes.

Suppose that $\Sigma_c(x)$ is the set of *controllable event set at state* $x\in \mathrm{X}$ and $\Sigma_{uc}(x)$ is the set of *uncontrollable events at state* $x\in \mathrm{X}$. If $\Lambda_e(x)$ is not defined (it has no rows), then event $e\in\Sigma$ is uncontrollable at state *x*, i.e. $e\in\Sigma_{uc}(x)$; otherwise *e* it is controllable at state *x*. The feedback function $\Phi: X\times\Sigma\to\{1,0,dc\}$ is defined as follows,

$$\Phi(x,e)=\begin{cases}1 & \text{if } f(x,e) \text{ is defined; (Cases 1 and 2)}\\ 0 & \forall a\in[1,r_e(x)],\exists\left[\Lambda_e(x)\right]_{a,b}\neq dc \\ & \text{and } \left[\Lambda_e(x)\right]_{a,b}\neq\Phi'(x,o_b);\ \text{(Case 5)}\\ dc & \text{otherwise.(Cases 3 and 4)}\end{cases}$$

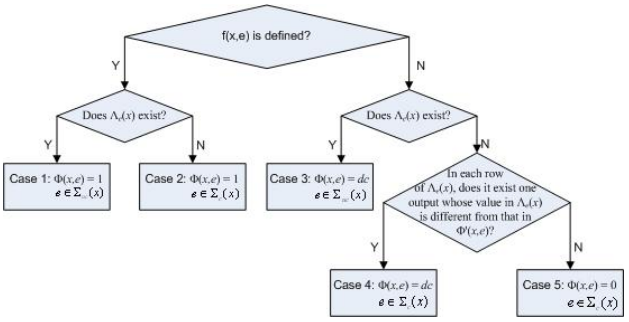


Fig. 3. Feedback Function Generation Cases.

Five cases in the feedback function generation are shown in Fig. 3. Further explanations of these cases are given below.

Case 1: because $f(x,e)$ is defined, event *e* must be enabled at state *x*; because $\Lambda_e(x)$ has no rows, therefore $e\in\Sigma_{uc}(x)$.

Case 2: because $f(x,e)$ is defined, event *e* must be enabled at state *x*; because $\Lambda_e(x)$ has at least one row, therefore $e\in\Sigma_c(x)$.

Case 3: because $f(x,e)$ is not defined, event *e* is not enabled at state *x*; because $\Lambda_e(x)$ is not defined, $e\in\Sigma_{uc}(x)$. For an uncontrollable event, if it is not enabled, the feedback on it can only be "don't care".

Case 4: because $f(x,e)$ is not defined, event *e* is not enabled at state *x*; because in each row of $\Lambda_e(x)$, there is at least one output o_b such that $\left[\Lambda_e(x)\right]_{a,b}\neq dc$ and $\left[\Lambda_e(x)\right]_{a,b}\neq\Phi'(x,o_b)$, event *e* is not disabled; because $\Lambda_e(x)$ has at least one row, $e\in\Sigma_c(x)$. For a non-enabled and non-disabled controllable event, the feedback can only be "don't care".

Case 5: because $f(x,e)$ is not defined, event *e* is not enabled at state *x*; because in each row of $\Lambda_e(x)$, there is no o_b such that $\left[\Lambda_e(x)\right]_{a,b}\neq dc$ and $\left[\Lambda_e(x)\right]_{a,b}\neq\Phi'(x,o_b)$, event *e* is disabled; because $\Lambda_e(x)$ is defined, $e\in\Sigma_c(x)$. In this case the current state of outputs (based on feedback function Φ' does not match with any of the output conditions shown by the rows of $\Lambda_e(x)$.

5. OBSERVATION ANALYSIS ON FA CONTROLLER

The converted FA model *S* and the feedback function Φ form an FA supervisory controller. The next step is to perform the observation analysis (shown by the box labeled "Observation Analysis" in Fig. 1). The observation analysis can provide the minimum sufficient input information that

a PLC needs to maintain the right control, predict the consequences caused by the lack of input information to the PLC, and also discover the proactive steps to prevent the undesirable consequences. The run-time reconfiguration strategies such as state aggregation, state disaggregation, and feedback adjustment techniques, adjust the control actions dynamically so that the controller can survive the sudden changes of input channel availability and (if possible at all) properly control the plant until the failed input channels are recovered.

As mentioned in the preliminaries, the reconfiguration strategies are implemented through the mega-controller $\Theta = (Y, E, g, y_0, Y_m)$ and the feedback function $\tilde{\Phi}$. When the input channel status change (failure and repair of channels), the mega-controller determines the right feedback for the events. The right feedback in some cases could generate a more restrictive controller until the necessary input channels become available. We define a new feedback function $\Phi_1 : Y \times X \times \Sigma \to \{0, 1, dc\}$ based on the feedback function of the mega-controller.

$$\Phi_1(y,x,e) = \begin{cases} 1 & x \in y, \tilde{\Phi}(y,e) = 1, \Phi(x,e) \neq dc \\ 0 & x \in y, \tilde{\Phi}(y,e) = 0, \Phi(x,e) \neq dc \\ dc & x \in y, \tilde{\Phi}(y,e) = dc \end{cases},$$

and $\Phi_1(y,x,e)$ is not defined if $x \notin y$. This function will be used in next section.

6. Implementation of Observation Analysis on LLD

The mega-controller Θ has a feedback function that is defined on its event set *E* which contains PLC input events and failure/repair events of input channels. However this function can not be implemented in an LLD, as the LLD feedback must be based on its outputs. For this reason we convert the mega-controller automaton to an extended mega-controller automaton. Before discussing the algorithm for this conversion, we would like to discuss an issue related to the implementation of output commands of LLD, that could arise during the construction of Θ. As mentioned before, in the construction of Θ, we might aggregate the states of *X* and generate some combined states, we could disaggregate a combined state to smaller combined or simple states, and we could adjust the feedback rules (switch between 0 and 1 rules) in the combined states to resolve the conflicts. During these operations, we assure the resulting feedback function of the combined states does not ask for any conflicting policy that is, enabling and disabling an event at the same time. However, guaranteeing a conflict free feedback rule does not necessarily mean that the corresponding output commands can be generated consistently. The following proposition shows this fact.

Proposition 2. If a control strategy (state aggregation, disaggregation and feedback adjustment) can be implemented on the controller with FA model *S* and feedback function Φ, it may not be implementable on the LLD.

Proof: we show the proof by the contradiction.

Example 1. The following list is a partial representation of the feedback function Φ' of a control program.

Table 1. Partial representation of Φ'.

	o_1	o_2
x_1	1	0
x_2	0	0
…	…	…

Suppose that $\Lambda_{e_1}(x_1) = [dc \quad 1]$, $\Lambda_{e_1}(x_2) = [1 \quad 1]$, $\Lambda_{e_2}(x_1) = \Lambda_{e_2}(x_2) = [dc \quad 0]$, and $f(x_1, e_2) = x_2$, then the partial representation of Φ is shown in Table 2.

Table 2. Partial representation of Φ.

	e_1	e_2
x_1	0	1
x_2	0	dc
…	…	…

We see that for Φ, states x_1 and x_2 are conflict free, therefore they can be combined, but for Φ', because $\Phi'(x_1, o_1) = 1$ but $\Phi'(x_2, o_1) = 0$, this strategy is not applicable on the LLD. ■

We notice that the observation analysis strategies are designed based on Φ, however their implementation on LLD is performed by the information from Φ'. As shown in the previous example, such implementation is not always feasible. Therefore we might need to modify our observation analysis strategies if they cannot be implemented on the LLD. In the following, we present an algebraic method to conduct the required modifications.

For a given row, say *b*, of the relationship matrix $\Lambda_e(x)$, we say the output $o_k \in O$ is *reversely activated* if (this output is activated at state *x* and $[\Lambda_e(x)]_{b,k} = 0$) or (the output is deactivated at state *x* and $[\Lambda_e(x)]_{b,k} = 1$). Accordingly $o_k \in O$ is said to be *positively activated* if (this output is activated and $[\Lambda_e(x)]_{b,k} = 1$) or (this output is deactivated and $[\Lambda_e(x)]_{b,k} = 0$) or ($[\Lambda_e(x)]_{b,k} = dc$).

The following rules in implementing feedback adjustments of outputs are considered. If for event $e \in \Sigma$, $y \in Y$ and $x \in X$, $\Phi(x,e) = 1$ but $\Phi_1(y,x,e) = 0$, then in any of the row vectors in $\Lambda_e(x)$ at least one output has to be reversely activated. If $\Phi_1(y,x,e) = 1$ then all the outputs of at least one row vector in $\Lambda_e(x)$ have to be positively activated. We can see that there are multiple ways of enabling event *e* at state *x* if there are multiple row vectors

in $\Lambda_e(x)$, and there might be multiple ways of disabling event *e* at state *x* if there are multiple columns in $\Lambda_e(x)$.

In general, if a consistent feedback can be found for each output, then this control strategy is implementable, otherwise, it is not. Here we introduce a mathematical model that can be used to find the values of outputs that can support the new feedback function Φ_1. The objective of this model is to find at least one feasible solution. Therefore if no feasible solution is found by solving the model, it means no consistent set of output values exists that can implement the new feedback function Φ_1.

Based on the applied observation analysis strategies, some of the states of the mega-controller are combined states. Suppose that v be such a state. We recall that $v \subseteq X$ and $|v| \geq 1$. Since the feedback rule for all states of Θ that are not combined does not change, i.e. $\Phi_1(x,x,e) = \Phi(x,e)$ for $x \in X$ and $e \in \Sigma$, we do not need to consider the modification of the output commands for these states. Therefore we apply the following model to the combined states of Θ only. We assume that the new feedback function $\Phi_1(v,x,e)$ is given for every $x \in v$ and $e \in \Sigma$. The goal is to find the values of outputs $o_1, o_2, \cdots, o_n$ such that the rules (enable, disable, dc) of $\Phi_1(v,x,e)$ can be implemented. We propose the following integer programming (IP) model to solve this problem.

Mathematical Model

$$\text{Min} \quad \sum_{i=1}^{n} z_i$$

Subject to

(1)
$$y_{x,e,b} \geq 1 - M\{ \sum_{\substack{1 \leq i \leq n \\ [\Lambda_e(x)]_{b,i} \neq dc}} (1 - 2[\Lambda_e(x)]_{b,i})(z_i - [\Lambda_e(x)]_{b,i}) \}$$

(2)
$$\sum_{\substack{x \in v, e \in \Sigma, 1 \leq b \leq r_e(x) \\ [\Lambda_e(x)]_{b,i} = 0}} y_{x,e,b} \leq M(1 - z_i) \qquad 1 \leq i \leq n$$

(3)
$$\sum_{\substack{x \in v, e \in \Sigma, 1 \leq b \leq r_e(x) \\ [\Lambda_e(x)]_{b,i} = 1}} y_{x,e,b} \leq M z_i \qquad 1 \leq i \leq n$$

(4)
$$\sum_{1 \leq b \leq r_e(x)} y_{x,e,b} = 0 \qquad x \in v,$$
$$e \in \Sigma \text{ and } \Phi_1(v,x,e) = 0$$

(5)
$$\sum_{1 \leq b \leq r_e(x)} y_{x,e,b} \geq 1 \qquad x \in v,$$
$$e \in \Sigma \text{ and } \Phi_1(v,x,e) = 1$$

$$y_{x,e,b} = 0,1$$
$$x \in v, e \in \Sigma \text{ and } 1 \leq b \leq r_e(x)$$
$$z_i = 0,1 \qquad 1 \leq i \leq n$$

Model Discussion: In this model, the definitions of variables are:

z_i =1(0) if output o_i is activated (deactivated), $1 \leq i \leq n$.

$y_{x,e,b}$ =1 if all the output conditions of the b^{th} row of $\Lambda_e(x)$ are satisfied, and $y_{x,e,b}$ =0 otherwise.

M is a large number. All other parameters have already been defined. Constraint set (1) forces variable $y_{x,e,b}$ to take value 1 if necessary. If all the conditions of the b^{th} row of $\Lambda_e(x)$ are satisfied then the expression in {} is set to 0 and therefore the right hand side of this constraint becomes 1. Consequently $y_{x,e,b}$ is forced to take the value of 1. If at least one of the conditions of the b^{th} row of $\Lambda_e(x)$ is not satisfied then the related constraint allows $y_{x,e,b}$ to be greater than $-\infty$. Constraint sets (2) and (3) ensure that variables $y_{x,e,b}$'s are set to zero if necessary. If output o_i is decided to be activated ($z_i = 1$), then all the rows in the relationship matrixes $\Lambda_e(x)$, $e \in \Sigma$ that require o_i to be deactivated are not satisfied. Therefore the corresponding $y_{x,e,b}$ of these rows must be set to zero. This is done by constraint set (3). Constraint set (4) has the similar mechanism for the case that output o_i is decided to be deactivated ($z_i = 0$). Constraint set (5) ensures that no row in the relationship matrix $\Lambda_e(x)$ is satisfied if $\Phi_1(v,x,e) = 0$. This means that the selected configuration of outputs disables event *e*. Constraint set (6) guarantees that at least one set of conditions shown by $\Lambda_e(x)$ for enabling event *e* is satisfied when $\Phi_1(v,x,e) = 1$. The objective function does not have any specific role in the formulation. The current function minimizes the number of output activation commands that the PLC sends to the plant after the reconfiguration policy is implemented. However, one can use other objective functions depending on the design goals. As far as we are concerned if the constraints are satisfied, the correct implementation of $\Phi_1(v,x,e)$ is guaranteed.

Example 2.

Suppose that $u = \{x_1, x_2\}$, $\Sigma = \{e_1, e_2, e_3\}$, $O = \{o_1, o_2, o_3, o_4\}$, and the output-event relation matrixes are

$$\Lambda_{e_1}(x_1) = \begin{bmatrix} 1 & dc & dc & dc \\ dc & 1 & 1 & dc \\ 1 & 0 & 1 & dc \end{bmatrix},$$

$$\Lambda_{e_2}(x_1) = \begin{bmatrix} 0 & dc & dc & dc \\ 1 & 0 & dc & 1 \end{bmatrix},$$

$$\Lambda_{e_3}(x_1) = \begin{bmatrix} dc & 0 & dc & dc \\ dc & dc & 0 & 0 \end{bmatrix},$$

$$\Lambda_{e_1}(x_2) = \begin{bmatrix} dc & 0 & 1 & 0 \end{bmatrix},$$

$$\Lambda_{e_2}(x_2) = \begin{bmatrix} 1 & 0 & dc & 1 \end{bmatrix}, \text{ and}$$

$$\Lambda_{e_3}(x_2) = \begin{bmatrix} 1 & dc & 1 & 0 \end{bmatrix}.$$

The new feedback function for states x_1 and is x_2 (when combined in u) is given by $\Phi_1(u, x_1, e_1) = 1$, $\Phi_1(u, x_1, e_2) = 0$, and $\Phi_1(u, x_1, e_3) = 1$, $\Phi_1(u, x_2, e_1) = 1$, $\Phi_1(u, x_2, e_2) = 0$, and $\Phi_1(u, x_2, e_3) = 1$. The resulting IP model is:

MIN $z_1 + z_2 + z_3 + z_4$

Subject to

$$y_{x_1,e_1,1} \geq 1 - M(1 - z_1)$$
$$y_{x_1,e_1,2} \geq 1 - M(1 - z_2) - M(1 - z_3)$$
$$y_{x_1,e_1,3} \geq 1 - M(1 - z_1) - Mz_2 - M(1 - z_3)$$
$$y_{x_1,e_2,1} \geq 1 - Mz_1$$
$$y_{x_1,e_2,2} \geq 1 - M(1 - z_1) - Mz_2 - M(1 - z_4)$$
$$y_{x_1,e_3,1} \geq 1 - Mz_2$$
$$y_{x_1,e_3,2} \geq 1 - Mz_3 - Mz_4$$
$$y_{x_2,e_1,1} \geq 1 - Mz_2 - M(1 - z_3) - Mz_4$$
$$y_{x_2,e_2,1} \geq 1 - M(1 - z_1) - Mz_2 - M(1 - z_4)$$
$$y_{x_2,e_3,1} \geq 1 - M(1 - z_1) - M(1 - z_3)$$
$$y_{x_1,e_1,1} + y_{x_1,e_1,3} + y_{x_1,e_2,2} + y_{x_2,e_2,1} + y_{x_2,e_3,1} \leq Mz_1$$
$$y_{x_1,e_2,1} \leq M(1 - z_1)$$
$$y_{x_1,e_1,2} \leq Mz_2$$
$$y_{x_1,e_1,3} + y_{x_1,e_2,2} + y_{x_1,e_3,1} + y_{x_2,e_1,1} + y_{x_2,e_2,1} \leq M(1 - z_2)$$
$$y_{x_1,e_1,2} + y_{x_1,e_1,3} \leq Mz_3$$
$$y_{x_1,e_3,2} \leq M(1 - z_3)$$
$$y_{x_1,e_2,2} + y_{x_2,e_1,1} \leq Mz_4$$
$$y_{x_1,e_3,2} + y_{x_2,e_1,1} \leq M(1 - z_4)$$
$$y_{x_1,e_1,1} + y_{x_1,e_1,2} + y_{x_1,e_1,3} \geq 1$$
$$y_{x_1,e_2,1} + y_{x_1,e_2,2} = 0$$
$$y_{x_1,e_3,1} + y_{x_1,e_3,2} \geq 1$$
$$y_{x_2,e_1,1} \geq 1$$
$$y_{x_2,e_2,1} = 0$$
$$y_{x_2,e_3,1} \geq 1$$

All variables are 0 or 1.

The solution will be

$z_1 = 1$, $z_2 = 0$, $z_3 = 1$, $z_4 = 0$,

$y_{x_1,e_1,1} = 1$, $y_{x_1,e_1,2} = 0$, $y_{x_1,e_1,3} = 1$, $y_{x_1,e_2,1} = 0$,

$y_{x_1,e_2,2} = 0$, $y_{x_1,e_3,1} = 1$, $y_{x_1,e_3,2} = 0$,

$y_{x_2,e_1,1} = 1$, $y_{x_2,e_2,1} = 0$, $y_{x_2,e_3,1} = 1$.

Therefore, we must activate outputs 1 and 2, and deactivate outputs 2 and 4.

After deriving the feedback function for every state of the mega-controller, we need to implement the mega-controller strategies on the LLD. Since the mega-controller is a finite automaton, its translation to LLD is straightforward and can be done by one of the available conversion methods in the literature [23]. However, the LLD generated from Θ is not enough to implement a correct reconfigurable control. We recall that in Section 3, during our conversion from LLD to FA, we did not consider the conversion of internal events of the PLC (such as timer events, counter events, changes in the PLC internal variables, etc.). We argued that these events cannot directly affect the observation analysis and therefore need not be included in the FA. Instead of these events, we considered their effects on the output vector. We created the virtual event ω that appeared on the FA model whenever we observed a change in the output that was not accompanied by changes in the inputs. While event ω was appropriate to perform the observation analysis, this event cannot be used in the LLD as it is not corresponding to any specific input or output element. Furthermore, the original LLD logic related to the internal components, that is not directly reflected in the FA model, must be maintained to manage the operations of internal components. In fact the extended mega-controller only manages the necessary output command modifications required to respond to the loss of information due to the unobservability of input events. However, the mega-controller cannot manage the operations of internal components.

For this reason, in the final implementation, we propose an LLD (called reconfigurable LLD) with two modules. Module 1 is the original LLD. Module 2 is the LLD generated from the extended mega-controller. Module 1 rungs are placed above the rungs of Module 2. In this case, if there is a conflict between these two modules on the activation/deactivation of outputs, the priority is given to the decisions made by Module 2. The reason is that in every PLC program scan, the rungs belong to Module 2 are executed after the execution of the Module 1's rungs. Therefore after the update of output values in the output image table by Module 1, if necessary, Module 2 overwrites some or all of these values. These values are then sent to the output channels (output hardware) for execution. Module 1 controls the original logic, including the internal variables. Module 2 tracks the extended mega-controller state and is responsible for digesting the input channel failure and repair events. As long as, the state of the mega-controller is not combined, Module 2 does not affect the output decisions made by Module 1. We recall that the IP model presented in this section is used to determine the values of the outputs for the combined states. For the individual states we do not have any modification of the output commands, that is we use the output strategy shown by Φ' . When the mega-controller is set to a combined state, the regular output decisions made by the original LLD may no longer be acceptable. In this case the output commands prescribed by the extended mega-controller (determined by the IP model) are used.

Suppose that for every $x \in X$ of the finite automaton $S = (X, \Sigma, f, x_0, X_m)$ as defined in Section 3,

$\bar{o}_x = \begin{bmatrix} \Phi'(x,o_1) \\ \Phi'(x,o_2) \\ \vdots \\ \Phi'(x,o_n) \end{bmatrix}$ determines the feedback on the outputs.

We assume that the extended mega-controller $\Theta = (U, E, g, u_0, U_m)$ is given (we use u instead y here to prevent notation confusion). Define U' as a subset of U that contains those states of U that are combined. By the construction procedure of Θ, we have $u_0 \in U - U'$. Also $U - U' \subseteq X$. Suppose that $\tilde{o}_u$ is the associated output command vector of $u \in U$. The components of $\bar{o}_u$ are determined from the solution of the IP model if $u \in U'$, and for $u \in U - U'$ we have $\tilde{o}_u = \bar{o}_u$. By this definition we have a slight violation in the definition of $\bar{o}$, as $\bar{o}$ is defined on X not $U - U'$. However we notice that every state in $U - U'$ has a member of X as its first component. This member can be used while applying $\bar{o}$ to the elements of $U - U'$. We assume that $\Delta(u)$ is the sets of incoming events to state u. We also define e_0 ($e_0 \notin E$) as the LLD initialization event. In the following, we present the generation process of Module 2 from the extended mega-controller.

Conversion from extended mega-controller to LLD (Module2):

Begin

1- Create an initialization rung. The only input contact for this rung is for event e_0, which is normally open, and the only output component is a latch coil for state u_0.

2- For every $u \in U - U'$ create one rung as follows:

2-1- For every $e \in \Delta(u)$ and $e \neq \omega$, define a branch that is the serial configuration of two normally open contacts, one for event e and the other for state v where $g(v,e)=u$.

2-2- For every $e \in \Delta(u)$ and $e = \omega$, let $g(v,e)=u$ and execute step 2-2-1.

2-2-1- For every $x \in X$ that has contributed in the construction of v (we recall that if v is a combined state then several members of X have contributed to the construction of v, and if v is not a combined state then one member of X is used in the construction of v.), and $f(x, \omega)=u$, create one branch that has a serial configuration of one normally open contact for event e, one normally open contact for every output o_i that $(\bar{o}_x)_i = 0$ and $(\bar{o}_u)_i = 1$, and one normally closed contact for every output o_i that $(\bar{o}_x)_i = 1$ and $(\bar{o}_u)_i = 0$.

2-3- Connect all the created branches in steps 2-1 and 2-2 in parallel and place the generated branch configuration as the input of the rung for state u.

2-4- Create one latch coil for state u and one unlatch coil for every $v \in U - \{u\}$. Put all these coils in a parallel structure in the output of the rung for u.

3- For every $u \in U'$ create one rung as follows:

3-1- For every $e \in \Delta(u)$ and $e \neq \omega$, define a branch that is the serial configuration of two normally open contacts, one for event e and the other for state v where $g(v,e) = u$.

3-2- For every $e \in \Delta(u)$ and $e = \omega e \in \Delta(u)$, let $g(v,e) = u$ and execute step 3-2-1.

3-2-1- For every $x \in X$ that has contributed in the construction of v and for every $x' \in X$ that has been used in the construction of u, if $f(x,w) = x'$, create one branch that has a serial configuration of one normally open contact for event e, one normally open contact for every output o_i that $(\bar{o}_x)_i = 0$ and $(\bar{o}_{x'})_i = 1$, and one normally closed contact for every output o_i that $(\bar{o}_x)_i = 1$ and $(\bar{o}_{x'})_i = 0$.

3-3- Connect all the created branches in steps 3-1 and 3-2 in parallel and place the generated branch configuration as the input of the rung for state u.

3-4- Create one latch coil for state u and one unlatch coil for every $v \in U - \{u\}$. Create one latch coil for every output o_i that $(\tilde{o}_x)_i = 1$ and one unlatch coil for every output o_i that $(\tilde{o}_x)_i = 0$.

3-5- Put all the coils generated in step 3-4 in a parallel structure in the output of the rung for state u.

End.

We have shown the LLD conversion of the mega-controller in the following example. This example includes all the required steps from the beginning to constructing the reconfigurable LLD.

7. ILLUSTRATIVE EXAMPLE

The system, shown in Fig. 4, includes a belt conveyor, a roller conveyor, a filling machine, a scale, a printer and two photoelectric sensors. The process is to fill the boxes lined up on the belt conveyor, and to weigh the boxes by a scale attached to the conveyor. The boxes are then transported onto the roller conveyor where the production related data are printed on the boxes. The boxes are fed with equal distances through a feeder in the upstream which is not considered as a plant component. Likewise, the downstream subsystems, such as packaging and storage have been excluded from the system under study. The system's operation is controlled by a PLC. The roller conveyor runs all the times. The PLC does not control the roller conveyor drive. The roller conveyor can only

accommodate one box due to its length limit. The photoelectric sensor consists of a transmitter and a receiver.

When the "Ready" signal is received, the PLC activates the belt conveyor motor to move the boxes forward. When a box breaks the photoelectric beam installed at the filling zone, the PLC stops the motor so that the box is right under the filling nozzle. After a preset of 2 second, the filling starts. When the filling is done, a "Filling_done" signal is sent to the PLC by the filling machine. Then the scale is activated to weigh the filled box. After the weight is received, the PLC starts the motor again. Because the roller conveyor runs at the same speed and direction as the belt conveyor, the box is transported smoothly onto the roller conveyor. When a box passes the photoelectric sensor located in the center of the roller conveyor, the PLC commands the printer to print on the box. This forms one cycle of the process on an individual box. When the belt conveyor transports the filled box onto the roller conveyor, the subsequent box is also moved forward to the filling zone. Simultaneously, two boxes are under process – one is being filled and another is being printed. Fig. 5 shows the original LLD. The descriptions of inputs and outputs in this LLD are given Table 3.

Figure 6 shows the finite automata model of this LLD. In converting the LLD to a finite automaton, input turn-on and turn-off actions are considered as events. The event set is, therefore, Σ ={Ready+, Ready-, ps1+, ps1-, ps2+, ps2-,

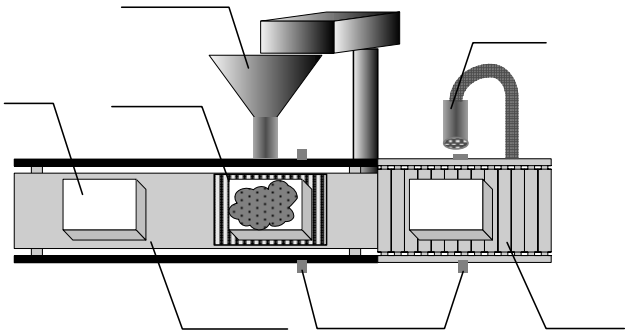


Fig. 4. Example system sketch.

Table 3. Inputs and output descriptions.

I/O	Tag Name	Physical Meaning
Input	Ready	System is ready for process.
	ps1	Photoelectric sensor 1
	ps2	Photoelectric sensor 2
	Filling_done	Filling is done.
	Weight_recd	Weight is received by main
Output	Motor	Motor drive actuator
	Filling	Filling machine actuator
	Weighing	Scale actuator
	Print	Printer actuator

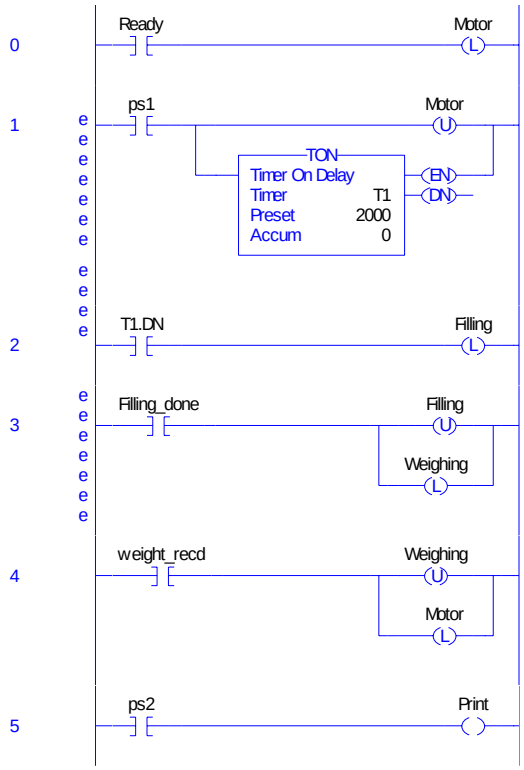


Fig. 5. Ladder logic diagram of the example system.

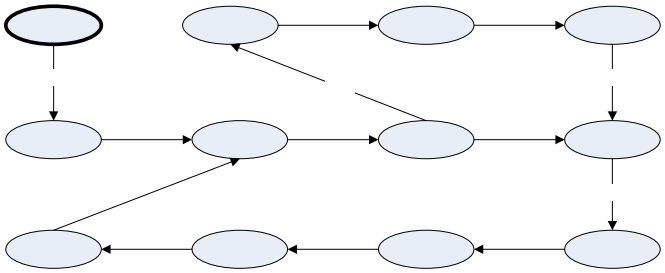


Fig. 6. Converted FA model.

Filling_done+, Filling_done-, Weight_recd+, Weight_recd-, ω }. Another product of the conversion is the feedback function Φ', given in Table 4.

To translate the feedback function Φ' to Φ, the relation between events and outputs is needed, which is listed in Table 5. Applying the algorithm discussed in Section 4, the feedback function Φ is obtained. The result is shown in Table 6.

In states 4, 10, 11 and 12, event ps2 is not controllable. The reason is that at those states, there is no combination of outputs to enable or disable this event. Therefore, from the PLC point of view, it is not controllable.

The converted FA model S and the feedback function Φ form an FA supervisory controller. We then perform the observation analysis on it. Suppose that the status of input channels (input observation means) is modeled as working, represented by letter O, and failed (including the case that the communication is significantly delayed),

Table 4. Feedback function $\Phi' : X \times O \to \{1, 0, dc\}$.

No.	States	Events			
		Motor	filling	weighing	Printing
1	00000_0000	0	0	0	0
2	10000_1000	1	0	0	0
3	00000_1000	1	0	0	0
4	01000_0000	0	0	0	0
5	01000_0100	0	1	0	0
6	01010_0010	0	0	1	0
7	01000_0010	0	0	1	0
8	01001_1000	1	0	0	0
9	01000_1000	1	0	0	0
10	01100_0001	0	0	0	1
11	01100_0000	0	0	0	0
12	01100_0100	0	1	0	0

Table 5. Event-Output relationships.

Event *e*	Λ_e
Ready+	Does not exist in any state
Ready-	Does not exist in any state
ps1+	[1 dc dc dc] at all the states
ps1-	[1 dc dc dc] at all the states
ps2+	[1 dc dc dc] at all the states expect state No. 4, 10, 11, 12 where it does not exist
ps2-	[1 dc dc dc] at all the states expect state No. 4, 10, 11, 12 where it does not exist
Filling_done+	[dc 1 dc dc] at all the states
Filling_done-	Does not exist in any state
Weight_recd+	[dc dc 1 dc] at all the states
Weight_recd-	Does not exist in any state

represented by letter X. The states in *S* are renamed by its index in Tables 4 or 6. In the initial state, state 1, we assume that all the input channels are working properly, so their status vector is "OOOOO" where each digit stands for a status of an input channel in the order listed in Table 4.

In the extended controller model Θ, all the possible statuses of each input channel are considered. Figure 7 shows a part of this model. Suppose that in state 1, the channel for the input "Ready" is failed, which makes event "Ready+" and "Ready-" unobservable. By the observation analysis, we find that there are conflicting control actions on the event "ps1+" in states 1 and 3. Therefore, the feedback adjustment strategy is applied, and the new feedback is $\Phi_1(123, 3, ps1+) = 0$. Then we apply the state aggregation strategy and accordingly states 1-3 are combined, and we call the new state "123". After solving the IP model of the previous section for states 1, 2, and 3,

Table 6. Feedback function $\Phi : X \times \Sigma \to \{1, 0, dc\}$.

No.	States	Ready+	ready-	ps1+	ps1-	ps2+
1	00000_0000	1	dc	0	0	0
2	10000_1000	dc	1	dc	dc	dc
3	00000_1000	dc	dc	1	dc	dc
4	01000_0000	dc	dc	0	0	1
5	01000_0100	dc	dc	0	0	0
6	01010_0010	dc	dc	0	0	0
7	01000_0010	dc	dc	0	0	0
8	01001_1000	dc	dc	dc	dc	dc
9	01000_1000	dc	dc	dc	1	dc
10	01100_0001	dc	dc	0	0	dc
11	01000_0000	dc	dc	0	0	dc
12	01100_0100	dc	dc	0	0	dc

No.	States	ps2-	Filling _done+	Filling _done-	Weight _recd+	Weight _recd-
1	00000_0000	0	0	dc	0	dc
2	10000_1000	dc	0	dc	0	dc
3	00000_1000	dc	0	dc	0	dc
4	01000_0000	dc	0	dc	0	dc
5	01000_0100	0	1	dc	0	dc
6	01010_0010	0	0	1	0	dc
7	01000_0010	0	0	dc	1	dc
8	01001_1000	dc	0	dc	0	1
9	01000_1000	dc	0	dc	0	dc
10	01100_0001	dc	0	dc	0	dc
11	01000_0000	dc	0	dc	0	dc
12	01100_0100	1	1	dc	0	dc

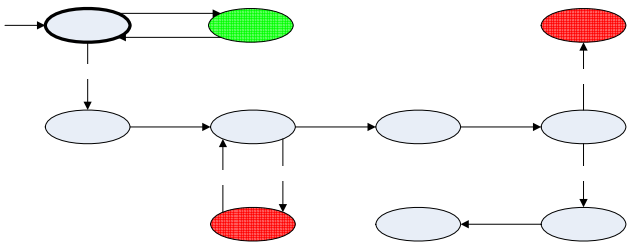
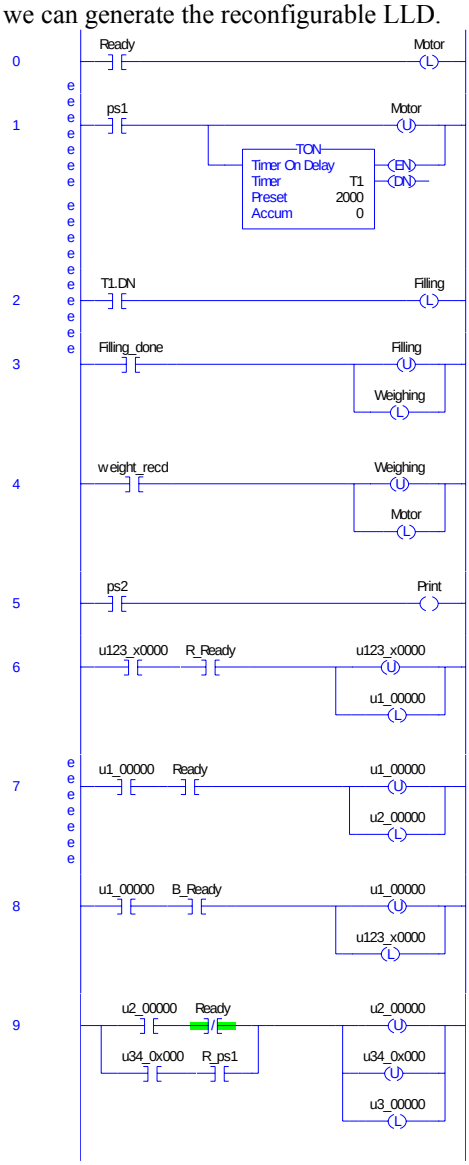


Fig. 7: Partial model of extended controller Θ.

the relationship matrices, and Φ_1, we obtain (the IP model is not shown here) the following output commands: Motor is deactivated, Filling is deactivated, Weighing is deactivated, and Printing is deactivated.

The physical explanation of this change is that in the plant events "Ready+" and "Ready-" may have or have not happened, and thus the current state could be either of states 1, 2 or 3. But the PLC is unaware of the occurrence of those events. As a result, it controls the conveyor system according to the feedback in state 1 even though the correct control instruction is supposed to be the feedback in state 2 or 3. In this case, we need to find a compromised control feedback suitable for all the three states, 1, 2 and 3. The

new control actions on outputs in states 2 and 3 are consistent with the current ones in state 1. So there is no need to change the logic of LLD to accommodate the change because the PLC program will not be set to states 2 and 3 until the "ps1" input channel is recovered. But in state 3, if the input channel for "ps1" is failed, the new feedback disables "ps1+" which is in conflict with the feedback Φ . Therefore, the logic of PLC has to be. Similarly in state 5, for the case that "Filling_done" input channel is failed, the changes are reflected in the extended mega-controller. After applying the conversion algorithm of Section 6, to the extended mega-controller, we generate the second module of the reconfigurable LLD. This module is partially shown in Fig. 8. By putting the first module (shown in Fig. 5) and the second module after that, we can generate the reconfigurable LLD.



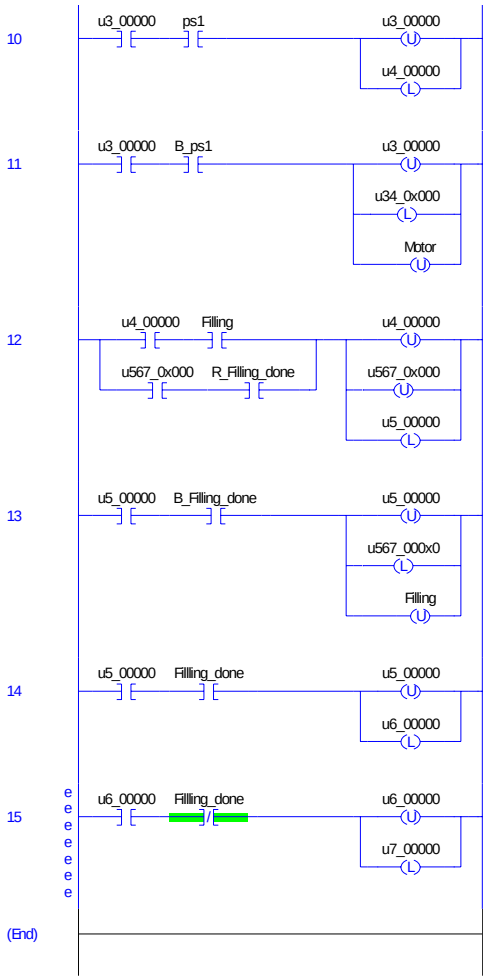


Fig. 8. Reconfigurable LLD.

8. CONCLUSION

In this paper, we introduce a framework for converting a traditional LLD to a reconfigurable LLD. The reconfigurable LLD modifies the control of its plant when the means reporting the plant events to the PLC fail. It is shown that the reconfigurable LLD can continue a proper control of the plant under conditions that the traditional LLD cannot. Partial observation theory is used in the construction of the new LLD.

One future research direction established by this work is to develop some efficient techniques to obtain the information needed by the presented framework. For example how one can efficiently derive the event-output relationships of a given controller, can be a research problem. Another research challenge is to implement the proposed method for large scale systems (LLD with thousands of rungs) and evaluate its effectiveness. Also modifying the discussed methods for the case of continuous PLC input and/or outputs is an open research problem.

REFERENCES

[1] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event Processes", SIAM Journal of Control and Optimization, vol. 25, No.1, pp. 206-230, 1987.

[2] R. Cieslak, C. Desclaux, A. S. Fawa and P. Varaiya, "Supervisory control of discrete-event processes with partial observations", IEEE Transactions on Automatic Control, vol. 33, no. 3, pp. 249-260, 1988.

[3] F. Lin and W. M. Wonham, "On observability of discrete event systems", Information Sciences, vol. 44, no. 3, pp. 173-198, 1988.

[4] H. Darabi, M. A. Jafari and A. L. Buczak, "A control switching theory for supervisory control of discrete event systems", IEEE Transactions on Robotics and Automation, vol. 19, no.1, pp. 131-137, 2003.

[5] J. Liu and H. Darabi, "Control Reconfiguration of Discrete Event Systems Controllers with Partial Observation", IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 34, No. 6, pp. 2262-2272, 2004.

[6] M. Fabian, A. Hellgren, "PLC-based Implementation of Supervisory Control for Discrete Event Systems", Proceedings of the 37the IEEE conference on Decision & Control, Tampa, Florida, USA, Dec. 1998.

[7] J. Zaytoon, "Specification and Desgin of Logic Controllers for Automated Manufacturing Systems", Robotics & Computer-Integrated Manufacturing, Vol. 12, No. 4, pp. 353-366, 1996.

[8] L. Lenart, "Formal Analysis of Existing Control Software in Cyclic Closed Production Line", 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics Proceedings 8-12, Jul. 2001.

[9] K. Venkatesh, and M.C. Zhou and R. J. Caudill, "Comparing ladder logic diagrams and Petri nets for sequence controller design through a discrete manufacturing system", IEEE Transactions on Industrial Electronics, Vol. 41 Issue 6, pp. 611-619, Dec. 1994.

[10] M.C. Zhou, and E. Twiss, "Design of industrial automated systems via relay ladder logic programming and Petri nets", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. 28 Issue 1, Feb. pp. 137-150, 1998.

[11] S.C. Lauzon, J.K. Mills and B. Benhabib, "An Implementation Methodology for the Supervisory Controller for Flexible Manufacturing Workcells", SME Journal of Manufacturing Systems, Vol. 16, No.2, pp. 91-101, 1997.

[12] G. B. Lee, J. S. Lee, "The State Equation of Petri Net for the LD Program", 2000 IEEE International Conference on Systems, Man, and Cybernetics, Vol. 4, 2000.

[13] P.J.G. Ramadge, and W.M. Wonham, "The control of discrete event systems", Proc. IEEE. 77: 81 – 98, 1989.

[14] U.A. Buy and H. Darabi, "Deadline-enforcing supervisory control for time Petri nets", IEEE Multiconference on Computational Engineering in Systems Applications, Lille, France, 2003.

[15] Ugo Buy and Houshang Darabi. Sidestepping verification complexity with supervisory control. The Monterey Workshop: Software Engineering of Embedded Systems, September 2003.

[16] J. Desel and J. Esparza, "Free Choice Petri Nets", Cambridge, Cambridge Univ. Press, 1995.

[17] T. Murata, "Petri nets: Properties, analysis, and applications", Proc. IEEE. 77, pp. 541–580, 1989.

[18] J.L. Peterson, "Petri Net Theory and the Modeling of Systems", Engelwood Cliffs, NJ, Prentice-Hall, 1981.

[19] W. Reisig, "Petri Nets", Berlin, Springer-Verlag, 1985.

[20] M.C. Zhou and F. DiCesare, "Petri Net Synthesis for Discrete Event Control of Manufacturing Systems", Amsterdam, Kluwer, 1993.

[21] L. E. Holloway and B. H. Krogh, "Synthesis of feedback control logic for a class of controlled Petri nets", IEEE Trans. Automat. Contr. 35: 514–523, 1990.

[22] B. A. Brandin, "The Real-Time Supervisory Control of an Experimental Manufacturing Cell", IEEE Transactions on Robotics and Automation, Vol. 12, No.1, Feb. 1996.

[23] T. O. Boucher, Computer Automation in Manufacturing Systems, Chapman & Hall, London, 1996.

[24] A. Hellgren, M. Fabian and B. Lennartson, "Synchronised Execution of Discrete Event Models Using Sequential Function Charts", Proceedings of the 38th Conference on Decision & Control, Phoenix, Arizona, Dec. 1999.

[25] J. Liu and H. Darabi, "Ladder logic implementation of Ramadge-Wonham supervisory controller", Proceedings of Sixth International Workshop on Discrete Event Systems (WODES), pp. 383 -389, 2002.

[26] Darabi, H., Sampath, R., and Naylor, D. "PLC formal control methodologies: Does academia supply what industry demands?" ISA TECH/EXPO Technology Update Conference Proceedings, v 422, pp. 131-143, 2002.

Jing Liu (S'04) received the B.S. degree from Hebei University of Technology, Tianjin, China, the M.S. degree from Beijing University of Posts and Telecommunications, Beijing, China. She is currently working towards the Ph.D. degree in industrial engineering at University of Illinois at Chicago.

Her research interests include formal methods, supervisory control, reconfiguration of discrete-event systems, and their applications in manufacturing.

She is a student member of IEEE, Honor Society of Phi Kappa Phi, and Institute of Industrial Engineers.

Houshang Darabi is currently an Assistant Professor with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago. He received the Ph.D. degree in industrial and systems engineering from Rutgers University, New Brunswick, NJ, in 2000.

His research interests include the application of discrete-event systems control theory in modeling and analysis of service and manufacturing systems, computer-integrated manufacturing, supply chain networks, and manufacturing information systems. His research has been supported by the National Science Foundation, the Department of Commerce, Motorola Inc., and several other agencies. He has published in different prestigious journals and has presented his research in national and international conferences. Dr. Darabi is a senior member of the Institute of Industrial Engineers (IIE), a member of the Instrumentation, Systems and Automation Society (ISA), Institute of Electrical and Electronics Engineers (IEEE), and the Institute for Operations Research and the Management Sciences (INFORMS).