International Journal of Innovative
Computing, Information and Control
Volume **9**, Number **11**, November **2013**

 ISSN 1349-4198
pp. 4519–4530

VALIDATING PROGRAMMABLE LOGIC CONTROLLER SYSTEMS WITH DURATION CALCULUS

Anping He[1,2], Jinzhao Wu[1], Shihan Yang[1] and Yongquan Zhou[1]

[1]Guangxi Key Lab of Hybrid Computation and IC Design Analysis
Guangxi University for Nationalities
No. 188, East Daxue Road, Nanning 530006, P. R. China
hapetis@gmail.com; himrwujz@yahoo.com.cn

[2]School of Information Science and Engineering
Lanzhou University
No. 222, South Tianshui Road, Lanzhou 730000, P. R. China

Received November 2012; revised March 2013

Abstract. *The programmable logic controller (PLC) system is a type of the hybrid systems, which is a widely used safety-critical system in industry. It is regarded that the formal method is a valuable and indispensable way of analyzing and validating the PLC systems. However, which formal methods and how to use the methods are challenges. In this article, we propose a specific formal method of the PLC systems, e.g., we formalize the PLC systems by the EDC formulae hierarchically, specify the property as the formulae, and then analyze and verify the system in the framework of the EDC calculus. So in fact we propose an EDC based hierarchical formal method of this type of system, we trust this method is applicable. We also harness two examples to demonstrate the effectiveness and feasibility of the method.*
Keywords: PLC, EDC, Hierarchical analysis

1. **Introduction.** The *Programmable logic controllers* (PLC) are widely used in the safety-critical industrial applications. A PLC system is a hybrid system where its outputs are produced in response to the input conditions with the time constraints. There is a growing demand for the validation of PLC systems.

The PLC system is one of the safety critical applications, requiring guarantees of safe operation, e.g., the unwarrantable configurations are not allowed. In order to design and produce a high credible PLC system, the (formal) verification is essential and effectual. Formally, it is considered that the PLC system is one of the hybrid systems in which computational processes interact with physical processes, then this specific system is always analyzed by the formal hybrid system theory to eliminate bugs as well as raise designers' confidence.

There are three main types of model based formal methods for PLC systems: Petri-net, timed automata and hybrid automata. In [1, 2], the authors analyze PLC system in terms of Petri-net, however, the Petri-net cannot directly express continuous-time in PLC systems. [3, 4, 5, 6, 7] adopt the timed automata for the analysis and validation of PLC systems, but it is impossible to bind the equations and the automata transitions together to make analysis simple and direct. The hybrid automata [8, 9, 10] combines automata with dynamic equations, but it still needs more manual work to solve these equations. On the other hand, there are few calculus based methods. [11] shows a way of translating duration calculus to automata to enable automation of verification procedure. [12] adopts duration calculus to analyze an aerospace application hierarchically, but it does not study

the relation between the hierarchical design and the formal calculus, e.g., the semantics. Andre Platze proposed a hybrid dynamic logic [13] , which is a very interesting logic system of the hybrid system, but it is not suitable to formalize the PLC system hierarchically. To summarize, the model based formal analysis of the PLC system can not reject the human intellection, e.g., the calculation of equations takes more time, while the whole procedure is lack of automation. On the other hand, the calculus combines the equation processing and the logic reasoning together, this type of combination makes the analysis consistent.

As we claimed previously, the PLC system is one of the high complex system in which computational processes interact with physical processes, so it is a big challenge to get the formal expression of the PLC system effectively and accurately. In our study, we focus on a widely-used hierarchical design methodology. With our study and observation of this type of hierarchical model, we find it is easy and direct to construct the *extended duration calculus* [14] (EDC for short) formulae to express time variance of the hierarchical model, so, with the EDC calculus, it is natural to specify, analyze and verify the PLC system hierarchically. To the best of our knowledge, no article studies the PLC system by the hierarchical way of analysis and verification.

In this article, we use $\mathbb{R}$ to denote the real numbers, $\mathbb{B}$ to the Booleans. We also note $=$ for equality, $\triangleq$ for a notation of definition.

2. Analysis of the PLC Systems with EDC.

In this section we want to map the EDC formulae to the PLC system, e.g., show the hierarchical model based semantics of the PLC system. Let us introduce a formal hierarchical conceptual model.

2.1. Hierarchical model of the PLC systems.

A hierarchical model is an effective way of analyzing complex systems. This type of model explores the system by abstraction and makes each hierarchy simple and isolated. A famous real-time hierarchical control conceptual model was proposed by Morin and Nadjm-Tehrani in [15]. We use a reduced hierarchical model in Figure 1.

In Figure 1, the continuous component, input/output devices, sensors, and actuators are seen as the environment, the above discrete part monitors the continuous variants

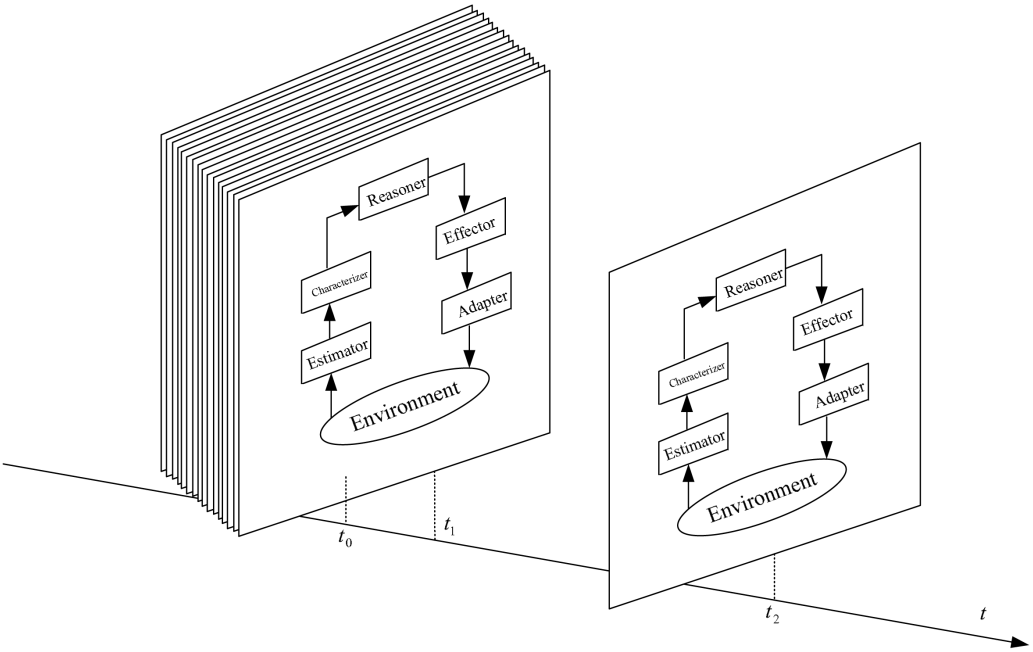

FIGURE 1. The hierarchical model

of the environment, meanwhile, the behavior of environment is adjusted instantaneously. This hierarchical model describes the principles of system design, then it is functionally equivalent to the practical PLC system.

Each hierarchy corresponds to the following functions: *Estimator*, *Adapter*, *Characterizer*, *Effector* and *Reasoner*. The model responds to variants of the environment in a very short time interval repeatedly. The response interval is so short that decision-making part is regarded responding continuously. Nevertheless, because the hierarchy model presents the design of the PLC system, the model keeps true during the system run (see Figure 1).

Let us show this hierarchy formally. The continuous component, e.g., the 'environment' in Figure 1, is expressed by the state equations derived from the physical laws and output functions, these functions are variant over time. Then we seen the environment is the function as the below:

$$environment : \mathbb{R}^{+,0} \to \mathbb{R}.$$

For the discrete part, e.g., decision-maker, it is necessary and convenient to use typed data while analysis, however, the data for the sensors or actuators are always non-typed analog value. So the estimator and adapter are necessary for data conversions:

$$estimator : \mathbb{R} \to \mathbb{R}$$

and

$$adapter : \mathbb{R} \to \mathbb{R}$$

Due to the size of PLC program compatible data, mixing of discrete and analog, as well as many events (such as button pushing), it is difficult to verify the PLC system, then the abstraction and isolation are utilities, we firstly classify/symbolize the data and then determine significant events by the characterizer:

$$characterizer : \mathbb{R} \to \mathbb{B}$$

Then, the reasoner makes decisions by the symbols from characterizer:

$$reasoner : \mathbb{B} \to \mathbb{B}$$

As soon as the decision is made, the effector implements those decisions as the typed written values with the intelligent algorithms:

$$effector : \mathbb{B} \to \mathbb{R}$$

Then, the hierarchical model is a conjunction of these above six formal concepts, it captures all characters of PLC system in a time point, and then describes the sequence of snapshots of the system behavior along time, see Figure 1.

2.2. **Semantics.** The same as the most formal methods, the semantic model of the EDC bases on the state. The difference is that EDC describes the state implicitly. A state is a time related measurement of system, which is composed of the current values of all variables of the system. In the EDC terminology, these variables are called *state names* (SN) and *state variables* (SV). In this section, we interpret the EDC formulae by the hierarchical model of the PLC system, e.g., showing a hierarchy based semantics of the EDC.

The variable set of a PLC system is composed of all variables describing the hierarchical model, which is noted by $V = SN \cup SV$. A state variable $\nu \in V$ is one over time, its forms are either $\nu(t) \in \mathbb{B}$ or $\nu(t) \in \mathbb{R}$ ($t \in \mathbb{R}^{+,0}$ is time). Then a state of PLC system will be an assignment of all state variable at a time point. So the variables expressing *environment*, *estimator* and *adapter* are seen as the state names, but the ones describing *characterizer*, *reasoner* and $effector$ are state variables.

The elementary or boolean functions over V are composed of the state variables, real constants and common operators including $+$, $-$, $\times$, $/$, $\int$, sin, cos, etc. This kind of functions are corresponding to the *state terms* in EDC, and describing the behaviors over time, e.g., the elementary functions show the continuous timed variant or transformation, but the boolean expressions describe the timed predicate, all of which become the *state expression*, *se*, and *state assertion*, P, separately. The state expression describes the continuous behavior, but the state assertion for satisfactory of the discrete predicate.

Let *interpretation*, $\mathcal{I}[\![\ ]\!]$, be a function which associates each state variable, type, and operator symbols with a fixed meaning of the PLC system. Then we interpret the expressions with their common meanings:

$$\begin{aligned}\mathcal{I}[\![x]\!] &\triangleq \int f \text{ with } x' = f\\ \mathcal{I}[\![x(t)]\!] &\triangleq f(t) \text{ with } x = f\\ \mathcal{I}[\![P]\!] &\triangleq P_1 \clubsuit P_2 \text{ with } P = P_1 \clubsuit P_2\\ \mathcal{I}[\![P]\!] &\triangleq \neg P_1 \text{ with } P = \neg P_1\end{aligned}$$

in which $x \in SN$, P, P_1, P_2 be assertions, f a function over $\mathbb{R}^{+,0}$, and $\clubsuit \in \{\wedge \Rightarrow, \vee, \Leftrightarrow \cdots\}$.

Now let us study the expression over time. For a given interval $[b, e]$, let $\mathcal{I}[\![\!]\!][b, e]$ be interpretation of a given time interval $[b, e]$. The interpretation of *durations*, d, could be:

- duration of state expression, e.g., $d = \int se$, specifies the continuous behavior by the state equation of the environment of the hierarchical model:

$$\mathcal{I}\left[\!\!\left[\int se\right]\!\!\right][b, e] \triangleq \int_b^e se(t)dt = \mathbf{e}.se - \mathbf{b}.se$$

  with $\mathbf{b}.se$ and $\mathbf{e}.se$ be initial and final values of the integration.
- duration of state assertion, e.g., $d = \int P$, showing how long the satisfaction of a property over a time duration,

$$\mathcal{I}\left[\!\!\left[\int P\right]\!\!\right][b, e] \triangleq \int_b^e Pdt = (e_0 - b_0) + (e_1 - b_1) + \cdots$$

  $e_i > b_i \wedge b_{i+1} > e_i \wedge b_i$, $e_i \in [b, e]$ with $i = 1, 2, \cdots$, and P keeps true in interval $[b_i, e_i]$.

The symbol ℓ is an abbreviation for the duration of state assertion keeping true on a whole interval. We denote the time interval as ℓ, e.g.,

$$\mathcal{I}[\![\ell]\!][b, e] \triangleq \int true$$

The duration is not strong enough to express some properties of PLC. For example, let P_1 show a light keeping on and P_2 a buzzer on, light and buzzer work exclusively. Supposing in a time internal, we only know duration of P_1, e.g., $\int P_1$, it is easy to know that duration of P_2 is equal to a *duration term*, $\ell - \int P_1$. We can build duration terms with durations, a time variable t, initial value ($\mathbf{b}.se$) and final value ($\mathbf{e}.se$), and a function F over $\mathbb{R}$ with $sig(F) = (\mathbb{R}, \mathbb{R}, \cdots, \mathbb{R}, \mathbb{R})$, then the interpretation of a duration term *dl*

is:

$$\mathcal{I}[\![dl]\!][b,e] \triangleq \begin{array}{cc} \mathcal{I}[\![d]\!][b,e] & \text{for } d \\ \mathcal{I}[\![t]\!] & \text{for } t \\ \mathcal{I}[\![se(b)]\!] & \text{for } \mathbf{b}.se \\ \mathcal{I}[\![se(e)]\!] & \text{for } \mathbf{e}.se \\ F(\mathcal{I}[\![dl_1]\!][b,e], \mathcal{I}[\![dl_2]\!][b,e], \cdots) & \text{for } F(dl_1, dl_2, \cdots) \end{array}$$

Moreover, the predicates over duration terms always become the top level expression of PLC system, e.g., the hierarchical model of the PLC system, so does its complex properties. We call the *duration formula* be the composition of predicates of duration terms with boolean operators, quantifiers and an extended 'chop' operator. Let R be a predicate symbol over $\mathbb{R}$ ($sig(R) = (\mathbb{R}, \mathbb{R}, \cdots, \mathbb{R}, \mathbb{B})$), its interpretation is its common meaning. Then the interpretation of the duration formula D is:

$$\mathcal{I}[\![D]\!] \triangleq \begin{array}{cc} \mathcal{I}[\![G(dt_1, dt_2, \cdots)]\!] & \text{for } R(dl_1, dl_2, \cdots) \\ \neg\mathcal{I}[\![D]\!] & \text{for } \neg D \\ \mathcal{I}[\![D_1]\!] \wedge \mathcal{I}[\![D_2]\!] & \text{for } D_1 \wedge D_2 \\ \mathcal{I}[\![D_1; D_2]\!] & \text{for } D_1; D_2 \\ \mathcal{I}[\![\forall v : \mathbb{R} \cup \mathbb{B} \bullet D]\!] & \text{for } \forall v : \mathbb{R} \bullet D \end{array}$$

The *chop* operator of $D_1; D_2$ specifies that PLC system holds D_1, then holds D_2. E.g., these two properties are satisfied sequently in a time interval $[b, e]$, we have following semantics:

$$\mathcal{I}[\![D_1; D_2]\!] \triangleq \exists m \in [b, e] : \mathcal{I}[\![D_1]\!][b, m) \wedge \mathcal{I}[\![D_2]\!][m, e]$$

We also specify the properties of the PLC system quantitatively:

$$\mathcal{I}[\![\forall v : \mathbb{R} \cup \mathbb{B} \bullet D]\!] = \quad \forall v \in \mathbb{R} \cup \mathbb{B} : \mathcal{I}[\![D]\!]$$

Moreover, the properties of the PLC system may hold a formula in a time interval, e.g., except numerable points, the formula is always satisfied. We use the *interval assertion*, $\lceil D \rceil$, to describe this predicate (D is a duration formula).

$$\mathcal{I}[\![\lceil D \rceil]\!][b, e] \triangleq \forall t \in [b, e] : D(t) = true$$

Finally, we use $\Box D$ and $\Diamond D$ to express some properties of PLC system are "eventually" or "always" true (similar to the temporal logic), their semantics are defined recursively:

$$\mathcal{I}[\![\Box D]\!][b, e] \triangleq \forall b, e \in [0, \infty] : b \leq e \bullet \mathcal{I}[\![D]\!][b, e]$$
$$\mathcal{I}[\![\Diamond D]\!][b, e] \triangleq \exists b, e \in [0, \infty] : b \leq e \bullet \mathcal{I}[\![D]\!][b, e]$$

Now we can reason the PLC system according to the laws of EDC calculus listed in [14].

2.3. **Verification of the holistic PLC systems.** Let us analyze the state equations of a PLC system. We can visualize the hierarchical conceptual model as a circle, which begins from and ends in the environment. The most significant component of environment is state equations of the physical part. The state equations are in differentiation forms [16].

The formal expression of the PLC system is the conjunction of *environment*, *estimator*, *adapter*, *characterizer*, *reasoner* and $effector$. We denote the design of the PLC system as S_{PLC} , which is defined as follows:

$$S_{PLC} = environment \wedge estimator \wedge adapter \wedge \\ characterizer \wedge reasoner \wedge effector$$

The engineers have their confidence of their design capturing all characters of the PLC system in any case, e.g., S_{PLC} would hold in each time point and interval. Then the system run of the PLC system is composed of the sequence of snapshots of S_{PLC} (see Figure 1), it is obviously that S_{PLC} is invariant over time, e.g., if we substitute each state variable of S_{PLC} by its current value, the S_{PLC} is still satisfied, so we have the following expression of the PLC system:

$$\lceil S_{PLC} \rceil$$

The requirements of the PLC system are changing by the concrete application, we just let R be a EDC formula of a system property. Then the basic question is whether S_{PLC} holds R in any case, which can be determined by S_{PLC} guarantee R in each time interval:

$$\lceil S_{PLC} \rceil \Rightarrow \Box R$$

The above equations are solved in the framework of the EDC calculus. We will show the feasibility of our methods in the next sections.

3. A PLC Controlled Tank System.

In this section, we introduce a PLC controlled tank system to demonstrate how the hierarchical model be built, as well as the EDC based validation procedure.

3.1. Specification of a PLC controlled tank.

The system consists of a water tank with the input and output channels and a PLC controller. There are two electromagnetic valves A and B, three buttons b_1, b_2 and b_3 for switching to automatic control, halt and manual control, a water level sensor h, and five lights l_1, l_2, l_3, l_4 and l_5. The lights represent system status of manual, automatical control, safe, low and high level; and a buzzer for alarm respectively.

The input water rate (v_{in}) is constant and larger than the output (v_{out}), it is determined by a controllable valve A with an analog value. Valve B is just a valve for enabling and disabling the output.

Initially, the tank is empty, the valves are closed, the buzzer and all indicators keep off. Water will be poured in the tank manually by pushing b_3 until the water level arrives at a standard value (75%H, H the height of the tank) with input rate v_{in}, e.g., A opened entirely. Meanwhile, l_1 keeps on during this procedure.

After the water level arrives 75%H, user pushes b_1, making A closed, l_1 turned off. The automatic control is activated, e.g., the PLC adjusts the valves, indicators and buzzer

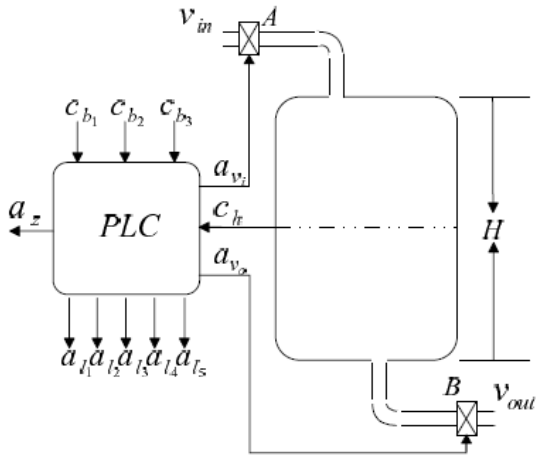

FIGURE 2. The tank

to keep water level automatically, as well as, shows system status and alarms workers if emergency appears.

During the automatically control procedure, some switching policies and the PI (Proportional and Integral) [17, 18, 19] algorithm are adopted to maintain water level and avoid overflow or dry of tank. PI is one of the most widely used industrial applications of intelligent algorithm. It is a part of PID (abbreviation of Proportional, Integral and Derivative intelligent controlling algorithm). The PID (PI in fact) algorithm has been implemented by the PLC system with a PID loop instruction, which involves the parameters of loop gain, loop sample time and integration period. More information about PID is in Section 3.2. In this design, the ideal value of the water level is assumed as $75\%H$, loop gain $K_C = 0.25$, loop sample time $T_S = 0.1s$ and the integration period of the loop $T_I = 30m$.

To avoid large deviation of PI algorithm (especially in previous phase), we need some simple but useful understandings for switching controllable A and B, i.e., if the tank approaches overflow (or dry), A (B) is closed but B (A) opened entirely until the water level reaches certain range, then PI is automatically used again. Let us show our understandings: if the water is in the safe interval ($70\%H \sim 80\%H$), PI algorithm calculates the analog value sent to controllable A and keeps the output B opened. However, if the water level is too low ($< 5\%H$), the buzzer is tuned on, B closed and A opened entirely, which makes the water pour in until the level arrives at the safe level. The case of too high ($> 95\%H$) is similar, B is opened and A closed to avoid overflow until safe level.

In contrast to concrete control understandings above, the understandings for both high ($80\%H \sim 95\%H$) or low ($5\%H \sim 70\%H$) water levels is specified indirectly: keep the previous understanding. In other words, if the previous occasion is the safe one, the algorithm will be the PI. Otherwise, it follows the way of too-high-water-level-control or too-low-water-level-control with buzzer off.

Whenever b_2 is pushed, the system halts: all valves are closed, buzzer and lights are all off.

3.2. **PID controller.** A proportional/integral/derivative controller (PID controller) [17, 18, 19] is a generic control loop feedback mechanism widely used in industry. A PID controller attempts to eliminate the deviation between the reading of measured sensor and a desired value by calculating and then emitting a corrective action to make the system adjusted. An industrial example of the PID algorithm can be found in the following equation [20]:

$$M(t) = K_C \times e(t) + K_C \times \int_0^t e(\tau)d\tau + M_{initial} + K_C \times \frac{de(t)}{dt} \tag{1}$$

where $M_{initial}$ is the constant initial value of the loop output, and all three loop gains use the same constant K_c. We adopt this format for the rest of this paper, with $M_{initial} = 0$.

3.3. **Formal modeling and specification.** Let us perform hierarchical analysis on this realistic case. The hierarchical model in Figure 1 is just a rough model helping designers map the specification to formulae of duration calculus. The environment will be expressed by the state equations of the controlled physical equipment. However, how about the PLC that may mix numeric computation (for intelligent-controlling algorithm) and symbolic reasoning (for decision-making)? We perform the following six steps for hierarchical analysis:

- *Estimator*:

  Estimator maps all input signal values, including the analog of water level sensor and the discrete of three buttons, to reals. Let v_h be the current signal value of water

level sensor, v_{b_1}, v_{b_2} and v_{b_3} for buttons, let c_h a function converting the integer analog value into real and then normalizing the resulting real number (taking Siemens S7-200 PLC for example, the standard is 32000.0), c_{b_1}, c_{b_2} and c_{b_3} simply map the integer signal value to 0 or 1:

$$c_h(v_h) = v_h/32000.0 \quad c_{b_i} = \begin{cases} 1, & \text{if } v_{b_i} > 0; \\ 0, & \text{others} \end{cases} \quad \text{let } i = 1, 2, 3.$$

- *Characterizer*:

  Characterizer is used to classify the values, mapping the real to Boolean, i.e., a predicate for these real numbers. According to Section 3.1, the condition of ideal water level, $70\%H \sim 80\%H$, is denoted by a Boolean symbol OK, as well as low, $high$, dry and $full$ for low, high, too low and too high level.

$$low \Leftrightarrow 5\% \leq c_h < 70\%H \quad high \Leftrightarrow 80\%H < c_h \leq 95\%H$$
$$dry \Leftrightarrow c_h < 5\%H \quad full \Leftrightarrow c_h > 95\%H \; OK \Leftrightarrow 70\%H \leq c_h \leq 80\%H$$

  unlike the sensors, buttons only have two discrete states, e.g., pushed or not. In our way of formalization, we consider if a button is pushed, then it will keep down until other buttons are pushed or it is pushed again, e.g., we regard the button will stay in the pushed position logically to avoid to record the previous button state. Three buttons event can be symbolized as

$$start \Leftrightarrow c_{b_1} = 1 \quad stop \Leftrightarrow c_{b_2} = 1 \quad cmd \Leftrightarrow c_{b_3} = 1$$

- *Reasoner*:

  Reasoner is a decision-maker based on the symbols (predicates) of characterizer, mapping the Boolean connection of symbols to the decisions in terms of the specification. Under the tank specification in Section 3.1, we have three main decision: manual control, automatic control and halt the system. However, in automatic control the phenomena occasionally varies from different water levels. So it is necessary to refine the automatic control decision. We use the following:

$$manual \Leftrightarrow cmd \wedge \neg auto \wedge \neg halt \quad auto \Leftrightarrow start \wedge \neg manual \wedge \neg halt$$
$$halt \Leftrightarrow stop \wedge \neg auto \wedge \neg manual$$
$$keep \Leftrightarrow OK \wedge \neg low \wedge \neg high \wedge \neg dry \wedge \neg full \wedge auto$$
$$lower \Leftrightarrow \neg OK \wedge low \wedge \neg high \wedge \neg dry \wedge \neg full \wedge auto$$
$$rise \Leftrightarrow \neg OK \wedge \neg low \wedge high \wedge \neg dry \wedge \neg full \wedge auto$$
$$handle\_dry \Leftrightarrow \neg OK \wedge \neg low \wedge \neg high \wedge dry \wedge \neg full \wedge auto$$
$$handle\_overflow \Leftrightarrow negOK \wedge \neg low \wedge \neg high \wedge \neg dry \wedge full \wedge auto$$

- *Effector*:

  Effector maps the decisions to the algorithms, which generate the value written in the actuators and output devices. Let us consider the tank system again. From the specification, we can find several algorithms, including: turn-on or turn-off of lights, open or close of output valves, open, close or intelligent control of input valves and open or close of buzzer. Let $f(t)$ be a PID function, a_{v_i} and a_{v_o} be the values needed by input and output valves, a_{l_i} for light i with $i = 1, 2, 3, 4, 5$, and a_z for buzzer. Using $\Rightarrow$ to denote the imply operator, we have:

$halt \Rightarrow a_{v_i} = a_{v_o} = a_{l_1} = a_{l_2} = a_{l_3} = a_{l_4} = a_{l_5} = a_z = 0$
$manual \Rightarrow a_{v_i} = a_{l_1} = 1 \wedge a_{v_o} = a_{l_1} = a_{l_2} = a_{l_3} = a_{l_4} = a_{l_5} = a_z = 0$
$keep \Rightarrow a_{l_3} = 1 \wedge a_{l_4} = a_{l_5} = a_z = 0$
$lower \Rightarrow a_{v_i} = f(t) \wedge a_{v_o} = a_{l_4} = 1 \wedge a_{l_3} = a_{l_5} = a_z = 0$
$rise \Rightarrow a_{l_5} = 1 \wedge a_{l_3} = a_{l_4} = a_z = 0$
$handle\_dry \Rightarrow a_{v_o} = a_{l_1} = a_{l_2} = a_{l_3} = a_{l_5} = 0 \wedge a_{v_i} = a_{l_4} = a_z = 1$
$handle\_overflow \Rightarrow a_{v_i} = a_{l_1} = a_{l_2} = a_{l_3} = a_{l_4} = 0 \wedge a_{v_o} = a_{l_5} = a_z = 1$

$f(t)$ is essentially a principle of PI control to determine the signal sent to input valve A. Let the ideal water level be $75\%H$, the deviation becomes $e(t) = 75\%H - c_h(t)$, then according to Equation (1):

$$f(t) = 0.25 \times (75\%H - c_h(t))c + 0.25 \times \int_0^t (75\%H - c_h(t))dt$$

with 0.25 for loop gain.

- *Adapter*: Adapter realizes the value of the algorithm. Since only the analog value needs to be converted with some standard, others are the same as the the ones in effector, so we only list r_{v_i} here for the analog-data controlled input valve:

$$r_{v_i} = 32000 \times a_{v_i}.$$

- *Environment*:

  Environment is expressed by the state equations and output equations, all of which are derived from physical laws. In this tank system example, we can get the state equation of the tank by analyzing the rate of water level. They are the input and output water rate impact on the water level. However, all the input/output rates are controlled by the electromagnetic valves A and B: A is limited by PLC with an analog value r_{v_i}, while B is a digital value r_{v_o}. So state equation of the environment will be

$$v_h(t)' = r_{v_i} \cdot v_{in} - r_{v_o} \cdot v_{out}. \tag{2}$$

  Let l_i denote the output function of the light i and z one of the buzzer, the environment could be described by those functions:

$v_h' = r_{v_i} \cdot v_{in} - r_{v_o} \cdot v_{out} \quad l_1 = r_{l_1}\ l_2 = r_{l_2}\ l_3 = r_{l_3} \quad l_4 = r_{l_4}$
$l_5 = r_{l_5} \quad z = a_z$

Then the set of state variables is the set of all variables involved in our modeling procedure above. We now present properties of PI control.

According to the definition of the 'Estimator' and the 'Adapter':

$$c_h = v_h/32000.0 \qquad r_{v_i} = 32000 \times a_{v_i}$$

When the input valve is controlled by PID, the input water rate is $a_{v_i} = f(t)$. So Equation (2) becomes:

$$32000 \times c_h(t)' = 32000 \times f(t) \cdot v_{in} - 32000 \times r_{v_o} \cdot v_{out}$$

Then we can get the following theorem:

Theorem 3.1. *During PI control, i.e., while the following function*

$$c_h(t)' = \left[0.25 \times (75\%H - c_h(t)) + 0.25 \times \int_0^t (75\%H - c_h(t))dt\right] \cdot v_{in} - v_{out}$$

is effected, the water level will eventually be stable at $75\%H$, with $75\%H$ for standard water level, 0.25 for loop gain and 0 for the initial output value of loop.

3.4. **Property deduction with EDC.** Now the formalization of the PLC system is

$$S_{PLC} = environment \wedge estimator \wedge characterizer \wedge reasoner \wedge effector \wedge adaptor$$

S_{PLC} keeps true during each time interval, e.g., $\lceil S_{PLC} \rceil$.

Let us validate this PLC controlled tank system by EDC. We will prove this system with following three safety properties.

Property 1.

$$\lceil S_{PLC} \rceil \Rightarrow \Box \lceil 0 \leq h_c \leq H \wedge auto \rceil$$

The property states that the water level of the tank is always kept in the safe range during the automatic control mode.

Property 2.

$$\lceil S_{PLC} \rceil \Rightarrow \Box \lceil (h \geq 95\%H) \wedge auto; (80\%H < h < 95\%H) \wedge auto; (75\%H \leq h \leq 80\%H) \wedge auto \rceil$$

The property states if the tank nearly overflows, then it can be self-controlled and reduced to the safe water level.

Property 3.

$$\lceil S_{PLC} \rceil \Rightarrow \Box \lceil (c_h \leq 5\%H) \wedge auto; (5\%H < c_h < 70\%H) \wedge auto; (70\%H \leq c_h \leq 75\%H) \wedge auto \rceil$$

If the tank nearly dries, then it can be self-controlled and poured in until arriving at the safe water level.

4. **A Thermostat.** We consider a PLC controlled thermostat [10].

Thermostats are widely used to control room temperatures. If the thermostat is set in heating mode, it will automatically turn a heater on or off to warm up the room. Similarly, if the thermostat is set in cooling mode, it will automatically turn an air conditioner on or off to cool down the room. The automatic control is performed using a PLC. The temperature can be expressed mathematically, e.g., when the heater turns off, the temperature, denoted by x, decreases exponentially $x(t) = \theta \times e^{-K \times t}$, where t is the time, θ is the initial temperature, and K is a constant determined by the room; when turns on, the temperature follows the function $x(t) = \theta \times e^{-K \times t} + h \times (1 - e^{-K \times t})$, where h is a constat depending on the power of the heater. Designers wish the temperature could be kept between m and M degrees and the heater be turned on and off accordingly to make users comfortable.

According to the above specification, we can derive the dynamic equations of the temperature. Let $a \in \{0, 1\}$ be a switch variable for heater control, we can derive the dynamics of the PLC controlled thermostat as follows:

$$x'(t) = a \times K \times h - K \times x \tag{3}$$

Moreover, the design of PLC controlled thermostat is so simple that *Estimator* and *Adapter* are not necessary to be listed. Let *cold*, *comfortable* and *hot* be three symbols describing our feel, then *Characterizer* could be expressed as follows:

$$\begin{aligned} cold &\Leftrightarrow x \leq m \\ comfortable &\Leftrightarrow m < x < M \\ hot &\Leftrightarrow x \geq M \end{aligned} \tag{4}$$

Let *heat* and *halt* be two result strategies deduced from reasoner, then *Reasoner* could be

$$over\_cold \Leftrightarrow cold \wedge \neg comfortable \wedge \neg hot$$
$$over\_hot \Leftrightarrow \neg cold \wedge \neg comfortable \wedge hot \tag{5}$$

The effector translates the strategies into real numbers:

$$over\_cold \Rightarrow a = 1$$
$$over\_hot \Rightarrow a = 0 \tag{6}$$

The hierarchial formal description of the thermostat design is the conjunction of Equations (3), (4), (5) and (6), which is denoted by S_T. So an EDC formula $\lceil S_T \rceil$ expresses the formal design of thermostat is valid for all time.

We can prove a property that people feel comfortable with the thermostat controlled temperature. We formalize this property as follows.

Property 4.

$$\lceil S_T \rceil \Rightarrow \Box \lceil m \leq x \leq M \rceil$$

5. **Conclusion.** We have shown a unified formal method of modeling and validating the PLC system by EDC in a hierarchical way: The PLC system is translated into the EDC formulae hierarchically, the conjunction of the formal hierarchies expresses the invariant design over time, the properties of the system is also written by the EDC formulae, and then the verification procedure is performed under the EDC calculus. In future, we plan to implement our method by a theory prover.

Acknowledgment. This work is partly supported by Grants (HCIC201110) of Guangxi HCIC lab Open Fund, the Fundamental Research Funds for the Central Universities of Lanzhou University, No. 860772, and NSF of China No. 60973147, the Doctoral Fund of Ministry of Education of China under Grant No. 20090009110006 the NSF of Guangxi No. 2011GXNSFA018154 and 2012GXNSFGA060003, the Science and Technology Foundation of Guangxi No. 10169-1, and Guangxi Scientific Research Project No. 201012MS274.

REFERENCES

[1] M. Heiner, A petri net semantics for the plc language instruction list, *IEE Control*, pp.161-166, 1998.
[2] L. Holloway and B. Krogh, Synthesis of feedback control logic for a class of controlled Petri nets, *IEEE Trans. on Automatic Control*, vol.35, no.5, pp.514-523, 1989.
[3] J. G. Thistle and W. M. Wonham, Control problems in a temporal logic framework, *International Journal of Control*, vol.44, no.4, pp.943-976, 1986.
[4] K. Sacha, Verification and implementation of dependable controllers, *The 3rd International Conference on Dependability of Computer System DepCoS-RELCOMEX*, pp.143-151, 2008.
[5] R. Wang, X. Song et al., Timed automata based programmable logic controller code synthesis, *Computers in Industry*, pp.23-31, 2011.
[6] R. Wang, X. Song and M. Gu, Modeling and verification of program logic controllers with timed automata, *IET Proc. of Software*, pp.127-131, 2007.
[7] R. Šusta, *Verification of PLC Programs*, Ph.D. Thesis, CTU-FEE Prague, 2003.
[8] O. Müller and T. Stauner, Modelling and verification using linear hybrid automata – A case study, *MISC*, 1996.
[9] T. A. Henzinger, The theory of hybrid automata, *Tech. Rep. UCB/ERL M96/28*, EECS Department, University of California, Berkeley, http://www.eecs.berkeley.edu/Pubs/TechRpts/1996/3019.html, 1996.
[10] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine, The algorithmic analysis of hybrid systems, *Theoretical Computer Science*, vol.138, pp.3-34, 1995.

[11] H. Dierks, Synthesizing controllers from real-time specifications, *IEEE Trans. on CAD*, vol.18, no.1, pp.33-43, 1999.
[12] S. Nadjm-Tehrani and J. E. Stromberg, Formal verification of dynamic properties in an aerospace application, *Formal Methods in System Design*, vol.14, no.2, pp.135-169, 1999.
[13] A. Platzer, A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems, *Logical Methods in Computer Science*, vol.8, no.4, pp.1-44, 2012.
[14] A. P. Ravn, Design of embedded real-time computing systems, *Technical Report IDTR: 1995-170*, Dept. of Computer Science, Technical University of Denmark, 1995.
[15] M. Morin and S. Nadjm-Tehrani, Real-time hierarchical control, *IEEE Software*, vol.9, no.5, pp.51-57, 1992.
[16] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, 1999.
[17] *PID Controller*, Wikipedia, http://en.wikipedia.org/wiki/PID_controller.
[18] T. Wescott, PID without a PhD, *Embedded Systems Programming*, http://www.embedded.com/2000/0010/0010feat3.htm, 2000.
[19] K. K. Tan, Q.-G. Wang and C. C. Hang, *Advances in PID Control*, Springer-Verlag, 1999.
[20] *S7-200 Programmable Controller System Manual*, SIEMENS.