Computers and Chemical Engineering 97 (2017) 220–241

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Automata-based operating procedure for abnormal situation management in batch processes

Chun-Jung Wang, Yi-Chung Chen, Shih-Ting Feng, Chuei-Tin Chang*

Department of Chemical Engineering, National Cheng Kung University, Tainan 70101, Taiwan

ARTICLE INFO

Article history:
Received 29 March 2016
Received in revised form 18 October 2016
Accepted 29 November 2016
Available online 2 December 2016

Keywords:
Automata
Abnormal situation management
Batch processes
Emergency response procedure
Dynamic simulation

ABSTRACT

"Abnormal situation management" (ASM) in general refers to the various tasks required for online fault diagnosis and also hazard mitigation. Although quite a few ASM-related studies have already been carried out in the past, none of them addressed the wide range of issues consistently and rigorously with the same modeling tool. An automata-based strategy is therefore proposed in this work to synthesize all operating procedures needed for diagnostic tests and also other emergency response operations in the batch processes. The proposed model building techniques are suitable not only for characterizing all components in any given process, but also for representing the operation targets of all ASM tasks. Finally, notice that every resulting procedure can be readily expressed with an implementable sequential function chart (SFC).

1. Introduction

Generally speaking, the term "abnormal situation management" (ASM) refers to a collection of distinct tasks that must be performed online in a chemical plant for timely identification and mitigation of any significant departure of the system state from acceptable normal conditions (Bullemer and Nimmo, 1994; Nimmo, 1995). The scope of ASM primarily encompasses fault diagnosis and the subsequent emergency response operations. Yeílamos et al. (2009) tried in a pioneering work to dynamically integrate the conventional techniques for offline hazard analysis into ASM in continuous processes, while fault diagnosis and hazard mitigation in the *batch* plants obviously cannot be handled with the same approach. Although there have been a few related studies discussing various aspects of ASM for the batch processes (Chen et al., 2010; Yeh and Chang, 2011; Li et al., 2014), none of them addressed the wide range of issues consistently and thoroughly. In particular, notice that the diagnostic resolution may be further enhanced via test actions and, also, more flexible emergency response procedures could be synthesized either to maintain a lower but acceptable production rate after confirmation of the abnormal situation(s), or simply to bring the system to a shutdown state safely. A brief literature review of the related works is first presented in the sequel:

Online fault diagnosis of the batch processes has always been a popular research issue. Nomikos and MacGregor (1994, 1995) utilized the multi-way principal component analysis for batch process monitoring, which has later been extended for online diagnosis applications (Lee et al., 2004; Ruiz et al., 2001a,b; Undey et al., 2003; Hashizume et al., 2004; Pierri et al., 2008; Caccavale et al., 2009; Chen and Jiang. 2011). Other AI techniques, such as the artificial immune systems, artificial neural networks and knowledge-based expert systems (Dai and Zhao, 2011; Ghosh and Srinivasan, 2011; Tan et al., 2012; Zhao, 2014), have also utilized for identify fault origins in the batch plants. However, these approaches are mostly effective in systems with relatively few interconnected units and, moreover, the diagnostic resolution in systems with coexisting failures may not always be satisfactory.

To circumvent the above drawbacks, Chen et al. (2010) developed several Petri-net based algorithms to configure fault identification systems for more complex plants. Since the event sequences in multi-failure scenarios cannot be conveniently generated with the Petri-net models, their approach was limited to the single-failure scenarios. On the other hand, the automata were widely adopted as more appropriate models to circumvent this drawback (Debouk et al., 2000; Benveniste et al., 2003; Zad et al., 2003; Qiu and Kumar, 2006; Sköldstam et al., 2007; Malik et al., 2011). Gascard and Simeu-Abazi (2013) utilized the software UPPAAL to build diagnosers with timed automata, while Gomes Cabral et al. (2015) built diagnosers for discrete-event systems modelled with the finite-state automata.

* Corresponding author.
E-mail address: ctchang@mail.ncku.edu.tw (C.-T. Chang).

http://dx.doi.org/10.1016/j.compchemeng.2016.11.042
0098-1354/

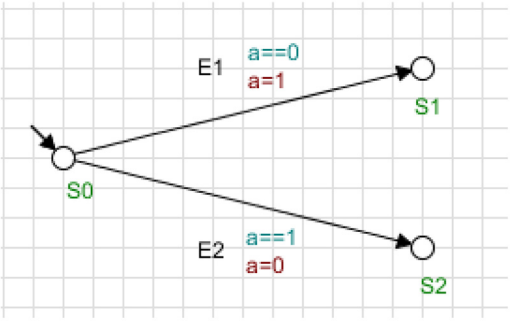

Fig. 1. Example of an EFA model.

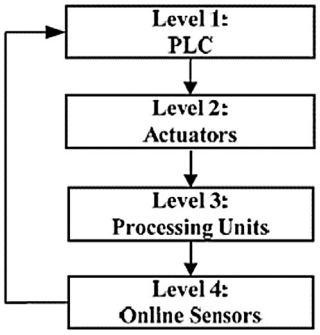

Fig. 2. Hierarchical structure of a batch process.

It should be noted that the aforementioned automata-based approaches preclude the diagnostic tests. Yeh and Chang (2011) developed a trial-and-error design procedure to introduce extra sensors and additional operating procedures into a batch process so as to improve its diagnostic performance, while Kang and Chang (2014) developed a systematic method with DESUMA (Ricker et al., 2006) to search for the optimal diagnostic test plans. It was found in the latter case that, for large systems, the required automata may be quite cumbersome. Finally, note that there have been relatively few published studies on the automata-based synthesis strategies for generating the emergency response procedures, e.g., see Yeh and Chang (2012) and Li et al. (2014).

To develop a consistent and comprehensive approach to ASM, a unified automata-based modeling strategy has been developed in this work to synthesize credible operating procedures needed for diagnostic tests and emergency responses. Specifically, the extended finite automata (EFA) have been adopted to facilitate model building and procedure synthesis with the free software SUPREMICA (Åkesson et al., 2006). The proposed ASM enabling methods can be divided into three groups, i.e., (1) automata building methods, (2) synthesis methods for stipulating the diagnostic test plans, and (3) synthesis methods for generating the emergency response procedures. These methods are presented sequentially in detail as follows.

2. The model building methods

2.1. Extended finite automata

To facilitate a clear description of the proposed model construction method, a brief review of the so-called extended finite automata (EFA) is given here. Let us first consider the standard structure of a deterministic automaton, which can be viewed as a six-tuple as follows:

$$A = (X, E, f, \Sigma, x_0, X_m) \tag{1}$$

where X is the set of system states; E is the event set; $f : X \times E \to X$ represents the state transition function; $\Sigma : X \to 2^E$ denotes the active event function and 2^E is the power set of E; x_0 is the initial system state; $X_m \subset X$ is the set of marked states. The function f can be viewed as a transition process (which is triggered by the feasible event $e \in E$) from state $x \in X$ to another state $x' \in X$, while the active event function Σ of state x is a set of corresponding active events.

The EFA is an improved version of the aforementioned standard structure. It is adopted in this study primarily for the purpose of managing large automata with existing software, e.g., SUPREMICA (Åkesson et al., 2006). To this end, each event in EFA is equipped with two extra auxiliary elements, i.e., *variable* and *guard*. The more specific explanations can be found in the sequel:

- An integer *variable* (with user-specified upper and lower bounds) can be used to update the equipment state after completing an event-driven transition. An example is shown in Fig. 1, in which variable "a" is updated to 1 (a = 1) via event *E1*. In the present study, the variables can be utilized to represent the states of processing units, e.g., the level, temperature or concentration of liquid in a tank.
- A *guard* is the sufficient condition of the corresponding state transition. Let us again consider Fig. 1 as an example and assume that the initial value of variable a is 0. Therefore, only event *E1* is permissible at the initial state S0 due to its guard "a = = 0" and, when S1 is reached after state transition, variable a should be updated to 1.

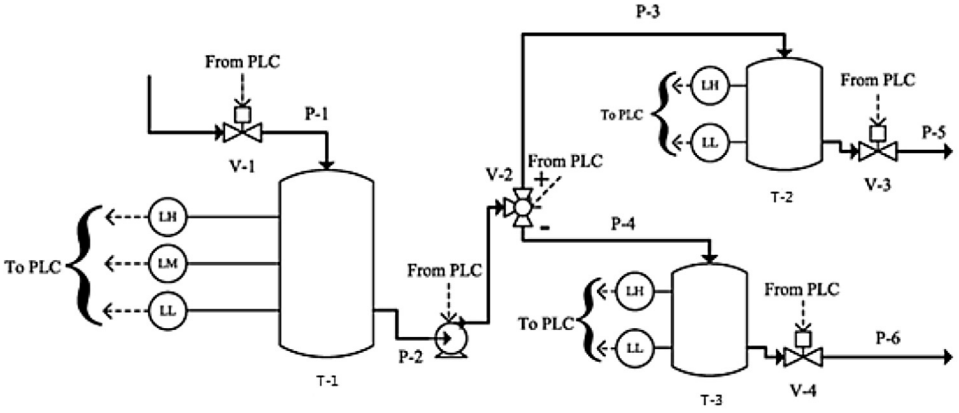

Fig. 3. Three-tank system.