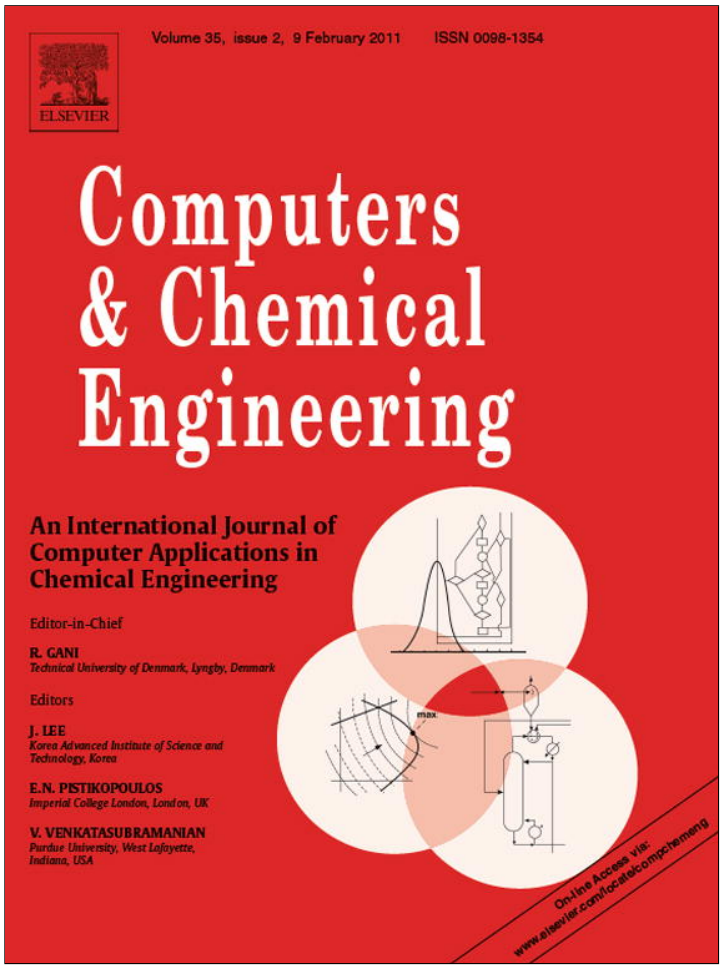

Computers and Chemical Engineering 35 (2011) 329–341

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Fault diagnosis with automata generated languages

Chuei-Tin Chang*, Chung Yang Chen

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

ARTICLE INFO

Article history:
Received 28 July 2009
Received in revised form 14 May 2010
Accepted 9 October 2010
Available online 3 November 2010

Keywords:
Safety
Chemical processes
Systems engineering
Simulation
Fault diagnosis
Automata

ABSTRACT

A SDG-based simulation procedure is proposed in this study to qualitatively predict the effects of one or more fault propagating in a given process system. These predicted state evolution behaviors are characterized with an automaton model. By selecting a set of on-line sensors, the corresponding diagnoser can be constructed and the diagnosability of every fault origin can be determined accordingly by inspection. Furthermore, it is also possible to define a formal diagnostic language on the basis of this diagnoser. Every string (word) in the language is then encoded into an IF-THEN rule and, consequently, a comprehensive fuzzy inference system can be synthesized for on-line diagnosis. The language generation steps are illustrated with a series of simple examples in this paper. The feasibility and effectiveness of this approach has been tested in extensive numerical simulation studies.

1. Introduction

The on-line fault diagnosis system has been widely recognized as an indispensable tool for enhancing process safety. For the purpose of building such systems, the qualitative cause-and-effect models, e.g., the signed directed graphs (SDGs), are used in the present study to characterize fault propagation mechanisms. The advantage of this modeling approach is mainly due to the fact that the causal relations in process systems can almost always be established on the basis of generic engineering principles, e.g., see Lapp and Powers (1977), Chang and Hwang (1992), Maurya, Rengaswamy, and Venkatasubramanian (2003a, 2003b) and Chen and Chang (2007). Notice that a wide variety of diagnosis methods has already been reported in this literature. Generally speaking, they could be classified into three distinct groups, i.e., the model based approaches, the knowledge based approaches, and the data-analysis based approaches (Venkatasubramanian, Rengaswamy, & Kavuri, 2003; Venkatasubramanian, Rengaswamy, Yin, & Kavuri, 2003). However, most of them require the measurement data and/or operational experiences obtained in the course of every possible accident. This requirement is often not satisfiable.

Although the causal models are easy to develop, it should be noted that they are basically static in nature. As a result, many available fault identification techniques are implemented on the basis of steady-state symptoms only, e.g., Maurya, Rengaswamy, and Venkatasubramanian (2006). However, the effects of fault(s) and/or failure(s) usually propagate throughout the entire system dynamically in sequence (Maurya, Rengaswamy, & Venkatasubramanian, 2007). A series of intermediate events may occur before the inception of catastrophic consequences. Thus, the performance of a diagnosis scheme should be evaluated not only in terms of its correctness but also its timeliness. To enhance the diagnostic efficiency, it becomes necessary to consider the precedence order (in time) between the fault propagation effects. Extensive studies have already been carried out in our previous works to develop fault identification techniques by incorporating both the eventual symptoms and also their *occurrence order* into a fuzzy inference system (FIS). This fault diagnosis approach has been applied successfully to a number of loop-free processes (Chang, Lin, & Chang, 2002) and also to systems with feedback and/or feed forward control loops (Chang & Chang, 2003; Chen & Chang, 2006, 2007).

Despite the fact that the diagnostic performance can be significantly improved with the aforementioned strategy, the representation, analysis and synthesis of inference systems are still very cumbersome. There is a need to develop a unified theoretical framework to extract the intrinsic properties of dynamic fault propagation behaviors in any given system. Our concern here is primarily with the sequence of states visited after the occurrence of fault origin(s) and the associated events causing the state transitions. Such fault propagation behavior can be described in terms of event sequences of the form $e_1\ e_2 \ldots e_n$. In the present work, a systematic procedure is developed to construct automata and language models for the purpose of representing these sequences accurately and succinctly. Consequently, the well-established discrete-event system theories can be adopted to analyze the event strings in fault propagation processes and, also, more compact inference rules can be produced accordingly. A series of examples are provided in this paper to illustrate the

* Corresponding author. Tel.: +886 6 2757575x62663; fax: +886 6 2344496.
E-mail address: ctchang@mail.ncku.edu.tw (C.-T. Chang).

0098-1354/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2010.10.003

proposed procedure and to demonstrate the effectiveness of the fault diagnosis strategy.

2. Automata construction

In order to represent the fault propagation behaviors in a given continuous chemical process with the proposed approach, an automaton model must be first constructed. This model can be simplified to form the so-called *diagnoser*, which is a special automaton with only observable events. A formal diagnostic language can then be generated accordingly for on-line implementation.

Following is a detailed description of the proposed automata construction procedure.

2.1. SDG-based simulation procedure

For the aforementioned model-building purposes, it is necessary to *predict* the system states visited in all possible fault propagation scenarios and also the corresponding state-transition processes. Although other qualitative models are equally acceptable, the SDG is adopted in the present study as the basis for simulating the effects of faults and failures. This selection is mainly due to the fact that the needed implementation procedure is conceptually straightforward.

It should be noted that, in a SDG model, the fault origins can usually be associated with the primal nodes, i.e., nodes with no inputs. In addition, a set of five values, i.e., $\{-10, -1, 0, +1, +10\}$, may be assigned to each node of the digraph to represent deviation from the normal value of corresponding variable. The value 0 represents the normal steady state. The negative values are used to denote the lower-than-normal states and the positive values signify the opposite. The magnitudes of non-zero deviations, i.e., 1 or 10, can be interpreted qualitatively as "small" and "large" respectively. The causal relation between two variables can be characterized with a directed arc and the corresponding gain. Again each gain may assume one of the five qualitative values, i.e., 0, ±1 and ±10. The output value of any arc can be computed with the gain and its input value according to the following equation:

$$v_{\text{out}} = \begin{cases} g \times v_{\text{in}} & \text{if } -10 \le g \times v_{\text{in}} \le +10 \\ +10 & \text{if } g \times v_{\text{in}} > +10 \\ -10 & \text{if } g \times v_{\text{in}} < -10 \end{cases} \quad (1)$$

where g, v_{in} and v_{out} denote respectively the gain, input and output values. It is obvious that the deviation values of all variables affected by one or more fault origin can always be computed with this approach, but the time at which each deviation occurs is indeterminable. Without the reference of time in the SDG-based simulation results, it can nonetheless be safely assumed that *the change in an input variable should always occur earlier than those in its outputs*. This is the basic assumption adopted for building the proposed automata.

2.2. System automata

A formal definition of a deterministic automaton A can be found in Cassandras and Lafortune (1999). Specifically, it is a six-tuple

$$A = (S, E, \Gamma, f, s_0, S_{\text{m}}) \quad (2)$$

where, S is the set of system states; E is the finite set of events associated with the transitions in A; $\Gamma : S \to 2^E$ is the active event function; $f : S \times E \to S$ is the transition function; s_0 is the initial system state; $S_{\text{m}} \subseteq S$ is the set of marked states. In the present application, each system state $s \in S$ is either a collection of all node values at a particular instance after an initiating failure occurs or the initial state itself. Every event $e \in E$ represents a previously nonexistent fault effect. The precedence order of these events is determined according to the basic assumption mentioned above. The active event function $\Gamma(\sigma)$ is used to specify the events which could change the system state s, while the transition function $f(s, e)$ is used for stipulating the resulting state caused by $e \in \Gamma(s)$. Finally, it should be noted that the initial state s_0 in this study is always associated with the *normal* condition and the set S_{m} contains the final steady states reached in all scenarios.

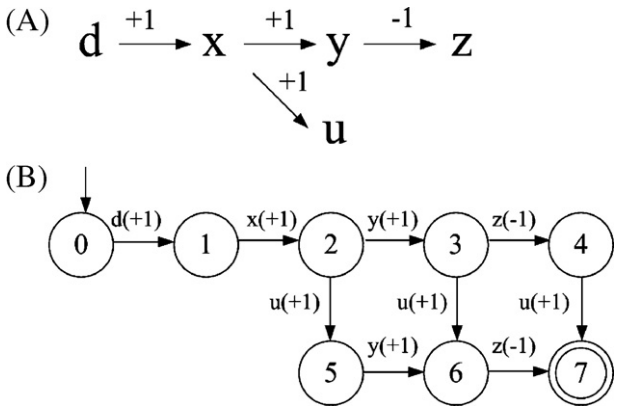

Fig. 1. (A) A tree-shaped SDG model; (B) the automaton resulted from $d(+1)$ in (A).

To facilitate illustration of the automaton construction steps, let us first consider the most fundamental digraph configuration, i.e., tree. More specifically, let us use the fictitious SDG model in Fig. 1(A) as an example and also assume that a positive deviation in the upstream variable d, i.e., $d(+1)$, is the only possible fault origin in this case. Notice that, although the precedence order of any two effects along the same branch path in this digraph can be uniquely identified with the proposed qualitative simulation procedure, the order of two distinct events located on *separate* branches should be considered as indeterminable. The corresponding automaton can thus be described with the state transition diagram presented in Fig. 1(B). Every system state here is characterized with a collection of the qualitative values of all variables in the digraph and these states are listed in Table 1A. It can be observed from Fig. 1(B) that state 0 is the initial state and 7 is the only marked state. Although the active event function $\Gamma(s)$ and the transition function $f(s, e)$ can be stipulated easily according to the aforementioned state transition diagram, the mappings defined by these two functions are still listed in Tables 1B and 1C respectively for clarity. Three possible event sequences between the initial and final system states can be identified from this automaton model, i.e.,

1. $d(+1)x(+1)y(+1)z(-1)u(+1)$;
2. $d(+1)x(+1)y(+1)u(+1)z(-1)$;
3. $d(+1)x(+1)u(+1)y(+1)z(-1)$

On the other hand, it should be noted that the same approach can also be adopted to construct automata for other typical digraph structures, e.g., feed forward loops (FFLs) and feedback loops (FBLs).

Table 1A
System states of the automaton in Fig. 1(B).

State	d	x	y	z	u
0	0	0	0	0	0
1	+1	0	0	0	0
2	+1	+1	0	0	0
3	+1	+1	+1	0	0
4	+1	+1	+1	−1	0
5	+1	+1	0	0	+1
6	+1	+1	+1	0	+1
7	+1	+1	+1	−1	+1

Table 1B
Active event function of the automaton in Fig. 1(B).

s (state)	$\Gamma(s)$ (active events)
0	$d(+1)$
1	$x(+1)$
2	$u(+1), y(+1)$
3	$u(+1), z(-1)$
4	$u(+1)$
5	$y(+1)$
6	$z(-1)$
7	–

Table 1C
Transition function of the automaton in Fig. 1(B).

s (original state)	e (active event)	$f(s,e)$ (resulting state)
0	$d(+1)$	1
1	$x(+1)$	2
2	$y(+1)$	3
2	$u(+1)$	5
3	$z(-1)$	4
3	$u(+1)$	6
4	$u(+1)$	7
5	$y(+1)$	6
6	$z(-1)$	7

Simple examples are presented in the sequel to illustrate the corresponding model-building steps:

In this study, a *feed forward loop* (FFL) is considered to be a collection of distinct paths in SDG with common starting and ending nodes. The FFLs can be found in numerous chemical processes, e.g., the feed forward control systems, the ratio control systems and various processing systems with parallel units, etc. To fix ideas, let us consider the fictitious SDG in Fig. 2(A) as an example. The feed forward loop in this case contains two paths, i.e. (1) $x \to y \to z$ and (2) $x \to u \to v \to z$. Notice that the products of the edge gains along these two paths can be found to be −1 and +1 respectively. Consequently, this FFL is also referred to as *negative* feed forward loop (NFFL). For illustration convenience, let us again assume that there is only one possible fault origin $d(+1)$ and its effects propagate along separate paths of NFFL *independently*. Since the SDG is essentially a static model, it is not possible to tell which effect reaches the ending node z first. The corresponding automaton can be described with the state transition diagram given in Fig. 2(B). In this diagram, the events $z^{(1)}(-1)$ and $z^{(2)}(+1)$ represent the changes in variable z caused by disturbances propagating along paths (1) and (2) respectively. These conflicting effects on the same variable are reconciled in this work according to the generic rules listed in Table 2. Notice that rules 4–6, 8, 9 and 13 yield more than one outcome. In actual applications, it may be possible to assign a definite value to the net effect in each scenario with additional process knowledge on a case-by-case basis. In the present example, let us determine the automaton states by applying the reconciliation rules only (see Table 3). Notice that the net outcomes of opposite effects (+1 and −1) on z must be evaluated for states 16, 17, 20 and 21. According to rule 8 in Table 2, there are three possibilities in each of these four cases. Notice also that w may also assume three qualitative values in state 21 and, thus, there may be 9 distinct combinations. However, since z is the ending node of a NFFL and w is its output, the reconciled states must be consistent with the edge gain (+1) between them. In other words, the deviation values of z and w should be of the same sign and there can be only 3 possibilities for state 21.

Table 2
Reconciliation rules.

Rule	Individual effects	Net effect
1	+10, +10	+10
2	+10, +1	+10
3	+10, 0	+10
4	+10, −1	+10/+1
5	+10/−10	+10/+1/0/−1/−10
6	+1, +1	+10/+1
7	+1, 0	+1
8	+1, −1	+1/0/−1
9	+1, −10	−1/−10
10	0, 0	0
11	0, −1	−1
12	0, −10	−10
13	−1, −1	−1/−10
14	−1, −10	−10
15	−10, −10	−10

Table 3
System states in Fig. 2(B).

State	d	x	y	u	v	z	w
0	0	0	0	0	0	0	0
1	+1	0	0	0	0	0	0
2	+1	+1	0	0	0	0	0
3	+1	+1	+1	0	0	0	0
4	+1	+1	+1	0	0	−1	0
5	+1	+1	+1	0	0	−1	−1
6	+1	+1	0	+1	0	0	0
7	+1	+1	+1	+1	0	0	0
8	+1	+1	+1	+1	0	−1	0
9	+1	+1	+1	+1	0	−1	−1
10	+1	+1	0	+1	+1	0	0
11	+1	+1	+1	+1	+1	0	0
12	+1	+1	+1	+1	+1	−1	0
13	+1	+1	+1	+1	+1	−1	−1
14	+1	+1	0	+1	+1	+1	0
15	+1	+1	+1	+1	+1	+1	0
16	+1	+1	+1	+1	+1	+1/0/−1	0
17	+1	+1	+1	+1	+1	+1/0/−1	−1
18	+1	+1	0	+1	+1	+1	+1
19	+1	+1	+1	+1	+1	+1	+1
20	+1	+1	+1	+1	+1	+1/0/−1	+1
21	+1	+1	+1	+1	+1	+1/0/−1	+1/0/−1

A feedback loop (FBL) in this work is essentially a path in digraph on which the starting and ending nodes coincide. If the product of all edge gains on the loop is negative, it is referred to as a *negative* feedback loop (NFBL). It is in general very difficult to fully simulate the transient behavior of a NFBL on the basis of SDG model alone. To illustrate this point, let us consider the effects of disturbance $d_x(+1)$ on the example system in Fig. 3(A). It is obvious that the incipient event sequence can be determined according to Eq. (1), i.e., $d_x(+1)x(+1)y(+1)z(+1)u(+1)$. However, since the net effect of two simultaneous inputs, i.e., $d_x(+1)$ and $u(+1)$, on x is uncertain afterwards according to Table 2, the subsequent event sequence is really indeterminable without further quantitative and/or qualitative knowledge of the physical system in question. Although the transient response of disturbance $d_x(+1)$ cannot be properly predicted, it is sometimes possible to determine the final steady-state values of loop variables *a priori*. In particular, if the NFBL in Fig. 3(A) is a *control* loop, these final values can be assigned by following the approach proposed by Ju, Chen, and Chang (2004). Table 4 is a complete listing of final states of this standard control NFBL resulting from various disturbances. Thus, one can express the fault propa-

Table 4
Steady-state values of loop variables in a standard control NFBL.

Fault origin	x	y	z	u
$d_x(+1)$	0	0	+1	+1
$d_y(+1)$	−1	0	+1	+1
$d_z(+1)$	0	0	0	0
$d_u(+1)$	0	0	−1	0

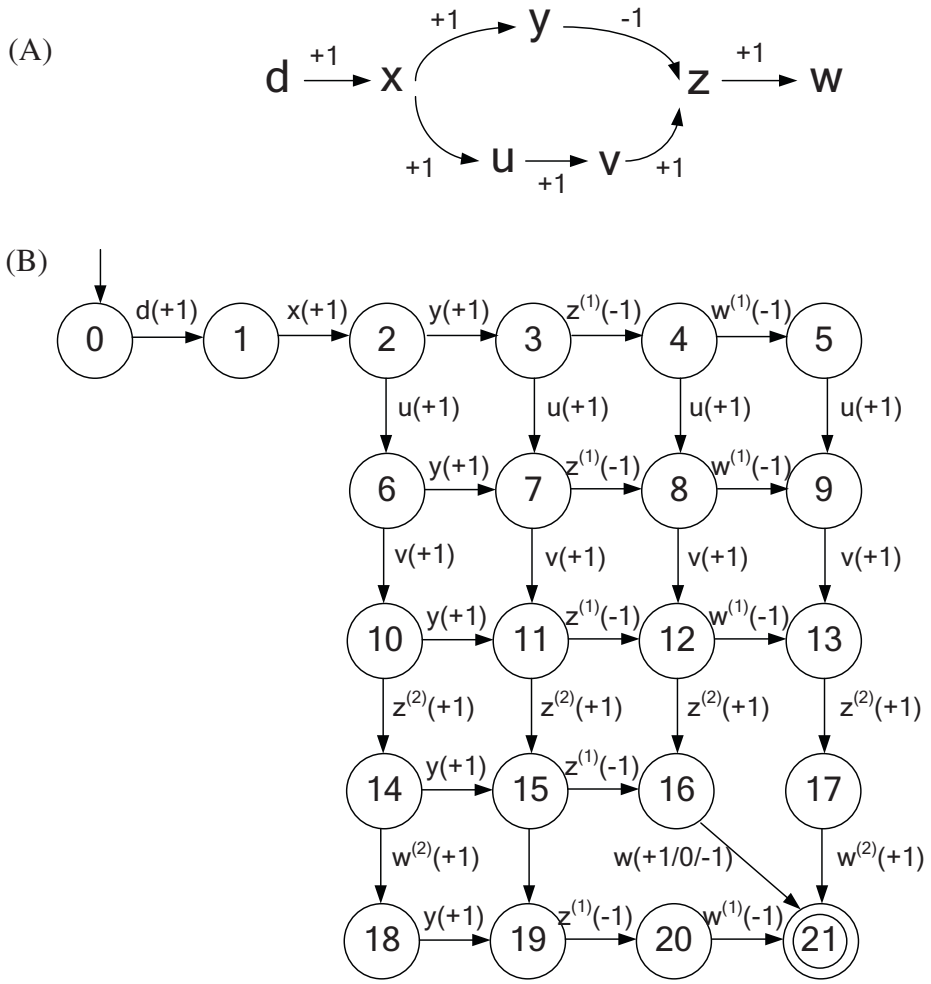

Fig. 2. (A) A SDG model with negative feed forward loop; (B) the automaton resulted from $d(+1)$ in (A).

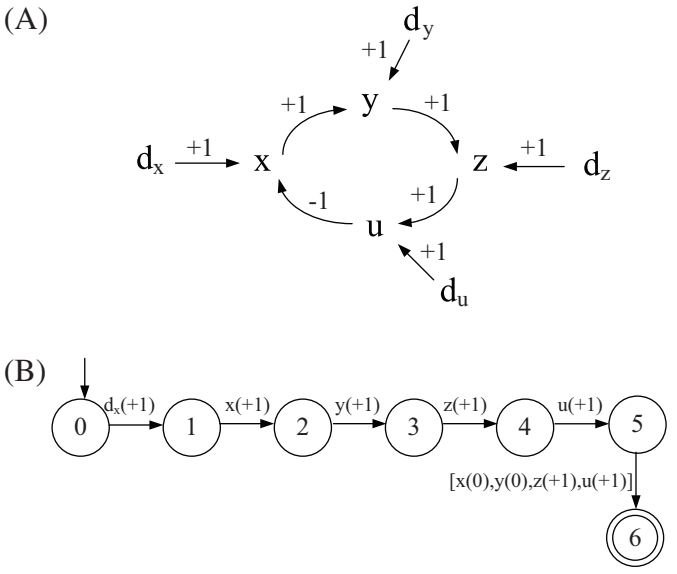

Fig. 3. (A) A SDG model with negative feedback loop; (B) the automaton resulted from $d_x(+1)$ in (A).

gation behavior caused by $d_x(+1)$ as the following event sequence

$$d_x(+1)x(+1)y(+1)z(+1)u(+1)\,[x(0)y(0)z(+1)u(+1)]$$

Notice that the final states of all loop variables are lumped into a single event in a square bracket and their precedence order is left unspecified. This is due to the difficulties in detecting the occurrence order of these symptoms in real time. Finally, it should be noted that the final steady states of *process* NFBLs can only be identified on a case-by-case basis and numerous examples have already been reported in the literature (Chen & Chang, 2007; Maurya, Rengaswamy, & Venkatasubramanian, 2004, 2006; Oyeleye & Kramer, 1988).

The automaton resulting from a "large" disturbance can be obtained by following the identical procedure described above if fault propagation is immediate. In the case of the tree-shaped SDG in Fig. 1(A), the corresponding automaton can be obtained simply by changing the magnitude level of deviations in Fig. 1(B) from 1 to 10. However, if a finite time constant is needed to characterize the transient response of an output variable to the disturbance in its input and, also, its direction remains unchanged most of the time during the time window of interest, then an auxiliary assumption must be introduced to facilitate a more accurate description, i.e., *the smaller deviation of a process variable must occur before reaching a larger one of the same variable*. Thus, the automaton in Fig. 1(B) should be further revised to incorporate this requirement (see Fig. 4). The same approach has been adopted to construct automata

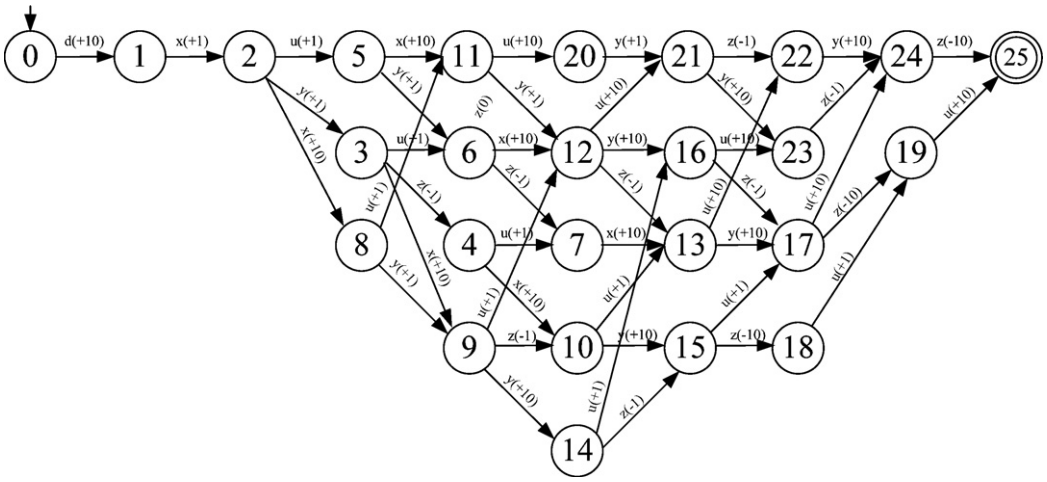

Fig. 4. The automaton resulted from $d(+10)$ in Fig. 1(A).

for fault origins with magnitude 10 in other diagraph configurations. For the sake of brevity, these results are not included in the present paper.

2.3. Diagnoser and diagnosability

In realistic applications, the fault origins (i.e., failures or upsets) and some of the process variables cannot be monitored on-line. Thus, the event set of an automaton model can be further divided into two subsets, i.e., $E = E_o \cup E_{uo}$, where E_o and E_{uo} denote the sets of observable and unobservable events respectively. To check diagnosability of each fault origin under consideration and also facilitate diagnostic inference with the available sensors, the system automaton A should be converted to a *diagnoser* A_{diag}, which is in essence a transformed automaton with E_o as its event set. Although a construction procedure has already been developed by Sampath, Sengupta, Lafortune, Sinnamohideen, & Teneketzis (1996) for the discrete event systems in general, the diagnosers for the present applications are built with an intuitive but more convenient alternative approach. Specifically, if a state is reached immediately after an unobservable event, then this state is merged with its predecessor(s) in the original automaton model. For example, let us assume that $d(+1)$ is the fault origin and $y(+1)$ is not observable in Fig. 1(B). The corresponding diagnoser can be easily obtained by applying this principle (see Fig. 5). The numerical node labels here are the same as those in the original automaton, while the subscript of each label is used to reflect whether or not the fault origin has occurred when the corresponding state is reached. More specifically, the subscripts Y and N are used to represent "yes" and "no" respectively.

If one or more NFFL is present in the digraph model, the *observed* effects on the variables corresponding to the ending node(s) and its outputs must be reconciled in the diagnoser according to Table 2. Let us use Fig. 2(B) as an example to illustrate this point. By assuming that only y, z and u are monitored on-line, this automaton can be converted to the diagnoser shown in Fig. 6. It should be noted that the net effects on z are treated as events in this automaton.

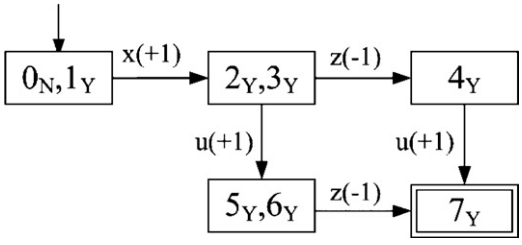

Fig. 5. A diagnoser obtained by assuming event $y(+1)$ in Fig. 1(B) is unobservable.

Finally, notice that the aforementioned diagnoser construction practice is applicable even when multiple scenarios are possible. As an example, let us consider the SDG model in Fig. 3(A) and assume that there are two measured variables, i.e., y and z, and four potential faults, i.e., $d_x(+1)$, $d_y(+1)$, $d_z(+1)$ and $d_u(+1)$. The automaton model of this system and the corresponding diagnoser can be found in Fig. 7(A) and (B) respectively. Obviously, the issue of diagnosability becomes important in this situation. Although the formal necessary and sufficient conditions of system diagnosability has been derived and proven rigorously by Sampath, Sengupta, Lafortune, Sinnamohideen, and Teneketzis (1995), the identifiability of each fault origin in our studies can be determined simply by inspecting the diagnoser. In particular, the diagnosability of a given fault origin is established if it is the unique cause of at least one diagnoser state. Otherwise, the corresponding fault propagation scenario should be indistinguishable from other possibilities. On the basis of this criterion, one can clearly observe from Fig. 7(B) that $d_z(+1)$ and $d_u(+1)$ are both diagnosable while the observable fault propagation behaviors of $d_x(+1)$ and $d_y(+1)$ are indistinguishable. The feasibility of this simple checking procedure is attributed mainly to the fact that the automata used in the present applications form a special subclass of those for modeling the discrete event systems. More specifically, since the *continuous* chemical processes are considered in this work, the following unique features can always be identified in corresponding automata:

1. The initial automaton state is always associated with the normal system condition.
2. Every initial state transition is triggered by failure event(s).
3. Recurrence of system state is not possible, i.e., the automaton is free of any feedback loop. Notice that this feature is due to our assumption that a final steady state is reachable in every possible scenario.

3. Language generation

A *language* $\mathfrak{L}$ is regarded in this work as a collection of finite-length event sequences. These sequences are referred to as *strings* or *words*. The set of all possible events is of course E in Eq. (2). An additional set E^* is also introduced here to include all possible *strings* (including the empty string ε) constructed over E. Thus, it is obvious that $\mathfrak{L} \subseteq E^*$.

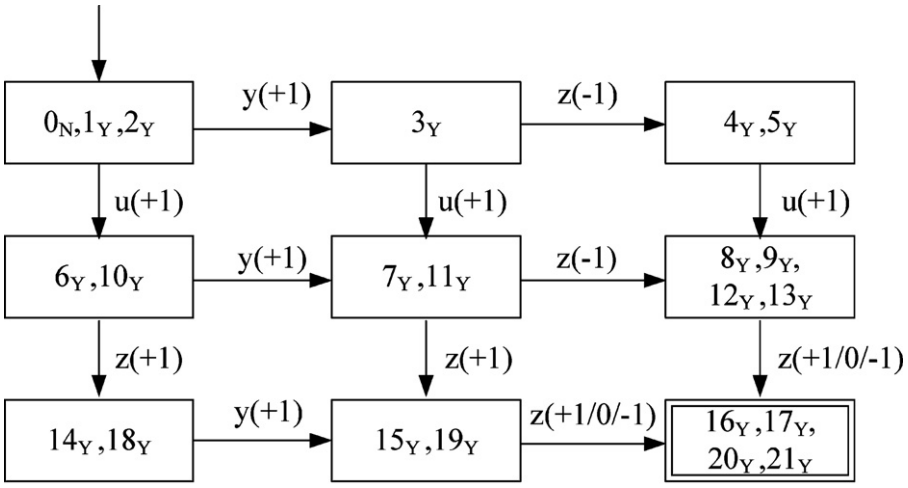

Fig. 6. A diagnoser obtained from the automaton in Fig. 2(B) with y, z and u as the measured variables.

Since fault diagnosis can only be performed according to the online symptoms, i.e., the events in E_o, the aforementioned automaton A_{diag} (not A) is thus used to generate a diagnostic language for the purpose of enumerating all *observable* event sequences caused by a given fault origin. Specifically,

$$\mathfrak{L}(A_{\text{diag}}) = \{t \in E* \mid f(s_0, t) \text{ is defined by } A_{\text{diag}}\} \tag{3}$$

The transition function $f(s_0, t)$ here can be evaluated recursively according to the following rules:

$$f(s, \varepsilon) = s \tag{4}$$

$$f(s, te) = f(f(s, t), e) \tag{5}$$

where, $t \in E^*$ and $e \in E$. In addition, the *marked* language of automaton A_{diag} can be defined as

$$\mathfrak{L}_{\text{m}}(A_{\text{diag}}) = \{t \in \mathfrak{L}(A_{\text{diag}}) \mid f(s_0, t) \in S_{\text{m}}\} \tag{6}$$

In essence, an automaton-based language can be synthesized by first identifying the longest strings and then obtaining all their *prefixes*. Since the marked states in the present application are always terminal, $\mathfrak{L}(A_{\text{diag}})$ can be produced by taking the prefix closure of

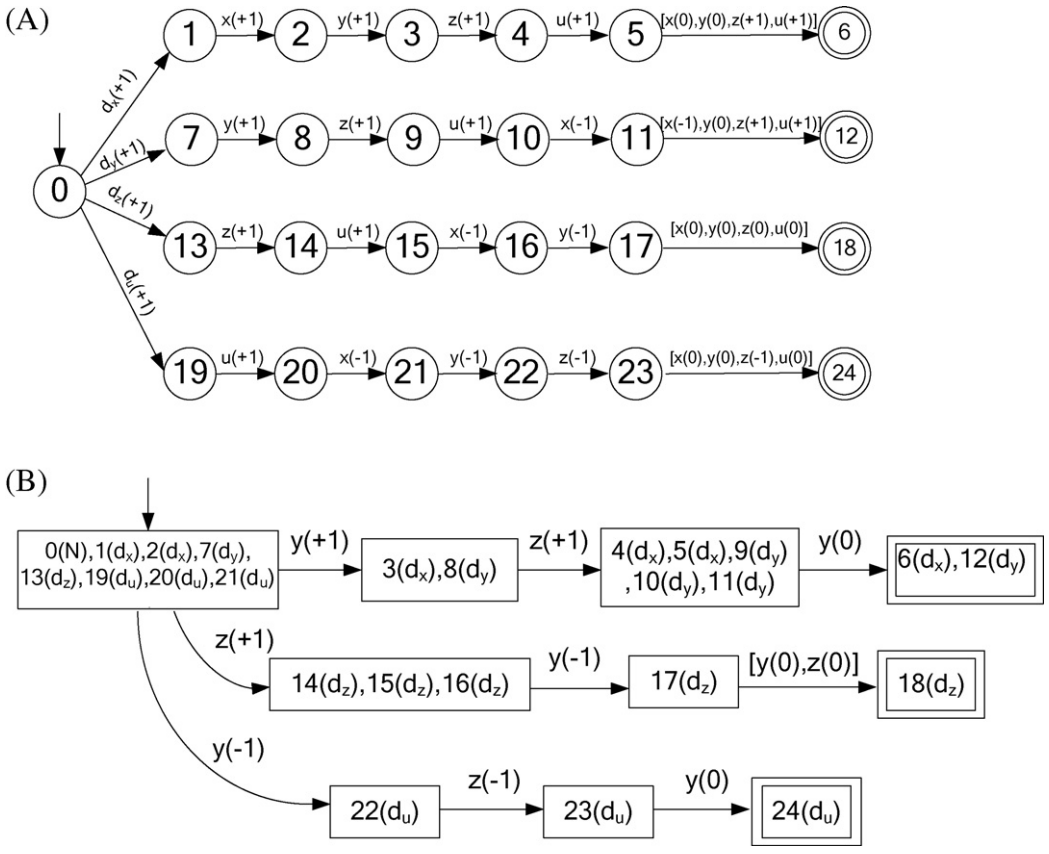

Fig. 7. (A) The automaton resulted from four different fault origins, i.e., $d_x(+1)$, $d_y(+1)$, $d_z(+1)$ and $d_u(+1)$, in Fig. 3(A); (B) a diagnoser obtained from the automaton in (A) with y and z as the measured variables.

$\mathfrak{L}_m(A_{diag})$ (Cassandras & Lafortune, 1999), i.e.,

$$\mathfrak{L}(A_{diag}) = \overline{\mathfrak{L}_m(A_{diag})} \tag{7}$$

where, $\overline{\mathfrak{L}_m(A_{diag})}$ denotes the set of all prefixes of the strings in $\mathfrak{L}_m(A_{diag})$. From Eq. (7), it can be shown that every diagnoser considered in this study must be *nonblocking*, i.e., any string $t \in \mathfrak{L}(A_{diag})$ can be always extended by another string t' such that $tt' \in \mathfrak{L}_m(A_{diag})$.

Let us use the diagnoser in Fig. 5 as an example to illustrate the aforementioned approach. The two languages marked and generated respectively by A_{diag} in this case should be

$$\mathfrak{L}_m(A_{diag}) = \{x(+1)z(-1)u(+1),\ x(+1)u(+1)z(-1)\} \tag{8}$$

$$\mathfrak{L}(A_{diag}) = \{\varepsilon, x(+1), x(+1)z(-1), x(+1)u(+1), x(+1)z(-1)u(+1), x(+1)u(+1)z(-1)\} \tag{9}$$

As another example, the diagnoser in Fig. 6 yields the following marked language:

$$\mathfrak{L}_m(A_{diag}) = \left\{ \begin{array}{l} y(+1)z(-1)u(+1)z(+1/0/-1), y(+1)u(+1)z(-1)z(+1/0/-1), \\ u(+1)y(+1)z(-1)z(+1/0/-1), u(+1)z(+1)y(+1)z(+1/0/-1), \\ u(+1)y(+1)z(+1)z(+1/0/-1), y(+1)u(+1)z(+1)z(+1/0/-1) \end{array} \right\} \tag{10}$$

The corresponding diagnostic language can be obtained by applying Eq. (7). Finally, if the possibilities of multiple fault origins are incorporated in a diagnoser, then there is a need to generate a sublanguage specific to every fault origin, i.e.,

$$\mathfrak{L}(A_{diag}) = \bigcup_i \mathfrak{L}(A_{diag}^{F_i}) \tag{11}$$

where, $A_{diag}^{F_i}$ is an automaton obtained by removing all the abnormal states in A_{diag} which are *not* caused by the ith fault origin F_i ($i = 1, 2, \ldots$). The marked sublanguages of the fault origins in Fig. 7(B) can be easily produced with this method, i.e.,

$$\mathfrak{L}_m(A_{diag}^{d_x(+1)}) = \mathfrak{L}_m(A_{diag}^{d_y(+1)}) = \{y(+1)z(+1)y(0)\} \tag{12}$$

$$\mathfrak{L}_m(A_{diag}^{d_z(+1)}) = \{z(+1)y(-1)[y(0), z(0)]\} \tag{13}$$

$$\mathfrak{L}_m(A_{diag}^{d_u(+1)}) = \{y(-1)z(-1)y(0)\} \tag{14}$$

The corresponding diagnostic sublanguages and language can be constructed with Eqs. (7) and (11).

4. Fuzzy inference system

Every string in $\mathfrak{L}(A_{diag})$ is encoded with an IF-THEN rule in this study. These rules can be incorporated in a fuzzy inference system

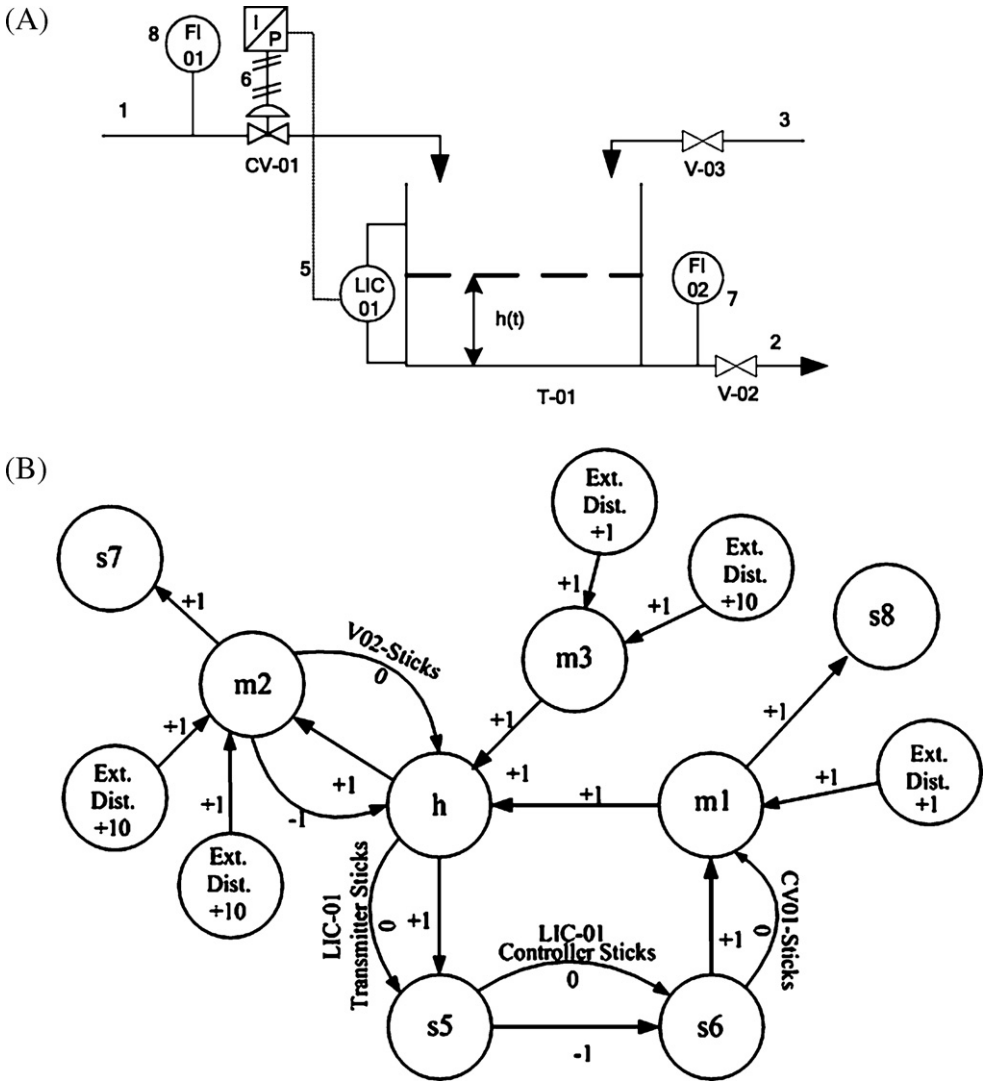

Fig. 8. (A) The process flow diagram of a single-tank storage system with feedback level-control loop (Example 1); (B) the SDG model of level-control system in Example 1.

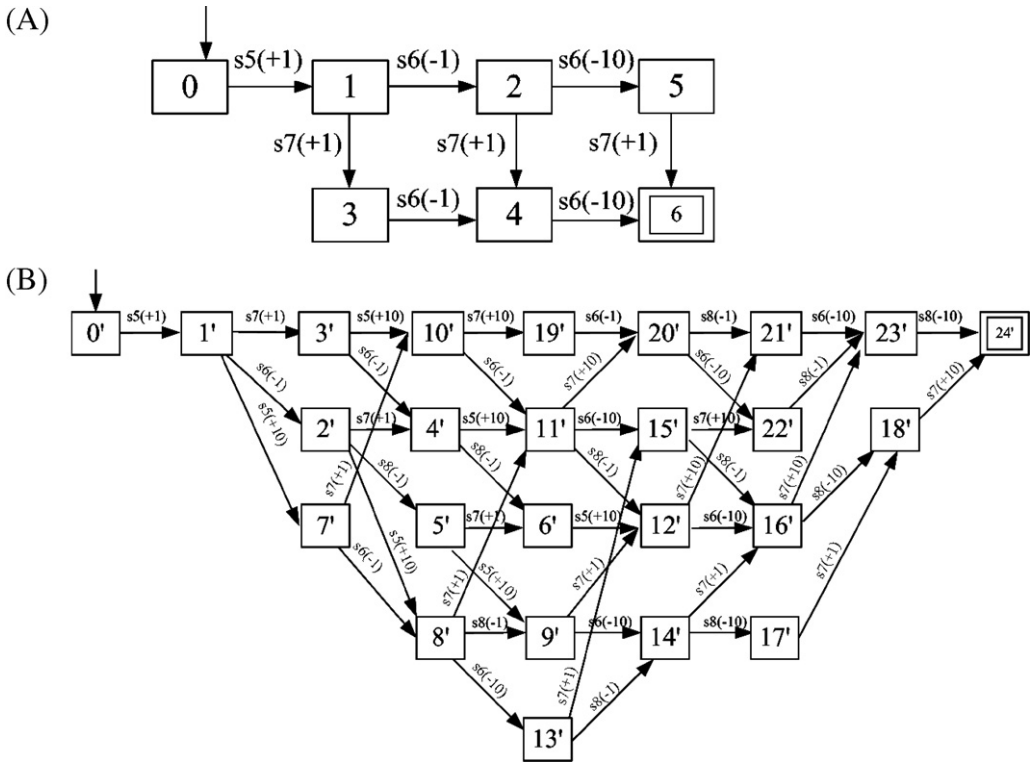

Fig. 9. The diagnoser constructed with measurement signals $s5$, $s6$, $s7$ and $s8$ in Example 1.

to evaluate the *existence potential* of the corresponding fault origin. In particular, if at least one event sequence in the marked sublanguage $\mathfrak{L}_m(A^{F_i}_{diag})$ can be confirmed, then it is highly possible that they are caused by the corresponding fault origin F_i. To assert such a belief, the fuzzy conclusion "o_i is OCR" is adopted in the inference rule, where OCR is the linguistic value of the *occurrence index* o_i reflecting the highest confidence level in confirming the existence of F_i. More specifically, this rule can be written as

$$\text{IF } t_o \in \mathfrak{L}_m(A^{F_i}_{diag}) \text{ THEN } o_i = \text{OCR} \tag{15}$$

where t_o denotes the observed event string. On the other hand, it is certainly reasonable to disregard the possibility of a fault if none of

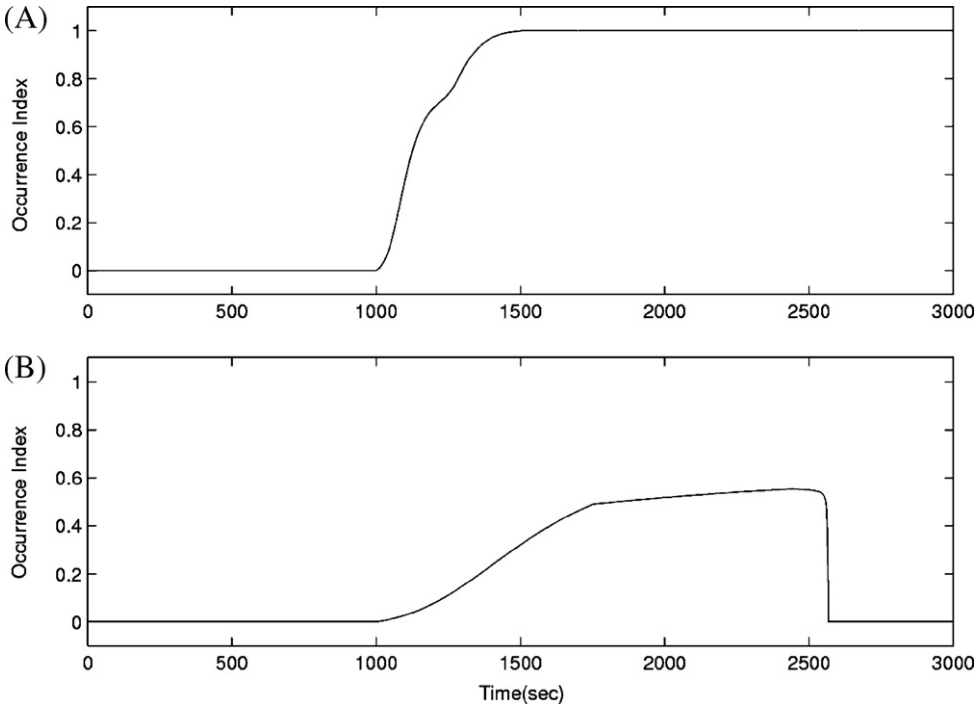

Fig. 10. Diagnosis results of two different scenarios in the level control system described in Example 1. (A) Occurrence index of $m3(+10)$ using simulation data obtained by introducing the same event; (B) occurrence index of $m3(+10)$ using simulation data obtained by introducing the basic events in the second scenario, i.e., $m3(+1)$ and CV-01 sticks.

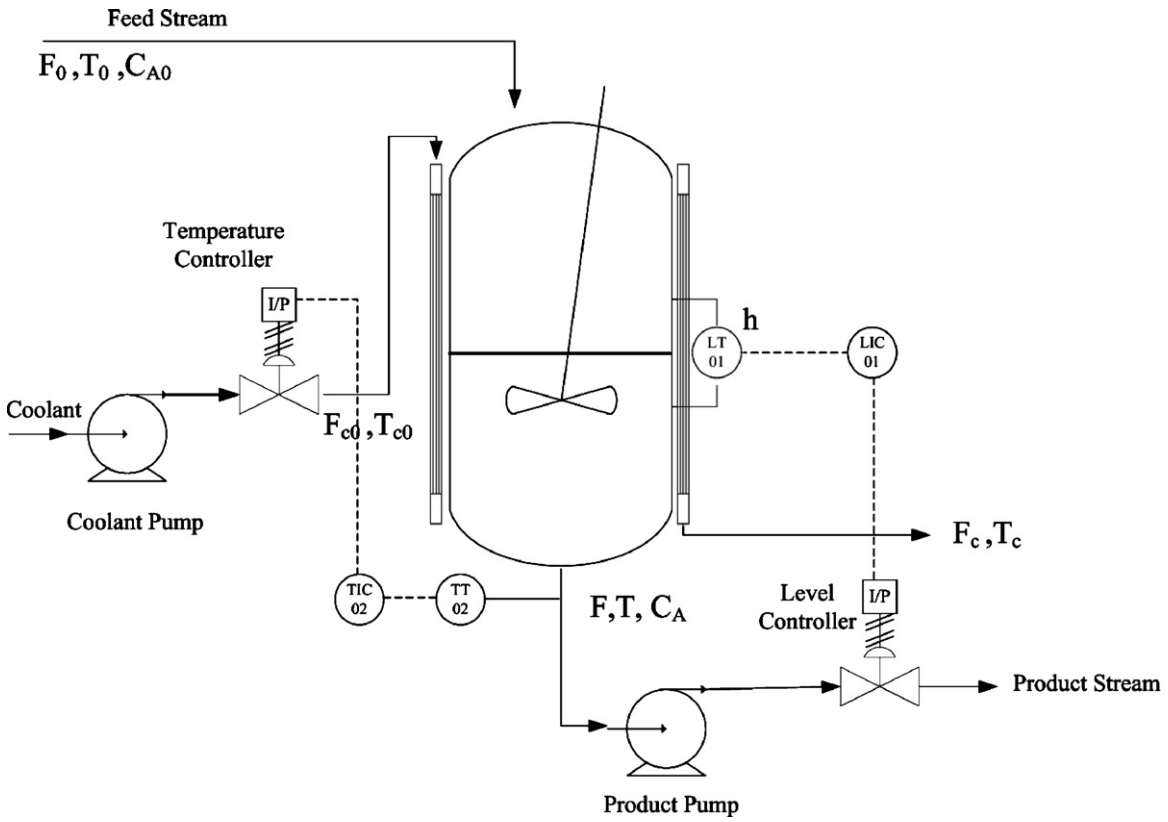

Fig. 11. The process flow diagram of a CSTR reactor system with level and temperature control loops.

the corresponding event strings in $\mathfrak{L}(A^{F_i}_{\text{diag}})$ can be observed. Thus, the diagnosis for this scenario should be "o_i is NOC", where NOC is the linguistic value representing the lowest level of confidence. In other words,

$$\text{IF } t_o \notin \mathfrak{L}(A^{F_i}_{\text{diag}}) \text{ THEN } o_i = \text{NOC} \tag{16}$$

The diagnostic conclusion for string $t_o \in \mathfrak{L}(A^{F_i}_{\text{diag}}) \setminus \mathfrak{L}_m(A^{F_i}_{\text{diag}})$ is UCT_ℓ, i.e., uncertain with confidence level ℓ. Thus, this rule should be written as

$$\text{IF } t_o \in \mathfrak{L}(A^{F_i}_{\text{diag}}) \setminus \mathfrak{L}_m(A^{F_i}_{\text{diag}}) \text{ THEN } o_i = \text{UCT}_\ell \tag{17}$$

In this study, the confidence level ℓ in confirming the existence of the root cause(s) is assumed to be proportional to the string length. The highest possible confidence level is of course assigned to the strings in $\mathfrak{L}_m(A^{F_i}_{\text{diag}})$.

The aforementioned IF-THEN rules can be implemented with a two-layer fuzzy inference framework (Chen & Chang, 2006). A brief description of this approach is presented in Appendix A for the sake of completeness.

5. Examples

The examples provided here are used only to demonstrate the feasibility of the automata-based approach for generating diagnostic languages in typical chemical engineering systems. Additional and more complex case studies can be found elsewhere (Chen & Chang, 2006, 2007, 2008).

5.1. Example 1

Let us consider the level control system presented in Fig. 8(A) and its digraph model in Fig. 8(B). All on-line signals in this system (i.e., *s*5, *s*6, *s*7 and *s*8) are assumed to be available for fault diagnosis. For illustration convenience, only two possible scenarios are studied here, i.e. (1) an uncontrollable increase in the flow rate of stream 3 and (2) a moderate (controllable) increase in the flow rate of stream 3 while control valve CV-01 sticks. The corresponding diagnoser can be found in Fig. 9. Notice that this automaton is presented in two parts for clarity. States 0 and 0′ are used to represent the combined states of the normal condition and the system condition reached immediately after the occurrence of fault origin in scenario 1 and scenario 2 respectively. These two states, i.e., 0 and 0′, should be lumped into a single one in the actual diagnoser.

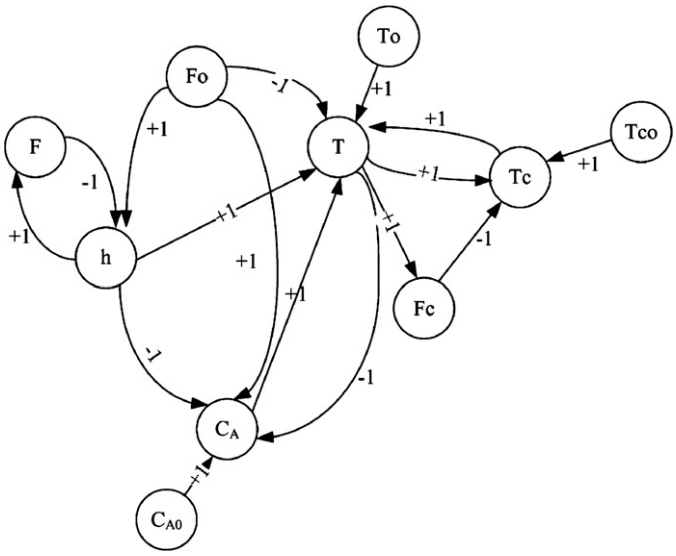

Fig. 12. The SDG model of CSTR reactor system.

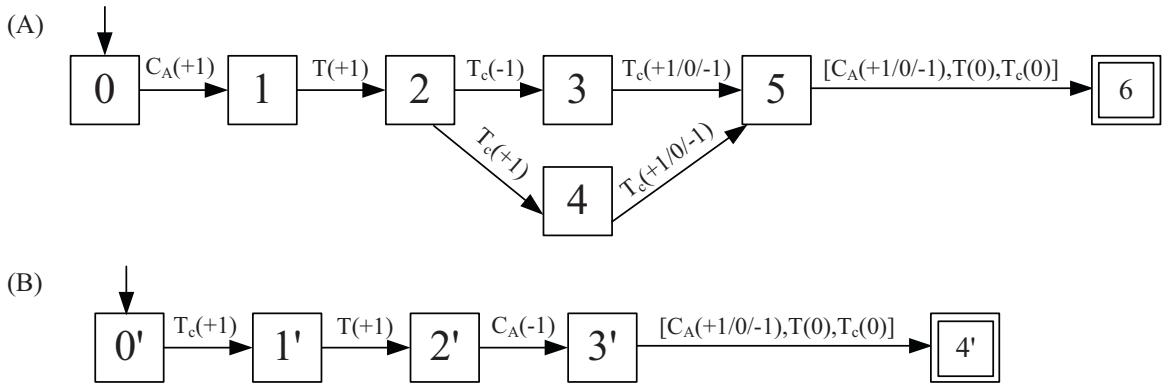

Fig. 13. The diagnoser constructed with on-line measurements of C_A, T and T_c in Example 2.

In this example, it is assumed that the height of tank wall is 100 cm and the outlet flow rate is proportional to the square root of liquid-level height. To carry out numerical simulation studies, it is also assumed that the system is initially at the normal steady state when the level height is 50 cm and the flow rates of both stream 1 and stream 2 are the same (707 cm^3/s). The valve on stream 3 is normally kept closed at any time. A PI controller was used in this case and its parameters were chosen with the IMC design method, i.e., $K_c = 1.4145$ and $\tau_I = 1436.2$.

The first scenario was simulated by introducing an additional flow rate of 900 cm^3/s in stream 3 at 1000 s during simulation run. The occurrence index of the event "$m3(+10)$" can be found in Fig. 10(A). It can be observed that the diagnosis is clearly swift and quite accurate. Specifically, the existence of fault origin is detected almost immediately and fully confirmed at about 500 s after its introduction. On the other hand, it should be noted that the second fault propagation scenario was also simulated by fixing the position of control valve CV-01 at 750 s and then introducing an additional flow rate of 200 cm^3/s in stream 3 at 1000 s. The occurrence index of the incorrectly assumed fault origin "$m3(+10)$" is presented in Fig. 10(B). Notice that the nonzero occurrence index in the period between 1000 and 2600sec can be attributed to the fact that the observed event strings caused by the two fault origins can be matched partially during the initial stage. More specifically, the set of matched strings is

$$\{\overline{s5(+1)s6(-1)s7(+1)}, \overline{s5(+1)s7(+1)s6(-1)}\}$$

As the on-line symptoms developed further, none of the longer strings generated by the automaton in Fig. 9(B) can be used to characterize the measurement data obtained after 2600 s and thus the occurrence index drops to zero abruptly.

5.2. Example 2

The final example discussed in this paper is concerned with an exothermic CSTR reactor with its temperature and level control loops (see Fig. 11). It is assumed that there are three measurable process variables, the temperature of cooling water (T_c), the temperature and reactant concentration at the outlet of CSTR (T and C_A). For simplicity, it is further assumed that the variations in these variables are always accurately reflected in their measurements and,

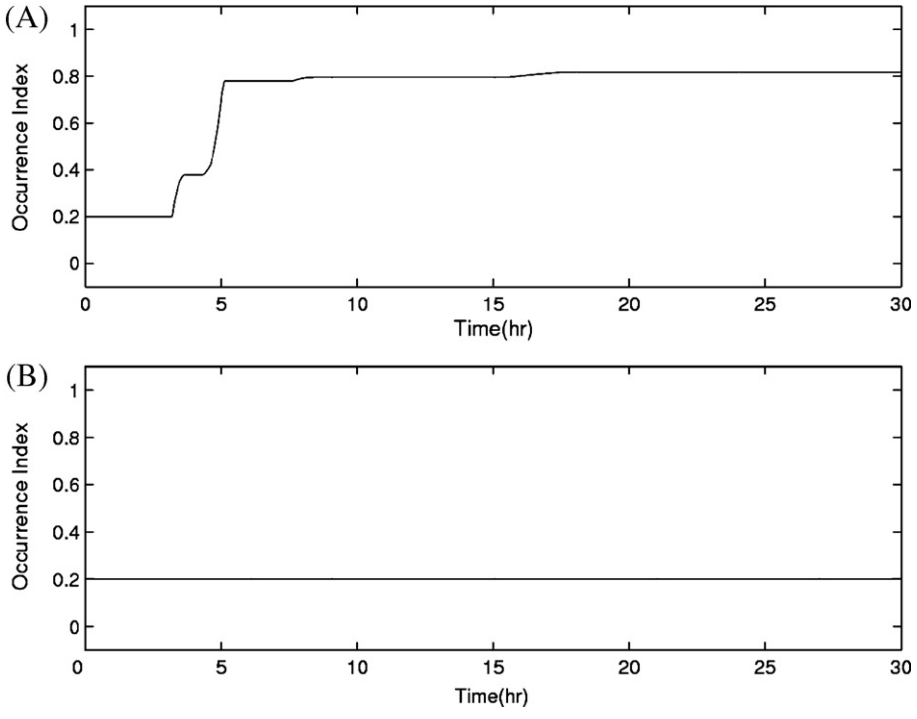

Fig. 14. The diagnosis results obtained with a two-layer inference system concerning the assumed fault origin $C_{A0}(+1)$. (A) Actual fault origin used in numerical simulation: $C_{A0}(+1)$; (B) actual fault origin used in numerical simulation: $T_{c0}(+1)$.

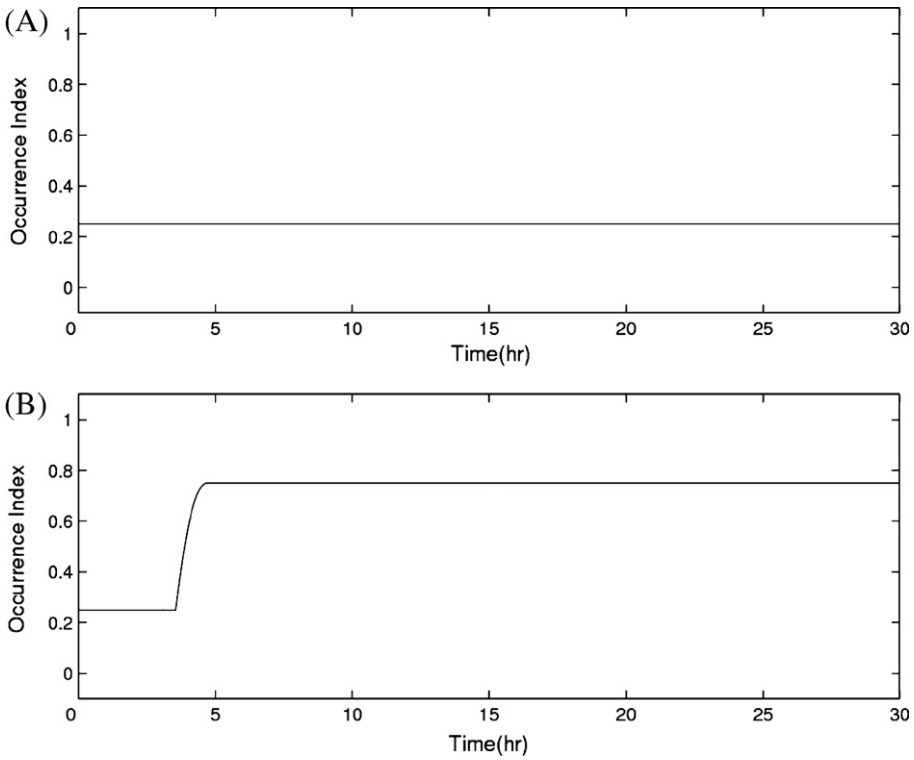

Fig. 15. The diagnosis results obtained with a two-layer inference system concerning the assumed fault origin $T_{c0}(+1)$. (A) Actual fault origin used in numerical simulation: $C_{A0}(+1)$; (B) actual fault origin used in numerical simulation: $T_{c0}(+1)$.

therefore, it is not necessary to distinguish a measured variable from its measurement signal in the digraph model. The resulting SDG is given in Fig. 12. The fault origins considered in this example are: (1) a moderate disturbance in the input concentration, i.e., $C_{A0}(+1)$ and (2) a small increase in the upstream temperature of cooling water, i.e., $T_{c0}(+1)$. The corresponding diagnoser can be found in Fig. 13. Notice that, it is obviously not feasible to uniquely identify a fault origin based only on the eventual symptoms, since the final values of the on-line measurements in the above two scenarios are essentially the same, i.e., $C_A(-1, 0, +1)$, $T(0)$ and $T_c(0)$. On the other hand, one can apply the two-layer diagnostic framework based upon the marked sublanguages generated from Fig. 13, i.e.,

$$\mathfrak{L}_m(A_{diag}^{C_{A0}(+1)}) = \left\{ \begin{array}{l} C_A(+1)T(+1)T_c(-1)T_c(-1/0/+1)[C_A(-1/0/+1), T(0), T_c(0)], \\ C_A(+1)T(+1)T_c(+1)T_c(-1/0/+1)[C_A(-1/0/+1), T(0), T_c(0)] \end{array} \right\}$$

$$\mathfrak{L}_m(A_{diag}^{T_{c0}(+1)}) = \{T_c(+1)T(+1)C_A(-1)[C_A(-1/0/+1), T(0), T_c(0)]\}$$

Numerical simulation studies have also been carried out to confirm the usefulness of these diagnostic languages. The needed mathematical model and its parameters can be found in Appendix B. These two fault propagation scenarios were simulated by setting the value of C_{A0} to be 0.97 lb-mol/ft^3 at 3 h and T_{c0} to be 540°R at 3 h respectively in two separate runs. Satisfactory diagnosis results can be obtained on the basis of these simulated measurement data and these results are presented in Figs. 14 and 15.

6. Conclusions

In this study, a SDG-based reasoning procedure is proposed to qualitatively predict all possible symptom patterns and also their progression sequences. These intrinsic features of the symptom evolution behaviors are captured with automata and language models. The resulting IF-THEN rules can be incorporated in a two-layer fuzzy inference system and this system can be installed on-line to identify not only the locations of fault origins but also their magnitude levels with relatively high resolution. The feasibility and effectiveness of the proposed diagnostic strategy are demonstrated in simulation studies.

Appendix A. Two-layer inference framework

First of all, it should be understood that not all the strings in sublanguage $\mathfrak{L}(A_{diag}^{F_i})$ show up in the actual fault propagation scenario caused by F_i. To illustrate this point, let us use the diagnoser in Fig. 5 as an example. Since there is only one fault origin in this case, the language in Eq. (8) should be the same as the *marked* sublanguage for fault origin $d(+1)$, i.e., $\mathfrak{L}_m(A_{diag}^{d(+1)})$. Notice that two alternative strings are included in this language. It is obvious that only one of the corresponding event sequences takes place in real time. The task of fault diagnosis can thus be viewed as that of establishing at least a partial match between the historical record of on-line symptoms and one of the two candidate strings. A two-layer inference structure has already been developed for this purpose (Chen & Chang, 2006) and a brief summary is presented below:

The first layer of the inference system is used for comparing the *current* on-line symptoms with every diagnoser state and then generating a measure of agreement between them. Let us again consider Fig. 5. The inference rules given in Table A.1 can be used to compute the *agreement measures* of all five diagnoser states. Notice that the premises of these rules are constructed on the basis of the qualitative deviation values of the observed variables at all possible states. These deviations are translated into linguistic values according to an interpretation function Φ_{in}, i.e.,

$$\Phi_{in}(\delta_k) = \left\{ \begin{array}{ll} \text{LP} & \text{if } \delta_k = +10 \\ \text{SP} & \text{if } \delta_k = +1 \\ \text{ZE} & \text{if } \delta_k = 0 \\ \text{SN} & \text{if } \delta_k = -1 \\ \text{LN} & \text{if } \delta_k = -10 \end{array} \right. \quad \text{(A1)}$$

Table A.1
Inference rules for computing agreement measures.

Rule no.	IF			THEN
i	x	z	u	st_j
1	ZE	ZE	ZE	$A_1^{(j)}$
2	SP	ZE	ZE	$A_2^{(j)}$
3	SP	SN	ZE	$A_3^{(j)}$
4	SP	ZE	SP	$A_4^{(j)}$
5	SP	SN	SP	$A_5^{(j)}$

Table A.2
Inference rules for computing closeness measures.

Rule no.	IF				THEN
	st_1	st_2	st_3	st_5	sq_1
1	OCR	NOC	NOC	NOC	NOC
2	OCR	OCR	NOC	NOC	UCT_1
3	OCR	OCR	OCR	NOC	UCT_2
4	OCR	NOC	OCR	NOC	UCT_1
5	OCR	OCR	OCR	OCR	OCR
6	OCR	NOC	OCR	OCR	UCT_2
7	OCR	OCR	NOC	OCR	UCT_2
8	OCR	NOC	NOC	OCR	UCT_1

In the above equation, δ_k denotes the deviation value of measurement k; LN, SN, ZE, SP and LP denote the linguistic values of −10, −1, 0, +1 and +10 respectively. If the observed system state is believed to be identical to a diagnoser state (say state j), then the conclusion "st$_j$ is OCR" should be assigned to the inference rule. Here, OCR is the linguistic value of the agreement measure of state j, i.e., st_j, reflecting the highest confidence level in confirming the match. Otherwise, the conclusion in the first-layer inference rule should be "st_j is NOC", where NOC is the linguistic value representing the lowest confidence. Thus, the linguistic values of the agreement measures listed in Table A.1 can be determined with the following equation:

$$A_i^{(j)} = \begin{cases} \text{OCR} & \text{if } i = j \\ \text{NOC} & \text{if } i \neq j \end{cases} \tag{A2}$$

Notice that a total of 25 rules can be generated accordingly. The maximum values of these agreement measures (i.e., $st_1^m - st_5^m$) and their respective occurrence times (i.e., $\theta_1^m - \theta_5^m$) are updated every time a new batch of measurements is taken. Clearly the occurrence times of the maximum agreement measures must be consistent with the precedence order given in one of the strings. For example, if the first string in Eq. (8) is the event sequence resulting from $d(+1)$, then the following inequality constraint must be imposed

$$\theta_1^m \leq \theta_2^m \leq \theta_3^m \leq \theta_5^m \tag{A3}$$

If, at any time, the inequality constraints associated with *both* strings in $\mathfrak{L}_m(A_{\text{diag}}^{d(+1)})$ are violated, the outputs of the first-layer inference system for fault origin $d(+1)$, i.e., $st_1^m - st_5^m$, should all be reset to zero.

The second-layer inference system is used for qualitatively comparing the time profiles of on-line measurements with each string in $\mathfrak{L}(A_{\text{diag}}^{F_i})$, and then generating a measure of closeness between the two. The inference rules for computing this measure can be constructed on the basis of the outputs obtained from the first layer. For example, the *closeness measure* sq_1 of the first string in Eq. (8) can be computed with the IF-THEN rules given in Table A.2. Notice that rules 1, 2, 3 and 5 in this table are adopted on the basis of Eqs. (15)–(17), where the confidence level ℓ of linguistic value UCT_ℓ is proportional to the string length. The remaining rules in Table A.2 are used to account for the possibilities that one or more intermediate state is not identifiable. Notice that the corresponding rules for the second string in Eq. (8) can be synthesized with the same approach. A complete listing is omitted here for the sake of conciseness.

Finally, it should be noted that the occurrence index of a fault origin should be determined by taking the largest value among all closeness measures. In the present case, this value is

$$o_{d(+1)} = \max\{sq_1, sq_2\} \tag{A4}$$

Appendix B. Mathematical model used for simulating the CSTR reactor system

$$\frac{dV}{dt} = F_0 - F \tag{B1}$$

$$V = A_r h \tag{B2}$$

$$r_A = k_0 e^{-(E/RT)} C_A \tag{B3}$$

$$\frac{dC_A}{dt} = \frac{F_0}{V}(C_{A0} - C_A) - r_A \tag{B4}$$

$$\frac{dT}{dt} = \frac{F_0}{V}(T_0 - T) + \frac{r_A(-\Delta H)}{\rho C_p} - \frac{UA(T - T_c)}{V\rho C_p} \tag{B5}$$

$$\frac{dT_c}{dt} = \frac{F_c}{V_j}(T_{c0} - T_c) + \frac{UA(T - T_c)}{V_j \rho_j C_j} \tag{B6}$$

$$F_c = F_{Cs} - K_c^T\left[(T_{\text{set}} - T) + \frac{1}{\tau_I^T}\int_0^t (T_{\text{set}} - T)dt\right] \tag{B7}$$

$$F = F_s - K_x^H\left[(h_{\text{set}} - h) + \frac{1}{\tau_I^H}\int_0^t (h_{\text{set}} - h)dt\right] \tag{B8}$$

Note that the definitions of model parameters and their steady-state values are given in Table B.1.

Table B.1
Model parameters used in Example 3.

Parameter	Definition	Steady-state value
h	Height of liquid level in reactor	48 ft
V	Reactor volume	4800 ft^3
C_{A0}	Reactant concentration in feed	0.47 lb-mol/ft^3
T	Reactor temperature	537°R
F_0	Feed flow rate	4000 ft^3/h
T_0	Feed temperature	530°R
C_A	Reactant concentration in reactor	0.474 lb-mol/ft^3
T_c	Outlet coolant temperature	537°R
F_c	Coolant flow rate	4836 ft^3/h
V_j	Volume of jacket	385 ft^3
k_0	Frequency factor	7.08×10^{10}/h
E	Activation energy	30,000 Btu/lb-mol
R	Universal gas constant	1.99 Btu/lb-mol °R
U	Overall heat transfer coefficient	150 Btu/h ft^2 °R
A	Heat transfer area	25,000 ft^2
T_{c0}	Inlet coolant temperature	530°R
ΔH	Heat of reaction	−30,000 Btu/lb-mol
C_p	Heat capacity (process side)	0.72 Btu/lbm °R
C_j	Heat capacity (coolant side)	1 Btu/lbm °R
ρ	Density of process mixture	50 lbm/ft^3
ρ_j	Density of coolant	62.3 lbm/ft^3
A_r	Cross-section area of reactor	100 ft^2
K_c^H	Proportional gain of level controller	10
K_c^T	Proportional gain of temperature controller	80
τ_I^H	Integral time of level controller	89.286 h
τ_I^T	Integral time of temperature controller	0.6557 h
h_{set}	Set point of the level height in tank	48 ft
T_{set}	Set point of the temperature in tank	537 °R

References

Cassandras, C. G., & Lafortune, S. (1999). *Introduction to discrete event systems*. Boston: Kluwer Academic Publisher.

Chang, C. T., & Hwang, H. C. (1992). New developments of the digraph-based techniques for fault-tree synthesis. *Industrial and Engineering Chemistry Research*, *31*, 1490–1502.

Chang, S. Y., & Chang, C. T. (2003). A fuzzy-logic based fault diagnosis strategy for process control loops. *Chemical Engineering Science*, *58*, 3395–3411.

Chang, S. Y., Lin, C. R., & Chang, C. T. (2002). A fuzzy diagnosis approach using dynamic fault trees. *Chemical Engineering Science*, *57*, 2971–2985.

Chen, J. Y., & Chang, C. T. (2006). Fuzzy diagnosis method for control systems with coupled feed forward and feedback loops. *Chemical Engineering Science*, *61*, 3105–3128.

Chen, J. Y., & Chang, C. T. (2007). Systematic enumeration of fuzzy diagnosis rules for identifying multiple faults in chemical processes. *Industrial and Engineering Chemistry Research*, *46*, 3635–3655.

Chen, J. Y., & Chang, C. T. (2008). Development of an optimal sensor placement procedure based on fault evolution sequences. *Industrial and Engineering Chemistry Research*, *47*, 7335–7346.

Ju, S. N., Chen, C. L., & Chang, C. T. (2004). Constructing fault trees for advanced process control systems – Application to cascade control loops. *IEEE Transactions on Reliability*, *53*, 43–60.

Lapp, S. A., & Powers, G. J. (1977). Computer-aided synthesis of fault-trees. *IEEE Transactions on Reliability*, *26*, 2–13.

Maurya, M. R., Rengaswamy, R., & Venkatasubramanian, V. (2003a). A systematic framework for the development and analysis of signed digraphs for chemical processes. 1. Algorithms and analysis. *Industrial & Engineering Chemistry Research*, *42*(20), 4789–4810.

Maurya, M. R., Rengaswamy, R., & Venkatasubramanian, V. (2003b). A systematic framework for the development and analysis of signed digraphs for chemical processes. 2. Control loops and flowsheet analysis. *Industrial & Engineering Chemistry Research*, *42*(20), 4811–4827.

Maurya, M. R., Rengaswamy, R., & Venkatasubramanian, V. (2004). Application of signed digraph-based analysis for fault diagnosis chemical process flowsheets. *Engineering Applications of Artificial Intelligence*, *17*, 501–518.

Maurya, M. R., Rengaswamy, R., & Venkatasubramanian, V. (2006). A signed directed graph-based systematic framework for steady state malfunction diagnosis inside control loops. *Chemical Engineering Science*, *61*, 1790–1810.

Maurya, M. R., Rengaswamy, R., & Venkatasubramanian, V. (2007). A signed directed graph and qualitative trend analysis-based framework for incipient fault diagnosis. *Chemical Engineering Research & Design*, *85*(A10), 1407–1422.

Oyeleye, O. O., & Kramer, M. A. (1988). Qualitative simulation of chemical process systems: Steady-state analysis. *AIChE Journal*, *34*, 1441–1454.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., & Teneketzis, D. (1996). Failure Diagnosis using discrete-event models. *IEEE Transactions on Control Systems Technology*, *4*, 105–124.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., & Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, *40*, 1555–1575.

Venkatasubramanian, V., Rengaswamy, R., & Kavuri, S. (2003). A review of process fault detection and diagnosis. Part 2. Qualitative model and search strategies. *Computers and Chemical Engineering*, *27*, 313–326.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., & Kavuri, S. (2003). A review of process fault detection and diagnosis. Part 1. Quantitative model based methods. *Computers and Chemical Engineering*, *27*, 293–311.