Front. Electr. Electron. Eng. China 2008, 3(2): 155–161
DOI 10.1007/s11460-008-0042-2

RESEARCH ARTICLE

Zhijun DING, Meiqin PAN, Changjun JIANG, Yaojun HAN

Using Petri nets to verify acyclic rule-based system

Abstract A series of Petri net-based definitions were formulated for describing four types of structural errors in a rule-based system (RBS), including inconsistency, incompleteness, redundancy and circularity. A marked ω-Petri net model of acyclic RBS was constructed. Then, its reachability tree was generated to record all reachable relations between propositions in RBS. Moreover, a backward reasoning forest of a reachable marking was generated for explicitly representing reachable paths in RBS. Finally, a set of theorems and algorithms were provided to analyze and check structural errors. The usability of the research results presented in this paper was illustrated by an example.

Keywords rule-based system (RBS), horn clause, Petri net, verification

1 Introduction

Rule-based system (RBS) has been widely applied to all kinds of expert systems. It can effectively express domain knowledge and inference rules, and solve difficult domain problems using logical reasoning. In general, the procedure for domain experts to accumulate knowledge and expertise is incremental and intuitive, thus, RBS needs to be constructed incrementally and refined multiple times. On one hand, an expert system is usually constructed by consulting with numerous experts, and these experts may have conflicting expertise. On the other hand, for development of domain knowledge, an expert system is adjusted periodically in practice. The adjustments include deleting, adding or modifying rules in RBS, which inevitably leads to inconsistent specifications. As a result, it is very important that such a system should be verified for correctness before it is used in any environment. The verification requires checking whether the rules in the system are consistent and complete. Also, it requires detecting and finding superfluous, incorrect or missing rules that have poor effect on the performance of the expert system.

Different techniques have been proposed to perform verification checks for RBS. Nazareth and Kennedy provided a directed graph approach to represent an RBS for the verification problem [1]. Ramaswamy et al. used directed hypergraphs to model simple and compound clauses in an RBS, and then presented the corresponding verification method to detect errors [2]. Petri nets can essentially be viewed as graphical models and mathematical tools, which can provide structural information and analyze dynamical properties of systems such as manufacturing systems, workflow, etc. Thus, Petri nets and their modified models have also been used to study RBS [3]. Murata et al. used Petri nets to model Horn clauses, and then studied consistency problems of rules [4]. Yang et al. proposed a Petri net formalism for the verification of RBS based on incidence matrix operations [5]. Moreover, the reachability of Petri nets can express logical relations between conditions and conclusions of a rule, thus some works focus on how to use reachability analysis method to verify the correctness of an RBS. Nazareth investigated the applicability of the reachability analysis method for RBS verification, but did not provide an effective way to realize error checking [6]. He et al. presented a special reachability graph technique based on a special class of low-level Petri nets, called ω-nets, to detect errors in RBS. However, this method requires appointing a set of facts and goals that may not guarantee to detect all potential errors involved in an RBS [7].

Grounded on the above researches, this paper presents an approach to verify acyclic RBS using Petri nets. Section 2 constructs a Petri net model for RBS, and presents rigorous definitions for four types of structural errors of RBS. A special class of Petri nets, called marked ω-nets, is proposed for modeling acyclic RBS. The reachability tree generation algorithm is provided for marked ω-nets in Sect. 3. In Sect. 4, a set of theorems and algorithms are provided to analyze and check structural errors. Section 5 presents the concluding remarks.

Translated from *Journal of Tongji University* (*Natural Science*), 2007, 35(2): 232–238 [译自: 同济大学学报 (自然科学版)]

Zhijun DING (✉), Meiqin PAN
College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266510, China
E-mail: zhijun_ding@hotmail.com

Changjun JIANG, Yaojun HAN
Department of Computer Science and Engineering, Tongji University, Shanghai 201804, China

2 Petri nets for RBS modeling

2.1 Structural errors in RBS

Horn clause is one of the important logic systems. It has been proven that most of the problems expressed in logic can also be represented by means of the Horn clauses [8]. A Horn clause is usually written as $P_1 \wedge P_2 \wedge \cdots \wedge P_n \rightarrow Q$, where P_i and Q are atomic propositions (literals). Moreover, the left hand side is called the conditions and the right hand side is called the conclusion. This notation means that if all conditions P_1, P_2, …, P_n hold, the conclusion Q will be implied. In this paper, it is supposed that every RBS is composed of some rules expressed by Horn clauses.

As pointed out by Nazareth [1,6], typical structural errors in RBS include redundancy (redundant rules), inconsistency (conflict rules), incompleteness (missing rules), and circularity (circular depending rules).

1) Redundancy: an RBS demonstrates redundancy when two or more distinct sets of rules lead to the same conclusion from a given set of conditions or facts. Redundant rules not only increase the size of the RBS, but also may cause additional useless inferences.

2) Inconsistency: it is also called contradiction or conflict. Rules are inconsistent when the same conditions lead to mutually exclusive conclusions. Clearly, inconsistent rules lead to incorrect decisions, and must be eliminated before the expert system is ready for implementation.

3) Incompleteness: incompleteness is relative to the given facts and goals of an RBS, and includes dead ends and unreachable goals. In detail, on one hand, if the conclusion of a rule is neither part of goals, nor part of the conditions of some other rules, then that rule constitutes a dead end. This indicates either that the rule should not be included in the RBS, or some other rules are missing which would enable the conclusion of this rule to be used to deduce the final goal. On the other hand, if the condition of a rule is neither part of given facts nor part of the conclusion of some other rules, the rule constitutes an unreachable goal. This implies that either the rule is irrelevant, or some other rules that can establish the condition of the rule are missing.

4) Circularity: circularity occurs when several rules have circular dependency. Circularity causes infinite reasoning and must be broken.

2.2 Petri net model of RBS

$N = (P,T,F,W)$ is a Petri net, where P is a finite set of places, T is a finite set of transitions such that $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation and W is a weight function such that $W(x,y) \in \mathbb{N}^+$ (the set of positive integers) if $(x,y) \in F$, and $W(x,y) = 0$ if $(x,y) \notin F$. The pre- and post-sets of a node $x \in P \cup T$ are defined respectively as ${}^{\cdot}x = \{y \in P \cup T | (y,x) \in F\}$ and $x^{\cdot} = \{y \in P \cup T | (x,y) \in F\}$. The marking (or state) of a net is a function $M : P \rightarrow \mathbb{N}$ (the set of non-negative integers) [9,10].

The modeling process of an RBS by a Petri net involves the representation of each rule as a transition, the conditions of the rule as input places, and the conclusion as an output place. For example, a rule R: $A \wedge B \rightarrow C$, which is obviously a Horn clause, can be modeled as a Petri net in Fig. 1. As stated above, RBS is composed of several such rules. Thus, we can use Petri nets to model each rule, and then obtain a Petri net model for the whole RBS.

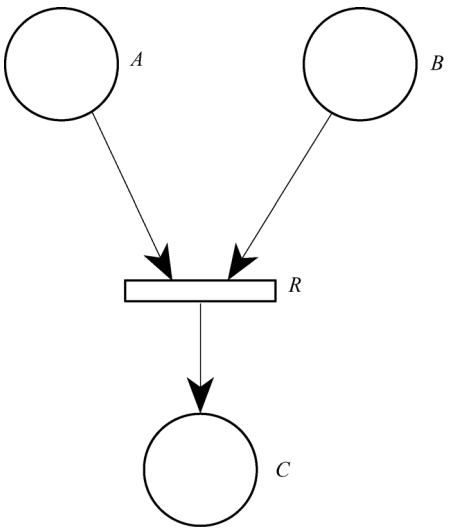

Fig. 1 Petri net model of Horn clause

Moreover, for the convenience of constructing the Petri net model and decreasing the complexity of analysis for an RBS, we need to normalize the rules before they are modeled by Petri nets. If both proposition P and its negation $\neg P$ are involved in an RBS $\Theta = \{R_1, R_2, \ldots, R_m\}$, then we replace them with symbols P_1 and P_2 respectively.

Here, we illustrate the Petri net representation using the following example of an RBS which is a modification of a project termination model for R&D project [2].

Rule 1 IF the likelihood of technical success is high (A). Then the project leader gains higher autonomy in carrying out the work (C).

Rule 2 IF the likelihood of technical success is high (A). Then continue funding (J).

Rule 3 IF the likelihood of technical success is high (A). And the likelihood of cost over-runs is high (B). Then anticipated return on investment is low (E).

Rule 4 IF the likelihood of cost over-runs is high (B). Then priority classification is C (G).

Rule 5 IF the likelihood of cost over-runs is high (B). Then funding from industry is not available (F).

Rule 6 IF anticipated return on investment is low (E). Then the priority classification is C (G).

Rule 7 IF funding from industry is not available (F). Then the project cannot be completed within three years (D).

Rule 8 IF funding from industry is not available (F). And priority classification is C (G). Then discontinue funding ($\neg J$).

Rule 9 IF profitability is low (I). Then discontinue funding ($\neg J$).

We can formalize and normalize the above RBS, and then obtain a set of Horn clauses.

$$R_1 : A \to C;\ R_2 : A \to J_1;\ R_3 : A \wedge B \to E;\ R_4 : B \to G;$$

$$R_5 : B \to F;\ R_6 : E \to G;\ R_7 : F \to D;\ R_8 : G \wedge F \to J_2;$$

$$R_9 : I \to J_2.$$

Here J and $\neg J$ are replaced with J_1 and J_2 respectively. A Petri net model $N^R = (P^R, T^R, F^R, W^R)$ for the RBS is shown in Fig. 2.

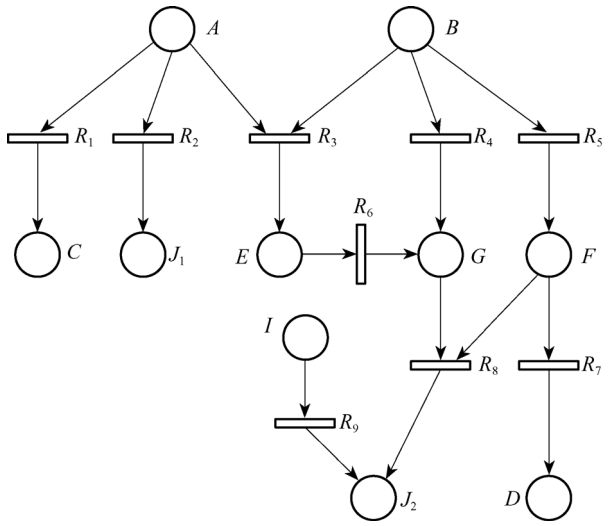

Fig. 2 Petri net model of R&D project RBS

2.3 Formal definition of structural errors based on Petri net model

Based on Petri net model of RBS, structural errors of RBS can be defined formally and rigorously expressed.

Definition 1 Let Petri net $N = (P,T,F,W)$ be a model of an RBS Θ. If there exist two distinct paths $\alpha_1 = (t_1,p_1, t_2,\ldots,t_p)$ and $\alpha_2 = (t_{p+1},p_{p+1},t_{p+2},\ldots,t_{p+q})$ satisfying the following three conditions, then the RBS Θ exists redundancy. Correspondently, α_1 and α_2 are in this sense redundant.

1) $t_i^{\cdot} = {}^{\cdot}t_{i+1}$, $p_i \neq p_j$, where $1 \leqslant i,\ j < p,\ p+1 \leqslant i$, and $j < p+q$.

2) ${}^{\cdot}t_1 \subseteq {}^{\cdot}t_{p+1}$ or ${}^{\cdot}t_1 \supseteq {}^{\cdot}t_{p+1}$.

3) $t_p^{\cdot} = t_{p+q}^{\cdot}$

Definition 2 Let Petri net $N = (P,T,F,W)$ be a model of an RBS Θ. If there exist two distinct paths $\alpha_1 = (t_1,p_1, t_2,\ldots,t_p)$ and $\alpha_2 = (t_{p+1},p_{p+1},t_{p+2},\ldots,t_{p+q})$ satisfying the following three conditions, then the RBS Θ is inconsistent. Correspondently, α_1 and α_2 are called inconsistent.

1) $t_i^{\cdot} = {}^{\cdot}t_{i+1}$, $p_i \neq p_j$, where $1 \leqslant i,\ j < p,\ p+1 \leqslant i$, and $j < p+q$.

2) ${}^{\cdot}t_1 \subseteq {}^{\cdot}t_{p+1}$ or ${}^{\cdot}t_1 \supseteq {}^{\cdot}t_{p+1}$.

3) p_{p+1} and p_{p+q+1} are conflicting, where $t_p^{\cdot} = \{p_{p+1}\}$ and $t_{p+q}^{\cdot} = \{p_{p+q+1}\}$.

Definition 3 Let Petri net $N = (P,T,F,W)$ be a model of an RBS Θ. F_s and G_s are given fact set and goal set respectively. $p \in P$ is a dead end if and only if $p \notin G_s$ and $p^{\cdot} = \varnothing$.

Definition 4 Let Petri net $N = (P,T,F,W)$ be a model of an RBS Θ. F_s and G_s are given fact set and goal set respectively. $p \in P$ is an unreachable goal if and only if $p \notin F_s$ and ${}^{\cdot}p = \varnothing$.

Definition 5 Let Petri net $N = (P,T,F,W)$ be a model of an RBS Θ, circularity occurs in Θ if and only if there exists at least one circuit $(p_1,t_1,p_2,\ldots,t_p,p_1)$ in N.

Moreover, we can define a subclass of RBS, called acyclic rule-based system as follows.

Definition 6 Let Petri net $N = (P,T,F,W)$ be a model of an RBS Θ. If there does not exist any circuits in N, then we call Θ an acyclic rule-based system.

In the following sections, we will propose a reachability based method for analyzing and verifying an acyclic RBS.

3 Marked ω-Petri net and its reachability tree

3.1 Marked ω-Petri net

The results in Ref. [6] indicate that reachability of Petri nets can be used to analyze and detect structural errors of an RBS. However, the model proposed in that paper is so complicated that it cannot be utilized in practice. The author presented an error detecting technique based on ω-net model, which used symbol ω to express the number of tokens in places [7]. After introducing the symbol ω, this method can deduce the scale of the model and the reachable states, but at the same time it inevitably causes missing of some important information such as reachable path, so its effectiveness cannot be guaranteed. Therefore, in this paper, based on the ω-net model, we define a marked ω-Petri net model in which reachable paths can be precisely recorded. Consequently, it is helpful to accurately detect structural errors of RBS.

Suppose that $\Omega = \{\omega_1,\omega_2,\ldots,\omega_k\}$ is a non-empty finite set. ω_i is an infinite symbol such that

1) $\omega_i \pm n = \omega_i$, $\omega_i > n$ for any integer n.

2) $\omega_i \geqslant \omega_i$. Here, ω_i is called a marked ω-number. An expression is called AND expression on Ω if it is composed of a finite number of elements of Ω concatenated with

symbol $\otimes$. While an expression is called OR expression on Ω if it is composed of a finite number of elements of Ω concatenated with symbol $\oplus$. Moreover, $\langle\Omega,\oplus,\otimes\rangle$ is a set of expressions, and every expression involves elements of Ω applying symbols $\otimes$ and $\oplus$ finitely.

Definition 7 $\Sigma=(N,M)$ is a marked ω-Petri net, where

1) $N=(P,T,F,W)$ is a Petri net.

2) $M{:}P\to\{0,\langle\Omega,\otimes,\oplus\rangle\}$ is a marking.

Moreover, a marking in Σ is changed according to the following firing rules:

1) A transition $t\in T$ is enabled at marking M if and only if $\forall p\in {}^{\cdot}t$, $M(p)\geqslant W(t,p)$. It is denoted as $M[t>$.

2) Firing an enabled transition t results in changing M into M' represented by $M[t>M'$, where

a) If $p\in t^{\cdot}$ and $M(p)=0$, then $M'(p)=\omega_p=(\omega_{p_{i1}}\otimes\omega_{p_{i2}}\otimes\cdots\otimes\omega_{p_{ik}})_p$, where ${}^{\cdot}t=\{p_{i1},p_{i2},\ldots,p_{ik}\}$.

b) If $p\in t^{\cdot}$ and $M(p)\neq 0$, then $M'(p)=\omega_p=(M(p)\oplus(\omega_{p_{i1}}\otimes^{\bullet\bullet\bullet}\otimes\omega_{p_{ik}}))_p$.

c) Otherwise, $M'(p)=M(p)$.

3.2 Marked ω-Petri net model of an acyclic RBS

It is clear that a Petri net model of an acyclic RBS is a directed acyclic graph. For a directed acyclic graph, all paths between two nodes can be found by topological sort algorithm, i.e., they begin with nodes whose in-degrees are zero, and then traverse each node according to topological order [11]. Based on the above method, we first define a marked ω-Petri net for modeling an acyclic RBS, and then provide a reachability tree generation algorithm for marked ω-Petri nets.

Definition 8 Let $N=(P,T,F,W)$ be a net model of an acyclic RBS Θ. $P_{\mathrm{NI}}\subset P$ is a set of places that have no input arcs, i.e., $\forall p\in P_{\mathrm{NI}}$, ${}^{\cdot}p=\emptyset$. $\Sigma=(N,M_0)$ is called a marked ω-Petri net model of Θ, where

$$M_0(p)=\begin{cases}\omega_p, & p\in P_{\mathrm{NI}}\\ 0, & p\notin P_{\mathrm{NI}}\end{cases}.$$

According to Definition 8, we can obtain a marked ω-Petri net model $\Sigma^R=(N^R,M_0^R)$ for R&D project RBS, where $M_0^R(A)=\omega_A$, $M_0^R(B)=\omega_B$, and $M_0^R(p)=0$ for $p\in P^R-\{A,B\}$.

Moreover, a reachability tree of a marked ω-Petri net Σ is constructed by the following algorithm.

Algorithm 1 A reachability tree generation algorithm of a marked ω-Petri net.

Input: $\Sigma=(N,M_0)$

Output: reachability tree $\mathrm{RT}(\Sigma)$

$T'\leftarrow T$;

Let initial marking M_0 be a root node of $\mathrm{RT}(\Sigma)$;

$M\leftarrow M_0$;

WHILE $T'\neq\emptyset$ DO {

Solve all enabled transitions under the current marking M, record them into a set $\bar{T}$;

$M'\leftarrow M$;

FOR each t in $\bar{T}$ DO {

Obtain the marking M'' that results from firing t at M';

$M'\leftarrow M''$}

Introduce M' as a node, draw an arc with label $\bar{T}$ from M to M';

$T'\leftarrow T'-\bar{T}$; $M\leftarrow M'$}

It is proved that this algorithm always terminates [7]. He et al. indicated that the worst-case computational complexity of the reachability algorithm of a ω-net is $O(|P|\times|T|^2/2)$ [7]. A marked ω-Petri net is proposed by introducing markings into a ω-net to record path information. Thus, the computational complexity of this algorithm is the same with that of the algorithm in Ref. [7]. That is, the worst-case computational complexity of the reachability tree generation algorithm of a marked ω-Petri net is $O(|P|\times|T|^2/2)$.

Using Algorithm 1, we can obtain a reachability tree $\mathrm{RT}(\Sigma^R)$ of Σ^R as shown in Fig. 3.

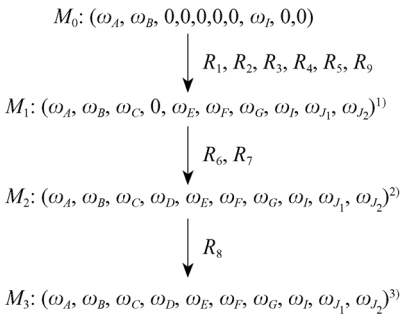

Fig. 3 Reachability tree of Σ^R

1) $\omega_C=(\omega_A)_C$, $\omega_E=(\omega_A\otimes\omega_B)_E$, $\omega_F=(\omega_B)_F$, $\omega_G=(\omega_B)_G$, $\omega_{J_1}=(\omega_A)_{J_1}$, $\omega_{J_2}=(\omega_I)_{J_2}$.

2) $\omega_C=(\omega_A)_C$, $\omega_D=\big((\omega_B)_F\big)_D$, $\omega_E=(\omega_A\otimes\omega_B)_E$, $\omega_F=(\omega_B)_F$, $\omega_G=\big(\omega_B\oplus(\omega_A\otimes\omega_B)_E\big)_G$, $\omega_{J_1}=(\omega_A)_{J_1}$, $\omega_{J_2}=(\omega_I)_{J_2}$.

3) $\omega_C=(\omega_A)_C$, $\omega_D=\big((\omega_B)_F\big)_D$, $\omega_E=(\omega_A\otimes\omega_B)_E$, $\omega_F=(\omega_B)_F$, $\omega_G=\big(\omega_B\oplus(\omega_A\otimes\omega_B)_E\big)_G$, $\omega_{J_1}=(\omega_A)_{J_1}$, $\omega_{J_2}=\Big(\omega_I\oplus\Big((\omega_B)_F\otimes\big(\omega_B\otimes(\omega_A\otimes\omega_B)_E\big)_G\Big)\Big)_{J_2}$.

Obviously, in this algorithm path information which is necessary for error detecting can be recorded in reachable markings. For example, in $\mathrm{RT}(\Sigma^R)$, we have $M_3(D)=\omega_D=\big((\omega_B)_F\big)_D$, which means that there exists a path $B\to F\to D$ from the fact B to the goal D.

3.3 Backward reasoning forest of reachable marking

As stated above, a reachable marking of a marked ω-Petri net records implicitly some path information. For representing this information explicitly, we first define

backward reasoning tree of a place in a marking based on the following Algorithm 2, and then define a backward reasoning forest of a reachable marking.

Definition 9 Let $\Sigma = (N, M_0)$ be a marked ω-Petri net, where $N = (P, T, F, W)$ is a net, and M_0 is an initial marking. For a reachable marking $M \in R(\Sigma)$, if $M(p) > 0$, where $p \in P$, a tree for $M(p)$ can be constructed by Algorithm 2. This tree is called a backward reasoning tree of p in M, denoted as BRT(M,p).

Algorithm 2 A backward reasoning tree generation algorithm.

Input: $M(p)$

Output: backward reasoning tree BRT(M,p)

A suffix outside outmost bracket is a root node v_0 of BRT(M,p);

IF an expression within bracket is a OR expression THEN{

FOR each subentry of the expression DO{

IF this subentry is an AND expression THEN{

Construct a tree for each subentry of this AND expression, and draw an arc from v_0 to this tree;

Draw a curve among these arcs representing "AND" relation}

ELSE

Construct a tree for this subentry, and draw an arc from v_0 to this tree}}

IF an expression within bracket is an AND expression THEN{

Construct a tree for each subentry, and draw an arc from v_0 to this tree;

Draw a curve among these arcs representing "AND" relation}

IF an expression within bracket has form like $(\ )_s$ THEN

Construct a tree for this expression, and draw an arc from v_0 to this tree

IF an expression within bracket has form like ω_{p_i} THEN

p_i is a leaf node of BRT(M,p), and draw an arc from v_0 to p_i.

The above algorithm can be used recursively to generate a tree for $M(p)$. A backward reasoning tree BRT(M,p) is composed of all directed paths whose terminal is place p, i.e., root node of BRT(M,p).

Let n be the number of suffix in the expression of $M(p)$. It is clear that BRT(M,p) has n nodes. The computational complexity of procedure that constructs a tree including n nodes is $O(n)$ [11]. Thus computational complexity of Algorithm 2 is $O(n)$.

Definition 10 Let $\Sigma = (N, M_0)$ be a marked as ω-Petri net, where $N = (P, T, F, W)$ is a net, and M_0 is an initial marking. For a reachable marking $M \in R(\Sigma)$, a forest of M is composed of all backward reasoning trees BRT(M,p) where $M(p) > 0$. Moreover, this forest is called a backward reasoning forest of M, denoted as BRF(M).

Using Algorithm 2, we can obtain a backward reasoning forest BRF(M_3) as shown in Fig. 4, where $M_3 \in RT(\Sigma^R)$.

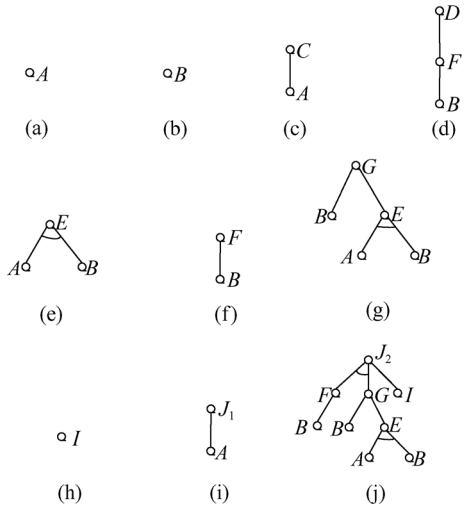

Fig. 4 A backward reasoning forest BRF(M_3). (a) BRT(M_3,A); (b) BRT(M_3,B); (c) BRT(M_3,C); (d) BRT(M_3,D); (e) BRT(M_3,E); (f) BRT(M_3,F); (g) BRT(M_3,G); (h) BRT(M_3,I); (i) BRT(M_3,J_1); (j) BRT(M_3,J_2)

4 Error checking by marked ω-Petri net

For a marked ω-Petri net, given a reachable marking, its backward reasoning forest provides information of all possible reachable paths whose terminal is any place in the net. In this section, we present a set of error checking methods for an RBS based on the reachable path information.

4.1 Redundancy checking

According to Definition 1, we can directly obtain a redundancy checking theorem based on backward reasoning trees. For an interior node v in a backward reasoning tree, we can classify its son nodes according to AND relations. If there exists a AND relation among some son nodes, these nodes can compose a set, otherwise, a single son node composes a set. Each set is called a condition set of v.

Theorem 1 Let $\Sigma = (N, M_0)$ be a marked ω-Petri net modeling an acyclic RBS Θ, and BRT(M,p) be a backward reasoning tree of p in M. If there exist two distinct paths $\alpha_1 = (p, v_1, \ldots, v_p)$ and $\alpha_2 = (p, v_{p+1}, \ldots, v_{p+q})$ in BRT(M,p), where v_p has condition sets $C_1, C_2, \ldots, C_s$ and v_{p+q} has condition sets $C_{s+1}, C_{s+2}, \ldots, C_{s+t}$, then α_1 and α_2 are redundant paths if and only if the following conditions are satisfied:

1) $\exists C_i$, C_j such that $C_i \subseteq C_j$ or $C_i \supseteq C_j$, where $1 \leqslant i \leqslant s$ and $s+1 \leqslant j \leqslant s+t$.

2) $v_x \neq v_y$, where $1 \leqslant x \leqslant p$ and $p+1 \leqslant y \leqslant p+q$.

3) There exists no AND relation between v_k and other nodes, where $1 \leqslant k \leqslant p+q$.

Theorem 1 is a presentation of redundancy based on backward reasoning tree. Thus its correctness can be

guaranteed by the definition of redundancy and backward reasoning tree.

Moreover, we propose an algorithm that is able to do redundancy checking according to Theorem 1. It is concluded that a leaf node of a reachability tree, i.e., a terminal marking M_F, records whole directed paths according to Algorithm 1. Thus, we only need to analyze M_F to find all redundant paths. Furthermore, for a place p, if the expression of $M_F(p)$ does not contain the symbol $\oplus$, there exists no redundant paths. Based on the above analysis, the following algorithm can be presented to detect redundancy.

Algorithm 3 Redundant paths checking algorithm.

Input: M_F, BRF(M_F)

Output: condition sets C_i and C_j of redundant paths.

FOR each place p such that the expression of $M_F(p)$ contains symbol $\oplus$, DO{

WHILE the number of nodes of BRT(M_F,p) is not equal to 1 DO{

Classify leaf nodes to obtain some condition sets $C_1,C_2,\ldots,C_k$;

FOR each couple of sets C_i and C_j such that $C_i \subseteq C_j$ or $C_i \supseteq C_j$ DO{

Traverse path from the root node to the father node of C_i and C_j respectively, record all nodes except the root node on the path in sets of H_i and H_j respectively;

IF $H_i \cap H_j = \emptyset$ and for any $v_k \in H_i \cup H_j$, there exists no AND relation between v_k and other nodes THEN

There exists redundancy, and output sets C_i and C_j}

Let set V be composed of all interior nodes of BRT(M_F,p);

FOR each set C_x where $1 \leqslant x \leqslant k$ DO{

IF $C_x \cap V = \emptyset$ THEN

Delete all elements of C_x and correspondent arcs, and obtain a new tree BRT(M_F,p)}}}

All leaf nodes of BRT(M_F,p) can at most compose $|T|$ different sets. We need a comparison to determine whether there exists a contain relation between the two sets. Thus, $(|T| \times |T-1|)/2$ comparisons are needed for $|T|$ sets. Moreover, the length of a path that needs to be traversed is at most $|P-2|$. Thus the computational complexity of the procedure that determines whether two paths are redundant is $(|T| \times |T-1|)/2 + (|T| \times |T-1|)/2 \times 2|P-2| = O(|T|^2 \times |P|)$.

According to Algorithm 3, we need to analyze trees BRT(M_3,G) and BRT(M_3,J_2) as shown in Figs. 4(g) and 4(j) respectively. The results indicate that there exist redundant paths in BRT(M_3,G) and there exists no redundant paths in BRT(M_3,J_2). In BRT(M_3,G), beginning with condition sets $C_1 = \{A,B\}$ and $C_2 = \{B\}$, there exist two distinct paths to node G. That is, from proposition B, two distinct sets of rules $\{R_4\}$ and $\{R_3,R_6\}$ lead to the same conclusion G.

4.2 Inconsistency checking

Similar to redundancy checking, we first obtain an inconsistency checking theorem based on backward reasoning trees.

Theorem 2 Let $\Sigma = (N,M_0)$ be a marked ω-Petri net modeling an acyclic RBS Θ. Places p and p' denote proposition Y and its negation $\neg Y$ respectively. Suppose that BRT(M,p) and BRT(M,p') are backward reasoning trees of p and p' respectively. If there exist two distinct paths $\alpha_1 = (p,v_1,\ldots,v_p)$ and $\alpha_2 = (p',v_{p+1},\ldots,v_{p+q})$ in BRT(M,p) and BRT(M,p') respectively, where v_p has condition sets $C_1,C_2,\ldots,C_s$ and v_{p+q} has condition sets $C_{s+1},C_{s+2},\ldots,C_{s+t}$, then α_1 and α_2 are inconsistent paths if and only if the following conditions are satisfied:

1) $\exists C_i$, C_j such that $C_i \subseteq C_j$ or $C_i \supseteq C_j$, where $1 \leqslant i \leqslant s$ and $s+1 \leqslant j \leqslant s+t$.

2) $v_x \neq v_y$, where $1 \leqslant x \leqslant p$ and $p+1 \leqslant y \leqslant p+q$;

3) There exists no AND relation between v_k and other nodes, where $1 \leqslant k \leqslant p+q$.

Moreover, we can present an algorithm to check inconsistent paths following Algorithm 3. Thus, the computational complexity of procedure that determines whether two paths are inconsistent is $O(|P| \times |T|^2)$.

We use the algorithm to analyze trees BRT(M_3,J_1) and BRT(M_3,J_2) as shown in Figs. 4(i) and 4(j) respectively. The results show that there exists no inconsistency in RBS Θ.

4.3 Incomplete checking

As stated in Sect. 2, incompleteness is relative to the given facts and goals of an RBS, and includes dead ends and unreachable goals. Let F_s and G_s be a given fact set and a goal set respectively. It is clear that F_s and G_s are subsets of place set P. We can obtain two theorems and corresponding algorithms to detect incompleteness.

Theorem 3 Let $\Sigma = (N,M_0)$ be a marked ω-Petri net modeling an acyclic RBS Θ. F_s and G_s are given fact set and goal set respectively. For any $M \in R(M_0)$ and $p \in G_s$, there exists a place p' such that $p' \notin E_c$, where E_c is a set which is composed of all nodes of BRT(M,p) for any $p \in G_s$, then p' is a dead end.

According to Theorem 3, we can provide the following algorithm for dead ends detection.

Algorithm 4 Dead ends checking algorithm.

Input: BRF(M_F)

Output: a set of dead ends C_d

FOR each place $p \in G_s$ DO

Traverse BRT(M_F,p), and record nodes of the tree in a set E_c;

$C_d \leftarrow P - E_c$.

The number of edges of a backward reasoning tree BRT(M_F,p) is at most $|P| \times |T|$. Thus the computational complexity of procedure that traverses a tree is $O(|P| \times$

$|T|$). Moreover, we need to traverse $|G_s|$ trees. So, the computational complexity of this algorithm is $O(|P| \times |G_s| \times |T|)$.

The following theorem and algorithm is presented for checking unreachable goals.

Theorem 4 Let $\Sigma = (N, M_0)$ be a marked ω-Petri net modeling an acyclic RBS Θ. F_s and G_s are given fact set and goal set respectively. For any $M \in R(M_0)$, there exists a place $p \notin F_s$ such that BRT(M,p) contains only one node, then p is an unreachable goal.

Algorithm 5 Unreachable goal checking algorithm.

Input: BRF(M_F)

Output: a set of unreachable goals C_N

FOR each $p \notin F_s$ DO{

IF the root node of BRT(M_F,p) has not any son nodes, THEN

p is an unreachable goal, and record it into C_N}

It is clear that the computational complexity of this algorithm is $O(|P|)$.

It is assumed that the observable facts are ''likelihood of technical success being high'' and ''likelihood of cost over-runs becoming high'' (i.e., A and B) for R&D project. The goal is to decide whether to continue funding the R&D project or not (i.e., J or $\neg$J). That is, $F_s = \{A, B\}$ and $G_s = \{J_1, J_2\}$. It is found that places C and D are dead ends through analyzing BRF(M_3) by Algorithm 4. Moreover, place I is an unreachable goal according to Algorithm 5.

5 Conclusions

This paper provides an approach to analyze and detect structural errors for an RBS. The approach effectively utilizes the characteristic of acyclic directed graphs, and changes structural problem (i.e., structural errors) of RBS into reachability problem (i.e., reachability trees) of Petri nets. Comparing with known approaches, this approach does not depend on given facts and goals. Thus, it can be used to check all potential errors involved in an RBS. Moreover, our further research will be engaged in structural errors checking for cyclic RBS.

Acknowledgements This work was supported by the National Basic Research Program of China (Nos. 2003CB317002 and 2007CB316502), the National Natural Science Foundation of China (Grant Nos. 60534060 and 90718012).

References

1. Nazareth D L, Kennedy M H. Verification of rule-based knowledge using directed graphs. Knowledge Acquisition, 1991, 3: 339–360
2. Ramaswamy M, Sarkar S, Chen Y S. Using directed hypergraphs to verify rule-based expert systems. IEEE Transactions on Knowledge and Data Engineering, 1997, 9(2): 221–237
3. Looney C G, Liang L R. Inference via fuzzy belief Petri nets. In: Chen I R, Anderson C eds. Proceedings of the Fifteenth IEEE International Conference on Tools with Artificial Intelligence. Sacramento: IEEE Computer Society, 2003: 510–514
4. Murata T, Subrabmanian V S, Wakayama T. A Petri net model for reasoning in the presence of inconsistency. IEEE Transactions on Knowledge and Data Engineering, 1991, 2(3): 281–292
5. Yang S J H, Lee A S, Chu W C, et al. Rule base verification using Petri nets. In: Hughes E eds. Proceedings of the 22nd International Computer Software and Applications Conference. Sacramento: IEEE Computer Society, 1998: 476– 481
6. Nazareth D L. Investigating the applicability of Petri nets for rule-based system verification. IEEE Transactions on Knowledge and Data Engineering, 1993, 5(3): 402–415
7. He X D, Chu W C, Yang H J. A new approach to verify rule-based systems using Petri nets. Information and Software Technology, 2003, 45(10): 663–669
8. Lin C, Chaudhury A, Whinston A, et al. Logical inference of Horn clauses in Petri net models. IEEE Transactions on Knowledge and Data Engineering, 1993, 5(3): 416–425
9. Jiang C J. Behavior Theory and Applications of Petri Net. Beijing: Higher Education Press, 2003, 19–28 (in Chinese)
10. Yuan C Y. The Principle and Application of Petri Nets. Beijing: Publishing House of Electronics Industry, 2005: 58–70 (in Chinese)
11. Yan W M, Wu W M. Data Structure. Beijing: Tsinghua University Press, 2002 (in Chinese)