# A Formal Method for Verifying Production Knowledge Base

Hongtao Huang*, Shaobin Huang*, Tao Zhang*
*College of Computer Science and Technology
Harbin Engineering University, HEU
Harbin, Heilongjiang, 150001, China
hhtdeemail@gmail.com; huangshaobin@hrbeu.edu.cn; zhangtaohrbeu@163.com

***Abstract*—The most important thing of using model checking technology to verify production knowledge base is to build system model from rule set. It is a fundamental but time-consuming job. This paper presents an efficient formal method to verify production knowledge base. Two main contributions of this paper are as follows. Firstly, we propose a dynamic modeling method to build system model of knowledge base, this method utilizes the dynamic procedural nature of production rule to build system model, and it improves the modeling efficiency significantly; Secondly, a conditional transition system based on the standard transition system is given to represent system model, our conditional transition system contains the whole information about the actual state transition process, which solves the information loss problem of the static transition system built by static modeling method and improves the efficiency of error diagnosis.**

***Keywords-production knowledge base; verification; model checking; CTS; modeling***

## I. INTRODUCTION

Production knowledge base is widely used in various artificial intelligent systems due to its advantages of simplicity and clarity of representation. However, the construction of knowledge base is a transformation process performed by knowledge engineers who transform knowledge coming from domain experts into the form of computer representation, which introduces many quality effecting factors, such as knowledge errors and inputting errors result from human negligence, different kinds of hidden errors brought in by inaccurate and unilateral comprehension of the intention of domain experts, etc. These factors are the fundamental reason why there exists inconsistent, incomplete and even wrong knowledge[1] in various of production knowledge base. With the extensive application and the continuous expansion of the scale of knowledge base, knowledge errors are becoming complicated and diversified, and have already become a significant factors influencing the operating efficiency of different kinds of intelligent systems. Thus, correctness verification has become an absolutely necessary step on the building of knowledge base.

With the rapid development of production knowledge base system, there appears many kinds of verification technologies , such as method based on decision table[2], directed graph[3], Petri net[4–6], knowledge network[7], DNA operation[8] and adaptive reasoning[9] and so on. Though these methods are capable of finding different kind of knowledge errors to a different extent, the verification process of most of which is limited to the rule set itself. In fact, the dynamic dependent relationship of rules during running period of knowledge base is far more complicated than the static dependent relationship of rule set itself. The exposure of some hidden errors only happens during the running period of knowledge base system, and test[10, 11] is a common method to deal with this problem. However, test has a low efficiency, and its ability relies on the experience of testing personnel to a great extent. In addition, test covers just a small part of running paths of the system.

Compared with the methods mentioned above, model checking[12] is a general verification technique, it needs neither high degree interaction with users nor sophisticated expert knowledge, and is able to answer whether a system model satisfies the given property or not rapidly and automatically. A few methods[13–15] using model checking technology to debug and verify inconsistency and circulation of production knowledge base have been proposed by researchers. However, the modeling methods have been put forward by these model checking technologies are a kind of Static Modeling Method(SMM), the reason is that the dynamic procedural nature of production rules is ignored in the modeling process of SMM.

SMM contains two main steps. The first one is the generation of system states according to rule set, the second one is the generation of transition relations, which has to traverse the whole rule set for any two states. Although SMM is clear and easy to understand, it has two limitations: 1) The dynamic procedural nature that the state transition is triggered by rule condition and the transition target is determined by rule action is neglected, the immediate consequence is the modeling efficiency is low; 2) The transition relation is a subset of the Cartesian Product of state set. Though verification can be achieved on this standard transition system built by SMM, the final goal of the verification of production knowledge base is not to obtain error path given by model checker, but to find error location, i.e., to find a certain error is triggered by which rule condition and caused by the execution of which rule action. System model built by SMM, which we call it as Static Transition System(STS), does not contain these important information

which can be used for error diagnosis. Thus, it has to traverse rule set repeatedly to find error location, which will lead to a high time overhead.

A formal method using model checking technology to verify production knowledge base is proposed in this paper based on the studies on existing verification methods. This paper is structured as follows. First, we introduce the dynamic conditional transition caused by rule and the definition of Conditional Transition System(CTS) which contains the state transition information triggered by rule condition. Then Dynamic Modeling Method(DMM) is proposed to build CTS, the DMM algorithm and an example are given to specify the modeling process. Finally,we have discussed how to describe properties of the system by the use of LTL formula.

## II. MODELING

One of the most important and complicated step of model checking is the building of system model, i.e., a Transition System. SMM does not take the inducement and process of state transition under consideration, while it just investigates the relationship between state and state. As a matter of fact, state transition induced by production knowledge base is a dynamic changing process triggered by rule condition.

### A. Conditional Transition

**Definition1.** Expression $s \xrightarrow{r.con:r.act} s'$ denotes the conditional transition process triggered by rule condition, $r$ is rule labeling, $r.con$ and $r.act$ denote the condition and action of rule $r$ respectively.

The semantic of conditional transition is as follows. When system arrive at state $s$, rule matching will take place according to current variable values of state $s$, if condition of rule $r$, namely $r.con$ , is satisfied at state $s$, $r.act$ will be triggered and perform calculation on variables of state $s$, variable values which have been changed form a new state $s'$ , $s'$ can be the same as state $s$. This is because the operation performed by $r.act$ does not change the variable values on state $s$. Conclusively, if state $s$ satisfies $r.con$ , state $s'$ is arrived by performing $r.act$ on state $s$.

The influence of procedural rule on state transition can be described intuitively by the using of conditional transition. As to declarative rule, we can treat the rule conclusion as a variable assignment process, i.e., treat a declarative rule as a procedural rule.

### B. Building CTS

Let there are $m$ variables involved in production knowledge base, suppose $V = v_1, v_2, ..., v_m$ is the set of these variables, where $m > 0$ , and every variable has a determinate data type, such as Boolean, Integer and String, etc. Each data type has a data domain, let $D(v)$ denotes the domain of $v$, where $v \in V$ .

**Definition 2.** Let $C(V)$ is a finite set of Boolean conditions over variable set $V$. Here Boolean condition is a proposition logic formula having the form $\overline{v} \in \overline{d}$ , where $\overline{v} = (v_1, v_2, ..., v_n)$ is a n-tuple, where $v_i \in V$ , $1 \leq i \leq n$ ,$1 \leq n \leq m$ , and for every $1 \leq j \leq n$ , if $i \neq j$ holds, then $\overline{D} = D(v_1) \times D(v_2) \times ... \times D(v_n)$ holds. denotes the data domain of $\overline{v}$ .

For example, let Boolean proposition logic formula $(v_1, v_2) \in (x, y) \in IN^2 \mid y - x \leq 2$ be denoted as $v_2 - v_1 \leq 2$ for short, then $(v_1 > 2) \wedge (v_2 - v_1 \leq 2) \wedge (v_3 = true)$ is a valid proposition condition, where $v_1$ and $v_2$ are Integer, $v_3$ is Boolean.

**Definition 3.** Suppose $E(V)$ is a finite set over variable set $V$, the elements of $E(V)$ is composed of all possible combinations of variable values.

**Definition 4.** Let $f_{act} : E(V) \rightarrow E(V)$ is an action function defined by $V$, and $F$ denote the finite set composed by all action functions.

A rule in knowledge base can be regarded as a 2-tuple $r : (con, act)$ , where $r$ is labeling of the rule, $con$ represents rule condition, and $act$ denotes action or conclusion of a rule. According to definition 1, $con$ is a Boolean condition defined by $V$, thus $con \in C(V)$ holds; According to definition 2, $act$ is a action function defined by $V$, i.e., $act \in F$ holds. Suppose $R$ is a finite set defined by all the rules of knowledge base, then $R \subseteq C(V) \times F$ holds.

As the system model built by SMM does not contains information about the cause of state transition, which will lead to a high cost to locate error. Thus, a complete transition system built from knowledge base should include this important information. The definition of CTS is as follows.

**Definition 5.** Let Conditional Transition System(CTS) defined over variable set $V$ is a 6-tuple $(S, A, \hookrightarrow, S_0, AP, L)$, where

- $S$ is a set of states, $S = E(V)$,
- $A$ is a set of actions, $A \subseteq F$,
- $\hookrightarrow$ is a set of transitions, $\hookrightarrow \subseteq S \times C(V) \times F \times S$,
- $S_0$ is a set of initial states, $S_0 \subseteq S$,
- $AP$ is a set of atomic propositions, $AP = C(V)$,
- $L : S \rightarrow 2^{AP}$ is a labeling function.

Rule set of knowledge base can be converted into CTS naturally according to definition 1 and definition 5. The basic idea is as follows. Starting from the initial states, if the condition of a rule is triggered at some states, rule action will be performed on the state to get the target state. If the target state does not belong to $S$, then put it into $S$. Whether the target state belongs to $S$ or not, the transition relation between the source state and target state will be added into $\hookrightarrow$. A whole CTS will be constructed gradually via the iterative extension process above.

The following is the DMM algorithm used for building CTS, in which $S_0$ is the set of initial states, $O$ is a set of states having been traversed, $N$ is a set of states being added into CTS in a traversal, $A$ is the set of actions, $\hookrightarrow$ is

the set of transitions.

***DMM Algorithm***

1) Initialization Operation, add the initial states set $S_0$ to $S$ and the new states set $N$ separately, both action set $A$ and conditional transition set $\hookrightarrow$ are empty.
2) Get a state $s$ from new state set $N$, then remove this element from $N$.
3) Get a rule $r$ that has not been match with state $s$ from rule set $R$, then perform the matching operation between $s$ and $r$, if $s$ does not satisfy $r.con$, turn step 7.
4) If $r.act(s)$ is already in the state set $S$, turn step 6, else do the next step.
5) Add state $s$ to both the states set $S$ and the new states set $N$.
6) Add $(s, r.con, r.act, r.act(s))$ to the conditional transition set $\hookrightarrow$, and $r.act$ to the action set $A$.
7) If there still exists rules that have not been matched with state $s$, turn step 3.
8) If the new states set $N$ is empty, return states set $S$, conditional transition set $\hookrightarrow$ and action set $A$, else jump to step 2.

In this modeling method, system model is generated mainly from the dynamic extension of state set and transition set. The dynamic characteristic of DMM is that the state transition only happens when special conditions are satisfied, and the determination of the transition target, namely the process of the generation of new states, is dependent dynamically on action which is triggered by a rule condition.

### C. An example of CTS

In this section, we take knowledge base of ferryman problem as an example to analyze the modeling process of CTS. Ferryman has to ferry wolf, goat and cabbage from the initial riverbank to the target riverbank, he can carry only one object among wolf, goat and cabbage every time, or carry nothing. Wolf and goat or goat and cabbage should not be at the same riverbank without the ferryman. There are 8 rules in the production knowledge base used for solving this problem. Where f, w, g and c denotes ferryman, wolf, goat and cabbage respectively, the value of f, w, g and c can be 0 or 1, value 0 denotes these four objects are at the initial riverbank, and value 1 the target riverbank. The rule set of this knowledge base is as follows.

$r1: \ IF \ \ (f = 0) \ \ THEN \ \ (f + 1)$
$r2: \ IF \ \ (f = 1) \ \ THEN \ \ (f - 1)$
$r3: \ IF \ \ (f = 0, w = 0) \ \ THEN \ \ (f + 1, w + 1)$
$r4: \ IF \ \ (f = 1, w = 1) \ \ THEN \ \ (f - 1, w - 1)$
$r5: \ IF \ \ (f = 0, g = 0) \ \ THEN \ \ (f + 1, g + 1)$
$r6: \ IF \ \ (f = 1, g = 1) \ \ THEN \ \ (f - 1, g - 1)$
$r7: \ IF \ \ (f = 0, c = 0) \ \ THEN \ \ (f + 1, c + 1)$
$r8: \ IF \ \ (f = 1, c = 1) \ \ THEN \ \ (f - 1, c - 1)$

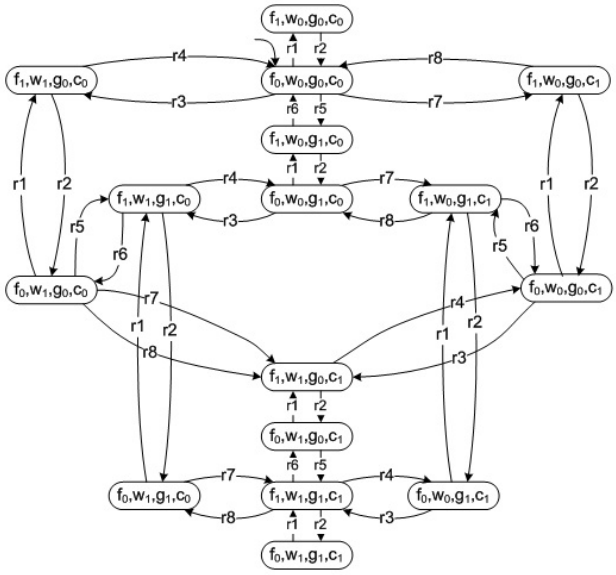


Figure 1. CTS of ferryman problem

In this system, initial states set includes only one state $(f = 0, w = 0, g = 0, c = 0)$ , denoted as $(f_0, w_0, g_0, c_0)$ for short. In the first iteration, four rules $r1.con$, $r3.con$, $r5.con$ and $r7.con$ are triggered on initial state, A new state $(f_1, w_0, g_0, c_0)$ can be achieved by performing $r1.act$ on the initial state, in a similar way, we can get new states $(f_1, w_1, g_0, c_0)$ , $(f_1, w_0, g_1, c_0)$ and $(f_1, w_0, g_0, c_1)$ via $r3.act$, $r5.act$ and $r7.act$ respectively. In this iteration, four conditional transition relations

$(f_0, w_0, g_0, c_0) \times r1.con \times r1.act \times (f_1, w_0, g_0, c_0)$,
$(f_0, w_0, g_0, c_0) \times r3.con \times r3.act \times (f_1, w_1, g_0, c_0)$,
$(f_0, w_0, g_0, c_0) \times r5.con \times r5.act \times (f_1, w_0, g_1, c_0)$,
$(f_0, w_0, g_0, c_0) \times r7.con \times r7.act \times (f_1, w_0, g_0, c_1)$,

are added into the transition set $\hookrightarrow$. Repeat this process until there is no new state can be arrived at a certain traversal. Fig. 1 shows the complete CTS of this production knowledge system.

### D. Property description

There are two necessary conditions which are needed to verify properties we care about. The first one is a transition system, another one is properties described by LTL or CTL formula. We can verify whether a transition system satisfies the given property or not by using NuSMV model checker. In this section, we mainly discuss the typical method of utilizing LTL to describe related properties. Ferryman problem is still taken as an example here.

1) Security property. Wolf and goat or goat and cabbage should not be at the same riverbank is a security property. The LTL formula

$$\phi_1 = G(w = g \vee g = c) \rightarrow g = f$$

describes this property, where $G$ is a temporal conjunction,

denotes all the future states. Put CTS and $\phi_1$ into NuSMV, we can obtain seven states $(f_1, w_0, g_0, c_0)$, $(f_1, w_1, g_0, c_0)$, $(f_1, w_0, g_0, c_1)$, $(f_0, w_1, g_0, c_0)$, $(f_0, w_0, g_0, c_1)$, $(f_0, w_1, g_1, c_0)$ and $(f_0, w_0, g_1, c_1)$, which violate $\phi_1$.

2) Liveness property. The existence of solution is a liveness property, i.e., there exists a transition path starting from initial state to target state, and this path has to satisfy security property. Target state can be described as LTL formula $\phi_2 = f \wedge g \wedge w \wedge c$. Then existence of solution is to find a path that satisfies formula $\phi_1 \sqcup \phi_2$. Thus, we can verify whether this CTS satisfies formula $\phi_3 = \neg(\phi_1 \sqcup \phi_2)$ or not, i.e., there is no path satisfies formula $\phi_1 \sqcup \phi_2$ . NuSMV will verify that this CTS does not satisfy $\phi_3$ .

3) Furthermore, we can verify whether there exists a solution of which the number of round trip is less than three. Let $\phi_4 = G(g \rightarrow Gg)$, Where $\phi_4$ means once goat arrives at the target riverbank, it will keep the state forever. This problem can be solved by find a path satisfies formula $\phi_5 = (\phi_1 \sqcup \phi_2) \wedge \phi_4$. NuSMV can verify $\neg\phi_5$ is true, i.e., there is no such solution.

## III. Performance analysis

### A. Computing cost of modeling

In order to build a transition system, SMM needs to generate states set of knowledge base at first, then it generates transition relation by matching rule set with all combinations of any two states. Thus the time complexity of SMM is $O(N^2 \times K)$ , while the time complexity of DMM is $O(N \times K)$ , because every state in CTS matches rule set only one time in the building process of DMM, where $N$ is state scale, $K$ is rule scale of knowledge base.

### B. Computing cost of error diagnosis

Suppose $\pi$ is the error path given by model checker, and $\pi$ is a finite state sequence $s_0, s_1, ..., s_n$, where $n \geq 0$, $|\pi| = n$ . For STS built by SMM, the computation of rule inference path for $\pi$ has to traverse the rule set with $n$ transitions, such as $s_0 \times s_1, s_1 \times s_2, ..., s_{n-1} \times s_n$, the computing cost is $O(|\pi| \times K)$; As to CTS, the computing cost is $O(|\pi|)$ because we can locate error directly from conditional transition relations.

## IV. Experimental results and analysis

An experiment was carried out to prove the correctness of performance analysis given in section 3. Experimental data comes from the Knowledge base of Social Insurance Audit Project, which is supported by National Natural Science Foundation of China(NSFC). The system models generated in this experiment and the experimental data are stored in database, and the database type is DB2 V9.1, the running environment is IBM x3950 server. SMM and DMM implemented on Netbeans 6.7 IDE to build transition system respectively under the same conditions. Table I shows the results of this experiment.

Table I
MODELING PERFORMANCE OF SMM AND DMM

| Rule scale | Algorithm | State scale | Transition scale | Comparison($10^3$) | Time cost(s) |
|---|---|---|---|---|---|
| 13 | SMM | 33 | 85 | 14.15 | 4.193 |
| 22 | SMM | 81 | 299 | 144.34 | 31.097 |
| 30 | SMM | 162 | 874 | 787.32 | 154.927 |
| 44 | SMM | 382 | 3323 | 6420.65 | 1196.200 |
| 51 | SMM | 693 | 9355 | 24492.69 | 4505.320 |
| 13 | DMM | 33 | 85 | 2.81 | 1.736 |
| 22 | DMM | 76 | 293 | 22.27 | 8.191 |
| 30 | DMM | 152 | 839 | 127.53 | 34.609 |
| 44 | DMM | 348 | 3127 | 1088.20 | 283.163 |
| 51 | DMM | 613 | 8628 | 5289.06 | 1064.375 |

Table II
ERROR LOCATING PERFORMANCE OF STS AND CTS

| Rule scale | TS Type | $\|\pi\| =$ 10(ms) | $\|\pi\| =$ 20(ms) | $\|\pi\| =$ 30(ms) | $\|\pi\| =$ 40(ms) | $\|\pi\| =$ 50(ms) |
|---|---|---|---|---|---|---|
| 13 | STS | 23.15 | 46.31 | 69.46 | 92.62 | 115.78 |
| 22 | STS | 39.18 | 78.37 | 117.56 | 156.75 | 195.93 |
| 30 | STS | 53.43 | 106.87 | 160.31 | 213.75 | 267.18 |
| 44 | STS | 78.37 | 156.75 | 235.12 | 313.50 | 391.87 |
| 51 | STS | 90.84 | 181.68 | 272.53 | 363.37 | 454.21 |
| 13 | CTS | 1.77 | 3.5578 | 5.336 | 7.11 | 8.89 |
| 22 | CTS | 1.78 | 3.56 | 5.350 | 7.13 | 8.91 |
| 30 | CTS | 1.78 | 1.79 | 3.598 | 5.39 | 7.19 |
| 44 | CTS | 1.74 | 3.49 | 5.235 | 6.98 | 8.72 |
| 51 | CTS | 1.75 | 3.51 | 5.279 | 7.03 | 8.79 |

According to data in table I, the modeling efficiency of DMM is obviously higher than SMM. Moreover, the scale of CTS built by DMM is slightly less than STS built by SMM, the reason is that CTS is extended dynamically by DMM on the basis of states it can reach via the rule set, which ensures there is no redundant state in CTS. Furthermore, the error diagnosis efficiency has been tested on transition systems constructed by SMM and DMM, table II shows the experimental results. It is clearly that the time cost of error location on STS is directly proportional to the scale of rule set and the length of error path, while this time cost on CTS is only related to the length of error path.

## V. Conclusion

In this paper, we have proposed a formal method to verify production knowledge base by the use of model checking technology. In this method, DMM algorithm is given to build CTS. Compared with SMM, DMM improves the modeling efficiency, and CTS constructed by DMM includes enough information about state transition, which solves the information loss problem caused by SMM, and reduces the computing cost of error diagnosis remarkably. However, the building process of CTS does not take the influence of inference control strategy into consideration. Further works will study on the influence of inference control strategy in the modeling process of production knowledge base.

## ACKNOWLEDGMENT

This work is sponsored by the National Natural Science Foundation of China under grant number 60873038.

## REFERENCES

[1] Chen Shifu, Pan Jingui and Xu Dianxiang. Verifying consistence and redundancy of production system knowledge base. Chinese Journal of Computers, vol. 15, no. 9, 1992, pp. 670-675.

[2] B.J. Cragun, and H.J. Steudel, "A decision-table-based processor for checking completeness and consistency in rule-based expert systems," International Journal of Man-Machine Studies, vol. 26, no. 5, 1987, pp. 633-648.

[3] M. Ramaswamy, S. Sarkar, and C. Ye-Sho, "Using directed hypergraphs to verify rule-based expert systems," Knowledge and Data Engineering, IEEE Transactions on, vol. 9, no. 2, 1997, pp. 221-237.

[4] D.L. Nazareth, "Investigating the applicability of Petri nets for rule-based system verification," Knowledge and Data Engineering, IEEE Transactions on, vol. 5, no. 3, 1993, pp. 402-415.

[5] S.J.H. Yang, A.S. Lee, W.C. Chu, and Y. Hongji, "Rule base verification using Petri nets," Computer Software and Applications Conference, 1998. COMPSAC '98. Proceedings. The Twenty-Second Annual International, pp. 476-481.

[6] B. Bostan-Korpeoglu, and A. Yazici, "Using fuzzy Petri nets for static analysis of rule-bases," 19th International Symposium on Computer and Information Sciences (ISCIS 2004), pp. 72-81.

[7] S. Beflefeuille, L. Capus, N. Tourigny, and J.B. Welcomme, "An approach using a knowledge network to verify consistency of rule bases," 7th World Multiconference on Systemics, Cybernetics and Informatics, pp. 63-67.

[8] Y. Chung-Wei, and C. Chih-Ping, "Molecular Verification of Rule-Based Systems Based on DNA Computation," Knowledge and Data Engineering, IEEE Transactions on, vol. 20, no. 7, 2008, pp. 965-975.

[9] Yan Lixiang, Tan Zheng. Knowledge refinement of self-adaptive reasoning. Journal of ZheJiang University(Science Edition) , vol. 31, no. 1, 2004, pp. 48-50.

[10] D. Hamlet, and Z. Marvin, "Software Quality, Software Process, and Software Testing," Advances in ComputersVolume 41, Elsevier, 1995, pp. 191-229.

[11] P.G. Frankl, and E.J. Weyuker, "Testing software to detect and reduce risk," Journal of Systems and Software, vol. 53, no. 3, 2000, pp. 275-286.

[12] G.O. Clarke E M, Peled D. , "Model Checking," MIT Press,Cambridge, 1999.

[13] L. Hui, Z. Jinglei, and L. Ruzhan, "Model Checking a Rule-based Parser," Natural Language Processing and Knowledge Engineering, 2007. NLP-KE 2007. International Conference on, pp. 221-228.

[14] L. Hui, Z. Jinglei, and L. Ruzhan, "Toward the Formal Verification of a Unification System," Book Toward the Formal Verification of a Unification System, Series Toward the Formal Verification of a Unification System 39, ed., Editor ed., 2009, pp. 399-408.

[15] K.X. Desheng Xu, Dezheng Zhang,Huangsheng Zhang, "Model Checking the Inconsistency and Circularity in Rule-Based Expert Systems," International Journal of Computer and Information Science, vol. 2, no. 1, 2009.