Martin L, Romanovsky A. [Stochastic activity networks for the verification of knowledge bases](#). In: *International Workshop on Software Engineering for Resilient Systems (SERENE 2017)*. 2017, Geneva, Switzerland: Springer Verlag.

Copyright:

The final publication is available at Springer via https://doi.org/10.1007/978-3-319-65948-0_3

DOI link to article:

https://doi.org/10.1007/978-3-319-65948-0_3

Date deposited:

06/11/2017

Embargo release date:

11 August 2018

This work is licensed under a [Creative Commons Attribution-NonCommercial 3.0 Unported License](#)

Stochastic Activity Networks for Modelling Knowledge Bases

Luke Martin, Alexander Romanovsky

Centre for Software Reliability, Newcastle University
Newcastle upon Tyne, United Kingdom
{luke.martin, alexander.romanovsky}@ncl.ac.uk

Abstract. This paper proposes the application of the stochastic activity networks (SANs) formalism for the verification and prototyping of knowledge-based systems. Typical structural errors of a KBS are redundancy, inconsistency, incompleteness and circularity, which affect the general reliability of the system. Since the verification is based on SANs and their incidence matrix, the rules need to transform rules into a SAN first, then derive an incidence matrix from the network. In order to let fuzzy rule-based systems detect above the structural errors, we are presenting a SANs-based mechanism. This mechanism consists of three phases: rule normalization, rules transformation, and rule verification. Rules will be first normalized into Horn clauses, then transform the normalized rules into a high-level network, and finally we verify these normalized rules. In addition, we are presenting our approach to simulate the truth conditions which still hold after a transition firing and negation in SANs for rule base modeling. In this paper, we refer to fuzzy rules as the rules with certainty factors, the degree of truth is computed in an algebraic form based on state equation which can be implemented in matrix computation in SANs. Therefore, the fuzzy reasoning problems can be transformed as the linear equation problems that can be solved in parallel. We use Mobius to realize the mechanism presented fuzzy rules in this paper.

Keywords: Knowledge Engineering, Reliability, Verification, Formal Methods, Stochastic Activity Networks, Mobius

1 Introduction

Knowledge-based systems (KBSs) have been widely used in many real-world applications and are fast emerging as the latest generation of embedded and real-time systems. The purpose of a KBS is to emulate a human expert in making a decision to solve a complex domain problem. The core component of a KBS is the knowledge-base, which is a rule-based system that lists facts, rules, abstract data models and interface mechanism as knowledge of a specific domain area. Users of a KBS can interact with the system with queries as the same way that they consult with the domain experts for obtaining knowledge. In general, the procedures for domain experts to accumulate knowledge and expertise is incremental and intuitive, thus, a knowledge base needs to be constructed incrementally through an evolutionary

development process. One of the major issues in building a knowledge base, however, is that it is constructed by consulting numerous domain experts, who are likely to have conflicting or varying views. Therefore, it is not surprising that many rules in a rule base may have structural errors. As addressed in [11], [13], structural errors can be classified as: redundant - due to redundant and subsumed rules; inconsistent - due to conflict rules; incomplete - due to missing rules; and cyclic - due to infinite inference. Each of these clearly have adverse consequences for reliability and a method for detecting these errors is required to ensure reliability. Researchers have been proposing the use of Petri nets and their extensions to model non-fuzzy rule-based systems [8], [10], [14]. While there are many advantages to applying the Petri net formalism, there are also several advantages to representing rules as stochastic activity networks (SANs), due to its capabilities in supporting fuzzy conditions before a transition can be made. Consequently, once rules have been transformed into SANs, the uncertainty reasoning problems can be transformed as the linear equation problems, which can be solved in parallel [15] and designers ensure a high quality of the system being developed [10]. The mechanism we propose is based on [1] and consists of three phases: rule normalization, rules transformation, and rule verification. As the scope of this paper is limited to developing an approach, a high level description of the method is given with preliminary results, with future work focusing on expanding the approach and deriving a more complete set of results. We have implemented a basic SAN model based on this approach using Mobius.

The remainder of the paper is organized as follows: Section 2 discusses related work of rule base verification and fuzzy rule reasoning with high-level Petri nets and provides a brief overview of the SANs formalism. Section 3 presents types of structural errors in a rule base and the issues of rule base verification and reliability estimation. Section 4 addresses our SANs-based approach and the phases of rule normalization, rule transformation, and rule verification. We conclude the paper with our future research in Section 5.

2 Related Work

As addressed in [11], structural errors in a KBS could be classified as redundancy, conflict, and incompleteness. Each of these affect the reliability of a KBS and must be addressed and repaired. In the case of redundancy, it includes redundancy in pairs of rules, subsumed in pairs of rules, redundancy in chains of inference, and subsumed in chains of inference. In the case of conflict, it includes conflict with initial facts, conflict between the deduced facts, and circularity. In the case of incompleteness, it includes unnecessary inputs, dead-ends, unreachable goals, and missing facts. Besides various mechanisms of rule base verification, there are many researches targeted to using the graphical and mathematical natures of Petri nets (PN) for fuzzy rule reasoning. Researchers have been proposing the extension of Petri nets to fuzzy Petri nets (FPN) and using FPN to model and reasoning fuzzy rule-based systems. We will summarize these mechanisms in this section.

2.1 Rule Verification

A Nguyen et al. [13] describes potential problems about a knowledge base and introduces their tool, CHECK, detecting errors on the Lockheed Expert System (LES). The LES is a rule-based building tool, which represents factual data in its frame database and heuristic and control knowledge in its production rule. The LES allows knowledge engineers to use both data-driven and goal-driven rules. In the knowledge base, consistency and completeness must be insured. There are five steps to implement the rule checking. At the first three steps, CHECK finds the relationship of clauses and records the information in 2D table of interclause relationship. The some problems can be detected by using the table to deduce rule relationships, SAME (redundant), CONFLICT, SUBSET (subsumption), SUPERSET (subsumption), UNNECESSARY CLAUSE, or DIFFERENT. Then, find the properties of unreachable conclusions, dead-end goals, and dead-end IF conditions in the 2D table. The 2D table also can generate the dependency chart for detecting the circularity. At last, this paper figures out how certainty factors affect the checking. Zhang and In [10], it was noted that Nazareth proposed an approach based on Petri nets to verify a rule-based system. His approach adds new arcs backward to all the input places and a new input place to ensure every transition will fire at most once and ensure the conditions will hold even if the transition has fired. He compares the difference of marking before and after firing any sequence of transitions. If a place has no token before firing any firing sequence and the place has two tokens after firing two individual sequences of transitions, then the system has redundancy or conflict. If a place has one token before firing a sequence of transitions and has two tokens after firing a sequence of transitions, then the system has circularity. If a place has no token before all transitions are fired and still has no token after all transitions are fired, then it has a dead end or unreachable goal.

2.2 Stochastic Activity Networks

Stochastic activity networks (SANs) are stochastic extensions to Petri nets and to the best of the author's knowledge have not been applied for verifying rules of KBSs. SANs have the modelling power of Petri nets and allow a compact representation of systems. They consist of: places, activities, input gates and output gates. - Places can be seen as a state of the modelled system. Each place of a SAN contains a certain number of tokens which represents the marking of the place. Places are represented graphically by circles. - Activities represent actions of the modelled system that could take some specified amount of time to complete. They are similar to transitions in ordinary Petri nets, and are of two types: timed and instantaneous. Timed activities have durations that impact the performance of the modelled system such as a packet transmission time. This duration can be stochastic. Instantaneous activities represent actions that complete or fire immediately when enabled in the system. Activities are graphically represented by thick lines for the timed ones, and thin lines for the instantaneous ones. Unlike autonomous Petri nets, SANs allow the use of

uncertainties associated with the completion of an activity. It is called Case probabilities, and is represented graphically by small circles on the right side of an activity (see Figure 2). Each case stands for a possible outcome, such as a routing choice in a network, or a failure mode in a faulty system. So each activity in the SAN can have a probability distribution associated with its cases. Moreover, this distribution can depend on the marking of the network at the moment of completion of an activity. This shows how SAN could be a high level modelling formalism.

3 Structural Errors in Rule-Based Systems

In this paper, we are interested in finding structural errors and the set of rules causing the structural errors, which affects the reliability of the knowledge base. The reasons of structural errors may be due to rule conflicting, missing/mismatched condition, and conclusion. As pointed out in [7], [11], [12], [13], we have identified four kinds of rules, which may commit structural errors. They are redundancy, inconsistency, incompleteness, and circularity rules. Inconsistency rules result in conflict, which is the direct source of incorrect rule derivation. Redundancy rules increase the size of rule base and cause non-necessary reasoning. Incompleteness rules prohibit rule bases from activating certain normal rule derivation. Circularity rules will force the rule base to run into an infinite loop of reasoning. Following, we describe those structural errors and give example to express.

Inconsistency. Inconsistency results in a conflict of facts and must be resolved for correct functioning of a KBS. This means a set of rules are conflicting if contradictory conclusions can be derived under a certain condition. An example of inconsistency rules is as follows:

```

R1  : -> P1
R2  : P1 -> P2
R3  : P1 -> P3
R4  : P2 -> P3

```

Incompleteness. Incompleteness occurs when there are missing rules in a rule base. Except the rules for representing facts and queries, a rule is called as a useless rule if the rule's condition (conclusion) cannot match other rules' conclusion (condition). The unmatched conditions are called dangling conditions, while the unmatched conclusions are called dead-end conclusions. Mostly, the reasons of useless rules are due to some missing rules. An example of incompleteness rules is as follows:

```

R1  : -> P1
R2  : P1 AND P3 -> P2
R3  : P1 -> P4
R4  : P4 -> .

```

Redundancy. Redundancy occurs when unnecessary rules exist in a rule base. Redundancy rules not only increase the size of the rule based but also may cause additional useless inferences. Redundancy is classified as being of two kinds: forms-redundant rule and subsumed. A rule is forms-redundant with respect to a conclusion if the conclusion can be reduced from other rules under the same conclusions. A rule is subsumed with respect to a conclusion if the conclusion can be reduced from both rules and the condition part of the former is included in the condition part of the latter. An example of redundancy rules is as follows:

```
R1 : P1 AND P3 -> P2
R2 : P1 AND P3 -> P2
R3 : P1 -> P2
R4 : P4 -> P5
R5 : P4 -> P5 AND P6
R6 : P2 -> P7
R7 : P1 -> P7
```

Circularity. Circularity occurs when several inference rules have circular dependency. Circularity rules can cause infinite reasoning and must be broken. If the cycle is formed within a rule itself, then we called such circular rule as self-circular rule. An example of circularity rules is as follows:

```
R1 : P1 -> P2
R2 : P2 -> P3
R3 : P3 -> P1
R4 : P1 AND P4 -> P4
R5 : P5 -> P5
```

4 Stochastic Activity Networks for Knowledge-Based Modelling and Verification

For effective verification using SANs, it is important that the KBS be formulated in manner so as to precisely represent the working of the original rule set. Furthermore, within this representation, all errors should be detectable, involving pairs and chains of rules. For the present, the verification is limited to systems without uncertainty, since it provides for general coverage of errors. The verification strategy also employs definitional domain knowledge, so as to improve error detection. This knowledge is easily procured and encoded for any domain. The formulation of a KBS as a SAN involves the representation of each rule as a transition. Tokens are used to indicate validity or establishment of facts. However, some constraints are to be placed on the input and output functions for proper transformation. In its most primitive form, each rule will have all the antecedents as input places and the consequents as output places for the transition in question. It would also require that (4 the in-degree and out-degree associated with places be set to 1. i.e.

$$\begin{aligned}\#(p_i, O(t_j)) &\in \{0, 1\} \\ \#(p_i, I(t_j)) &\in \{0, 1\}\end{aligned}$$

Thus the rule of the form:

$$A \rightarrow B \ \& \ C$$

Would be represented as

$$I(t_1) = \{p_a\} \text{ and } O(t_1) = \{p_b, p_c\}$$

This indicates that the firing of transition t_1 will add both B and C to the set of known clauses. However, in its current state, transition t_1 cannot be fired as there are no tokens in its input places, viz., $\{A\}$. However, information concerning the validity of A is lost as a result of the firing. This can be rectified by including A in the set of output places when t_1 is fired indicating A, B, and C as known facts. This represents an improvement, but is still problematic in that transition t_1 can be fired again, more so, at will, thus increasing the tokens in places A, B, and C ad infinitum. To counteract this, each transition will have associated with it, a special place with a single token in it. Additionally, the transition can be fired only once, indicating a persistent Petri net with no information loss. The use of definitional knowledge in a domain does pose some problems. Most KBSs will employ some such knowledge, since it is natural information as far as the experts are concerned. This knowledge may take the form of data abstraction or mutual exclusiveness. Data abstraction rules cover qualitative abstraction, definitional abstraction and generalization [SI. Qualitative abstraction involves the transformation of a value on one dimension (usually quantitative) to an equivalent value on another dimension, as in “if the patient’s temperature exceeds 104OF, then the patient’s temperature is high.” Definitional abstraction involves the use of related concepts to describe the same property of an object, as in “if the patient’s blood count is low, then that patient is anaemic.” Generalizations, on the other hand tend to relate concepts through hierarchical structuring, e.g., “if the patient is male, then the patient cannot be pregnant.” Mutually exclusive knowledge relates different classes of a concept, as in “if the patient has infection X, then the patient cannot have infection Y.” Knowledge of this sort is frequently employed by domain experts when reasoning, but may not be explicitly incorporated into the system. Any verification strategy that excludes this knowledge would be suspect in that it would ignore potential errors, and possibly indicate the possibility of error when none is present. Incorporating knowledge about abstraction in a verifier is relatively straightforward; the abstracted knowledge can be represented as rules and added to the current rule set. Mutually exclusive knowledge can be handled more elegantly through the use of a single place denoting inconsistency, as opposed to the use of several rules denoting combinations of incompatible clauses. Adopting a more formal notation, let the rule set R consist of k rules, with $T_i = R$ representing individual rules.

Using graph theory for rule verification has been addressed in [13]. We can use the approach of traditional SANs analysis methods for detecting structure errors. The

dynamic simulated SAN model can be used for explaining processes of logic reasoning and visualizing the structure for engineers detecting errors, where Mobius can be applied for carrying out this analysis.

5 Conclusion and Future Research

In the paper, we have briefly presented the preliminaries of normalizing and ordering rules of a KBS and explained the reasoning for this. Once the rules have been transformed into high-level SANs, the approach should (in theory) be able to answer queries over the net structure and compute the degree of truth to answers, however, further technical investigation is required for evaluating this. In future work, our focus will be to apply the method presented to an example case study of a KBS, where an ongoing PhD research project is focused on developing an advisory system for use in railway traffic planning. The knowledge base of the advisory system has been designed and developed, but not yet fully evaluated, where it is hoped that this presented mechanism can evaluate and verify the knowledge base. We also plan to integrate the approach into a reliability estimation method that can help in motivating and justifying design decisions at each iterative stage of development. We also aim to concentrate on formalizing and automating the transformation between rules and SANs with certainty factors. Furthermore, we have addressed types of structural errors in a KBS and proposed a SANs formalism for verifying these structural errors. By following the four phases as presented in this paper, we can automatically detect types and causes of error. We will further integrate the modeling and reasoning strength of SANs and fuzzy set theory to extend the usage of our approach to real-world applications such as multimedia synchronization, telecommunication, reuse, and maintenance. We have applied Mobius, which allows the drawing, execution, and reasoning of SANs.

Acknowledgements

The authors would like to thank Siemens and EPSRC for their support in this PhD project and for the support of Newcastle University.

References

1. R. Agarwal and M. Tanniru, "A Petri-Net Based Approach for Verifying the Integrity of Production Systems," *Int'l J. ManMachine Studies*, pp. 447-468, 1992.
2. S. Ahson, "Petri Net Models of Fuzzy Neural Networks," *IEEE Trans. Systems, Man, Cybernetics*, vol. 25, no. 6, pp. 926-932, June 1995.
3. A.J. Bugarin and S. Barro, "Fuzzy Reasoning Supported by Petri Nets," *IEEE Trans. Fuzzy Systems*, vol. 2, no. 2, pp. 135-149, May 1994.
4. T. Cao and A.C. Sanderson, "Task Sequence Planning Using Fuzzy Petri Nets," *IEEE Trans. Systems, Man, Cybernetics*, vol. 25, no. 5, pp. 755-768, May 1995.

5. S.M. Chen, J.S. Ke, and J.F. Chang, "Knowledge Representation Using Fuzzy Petri Nets," IEEE Trans. Knowledge and Data Eng., vol. 2, no. 3, pp. 311-319, Sept. 1990.
6. P.L. Hammer and A. Kogan, "Essential and Redundant Rules in Horn Knowledge Bases," IEEE Proc. 28th Hawaii Int'l Conf. System Sciences, pp. 209-218, 1995.
7. C.H. Wu and S.J. Lee, "A Token-Flow Paradigm for Verification of Rule-Based Expert Systems," IEEE Trans. Knowledge and Data Eng., vol. 30, no. 4, pp. 616-624, July/Aug. 2000.
8. T.A. Nguyen, W.A. Perkins, T.J. Laffey, and D. Pecora, "Knowledge Base Verification," AI Magazine, no. 49, pp. 69-75, 1987.
9. K. Jensen, "Coloured Petri Nets: A High Level Language for System Design and Analysis," High-Level Petri Nets, K. Jensen and G. Rozenberg, eds., pp. 44-122, 1991.
10. A. Konar and A.K. Mandal, "Uncertainty Management in Expert Systems Using Fuzzy Petri Nets," IEEE Trans. Knowledge and Data Eng., vol. 8, no. 1, pp. 96-105, Feb. 1996.
11. T.J. Laffey, W.A. Perkins, and T.A. Nguyen, "Reasoning About Fault Diagnosis with LES," Proc. IEEE, pp. 13-20, 1986.
12. C. Lin, A. Chaudhury, A.B. Whinston, and D.C. Marinescu, "Logical Inference of Horn Clauses in Petri Net Models," IEEE Trans. Knowledge and Data Eng., vol. 5, no. 3, pp. 416-425, June 1993.
13. N.K. Liu and T. Dillon, "An Approach towards the Verification of Expert Systems Using Numerical Petri Nets," Int'l J. Intelligent Systems, vol. 6, pp. 255-276, 1991.
14. C.G. Looney and A.R. Alfize, "Logical Control via Boolean Rule Matrix Transformation," IEEE Trans. Systems, Man, Cybernetics, vol. 17, no. 6, pp. 1077-1082, Nov./Dec. 1987.
15. C.G. Looney, "Fuzzy Petri Nets for Rule-Based Decisionmaking," IEEE Trans. Systems, Man, Cybernetics, vol. 18, no. 1, pp. 178-183, Jan./Feb. 1988.