Verification of a Rule-Based Expert System by Using SAL Model Checker

Maria Ulfah Siregar[1], Sayekti Abriani[2]
[1]*Department of Informatics, Graduate Program, Faculty of Science and Technology*
UIN Sunan Kalijaga
Yogyakarta, Indonesia
[1]maria.siregar@uin-suka.ac.id, [2]yekti001@gmail.com

***Abstract*— Verification of a rule-based expert system ensures that the knowledge base of the expert system is logically correct and consistent. Application of verification into a rule-based expert system is one approach to integrate software engineering methodology and knowledge base system. The expert system, which we has built, is a rule-based system developed by using forward chaining method and Dempster-Shafer theory of belief functions or evidence. We use Z language as the modelling language for this expert system and SAL model checker as the verification tool. To be able to use SAL model checker, Z2SAL will translate the Z specification, which models the system. In this paper, we present some parts of our Z specification that represent some parts of our rule-based expert system. We also present some parts of our SAL specification and theorems that we added to this SAL specification. At the last, we present the usage of SAL model checker over these theorems. Based on these model-checking processes, we argue that the results are expected. This means that each of theorems can be model checked and the outputs of those model checking are the same as the outputs that we obtain from manual investigation; either it is VALID or INVALID. Other interpretation of the model check's results is some parts of our rule-based expert system have been verified.**

Keywords*— *verification, expert system, rule-based system, Z2SAL, SAL model checker.

I. INTRODUCTION

Integration of validation and verification to a system is inevitably abundant evidence now, especially on a complex system. It is unfortunate if a system cannot be evaluated which limit our ability to use such a system [1].

Any field of intelligent system, such as an expert system, is no exception needing verification and validation, though verification and validation are two distinct activities [2]. The purpose of V & V is to cut errors from any intelligent system. Another purpose is to certify system correctness [3].

As one of formal languages, the usage of Z in academia [4] and industry has increased significantly. Z is a specification language, which has universal purpose. Z uses mathematics to state systems, schemas to build and modularized the specification [5]. This language has also the international standard. Z produces more formal specifications and they are ambiguity free which mechanic analyzation can be performed on them [6].

Specifications of a system will make verification of the system easier because we can perform the verification in the beginning stage of the development. Thus, it can decrease cost in implementation and test phases [7][8][9]. Formal methods are suggested to use in the development of critical systems [10]. Formal methods are a set of mathematical based tool which is the most promising techniques that allow the development of a complete, precise, and correct specification or model for system behavior and properties [11][12]. It also allows analyzing complex software systems. For this reason, we modelled a rule-based expert system as a Z specification to verify this expert system by using model checking techniques. We use Z2SAL [13] to translate our Z specification to a SAL specification which then can be verified by SAL model checkers [14].

II. RELATED WORKS

As mentioned in [15][2], several techniques have been proposed to verify rule-based systems to detect inconsistencies in knowledge bases, such as checking rules pair-wisely, implementing multiple rules in longer inference chains, using some graphical notation such as Petri nets and graphs, or using Algebraic methods. All of these approaches are before 2000.

A relatively new approach is found in [2], which is to use formal method to verify a rule-based expert system. Another approach is to use graph rewriting-based solution to verify and validate a rule-based expert system [16]. Unfortunately, we could only find both paper relating to verification or validation of rule-based expert systems that are published at least 5 years ago.

We use the same approach as the one found in [2], which is to use formal method for verifying a rule base expert system. However, our research uses different formal language. We chose Z formal language to model our rule-base expert system because this language can model a system more formal and free from ambiguity [17]. Furthermore, since Z language has schemas, we can use these properties to present states. For the verification, we use SAL model checker. Our approach has several advantages over other approaches:

- By using Z formal language, we can design our rules as predicates in schemas. These predicates are easily defined using logical operator supported by Z.
- Providing this Z specification modelled our expert system, we can translate it into another specification in SAL language by using Z2SAL.
- We could input the resulted SAL specification to SAL model checker for verification.

III. Our Rule-Based Expert System

An expert system is "a computer system that emulates, or acts in all respects, with the decision-making capabilities of a human expert" [18]. Knowledge-based systems as one type of expert systems are worthy in cases that are difficult to solve by using purely algorithmic or mathematical solutions [19]. One technique to represent or store knowledge on a knowledge-based system is by representing knowledge from experts as rules, which are declarative, in the form of "if antecedent then consequent". The antecedent clause is a test, which evaluate to True or False [20].

Increasing interest in AI with the management of uncertainty and evidential reasoning, resulted some methods [21]. One of them is Dempster-Shafer theory of evidence. This theory is claimed as a promising improvement on traditional approaches to decision analysis [22].

Our rule-based expert system serves as an expert gives recommendation to home medication for mild ingestion diseases [23][24]. There are 20 symptoms of these diseases and 6 mild digestive diseases, see Table 8 in [24]. A set of rules represent relations of these symptoms and mild digestive diseases.

This expert system uses uncertainty reasoning based on Dempster-Shafer. Calculation with Dempster Shafer requires probabilities of density functions. This density represents a belief value in indications of a sickness. We obtained these densities from a pharmacist. Table 9 in [24] shows those densities.

A user does consultations to this system to know what the sickness is and medication that will help to give first aid to the sickness. This user performs a consultation by entering symptoms of his disease.

IV. Model Checking Our System

The model checking method is a proper choice when compared with methods relying upon simulation, testing and deductive reasoning [25]. Verification by model-checking techniques is a well-established area of research [26]. Proficiency in mathematical disciplines is not necessary available to model check specifications [25].

Although it has advantages, there are also drawbacks. First, it only applies to finite state systems, and second, these cannot be so large since it can suffer from state space explosion problems [24][26][27]. These are due to the search strategy, which uses an exhaustive searching of the state space of a system using suitable graph algorithms [24][26].

We use SAL model checker to verify our system. To do this, we should have a SAL specification of our system.

SAL is a framework, which is used to change perceptions and implementations of model checkers and theorem provers. These perceptions and implementations at first were based on verification to a calculation of properties or symbolic analysis such as abstraction, slicing and composition [28]. SAL combines some different tools such as abstraction, program analysis, theorem proving and model checking towards a symbolic analysis of transition systems [14]. The current version of SAL is 3.3 which can be downloaded from [29]. The SAL language syntax is given in [14].

We added six theorems to our SAL specification. This specification is the result of translation by Z2SAL. Z2SAL [13] is a translator of a Z language specification into a SAL language specification [14].

A SAL file consists of a SAL module and/ or several SAL contexts. The module defines a transition system of Z states [30]. The outline of a SAL module is as follows:

```
State : MODULE =
 BEGIN
        INPUT ...
        LOCAL ...
        OUTPUT ...
        INITIALIZATION [ ... ]
        TRANSITION [
        ...
 ]
END
```

The SAL context declares types, constants, modules, and modules properties [14]. Z2SAL defines several Z mathematical tool-kits, which are necessary for the related Z specification, in separated but integrated SAL context files. More information about Z2SAL is provided in [31]. It includes also a downloadable version of this translation tool.

We have added six theorems to our SAL specification. The first three and the last two theorems represent safety properties. The first theorem says that it is always the case that the unavailability of a symptom indicates there is no infection. The second theorem says that it is always the case that the absent from `C` disease shows that there is neither `G1`, `G5`, `G6`, `G7`, `G8`, nor `G9` symptom exists. The third theorem in this classification asserts that the same symptom will never exist more than once. The five theorem is the opposite of the first theorem. We add this theorem to see how SAL model checker generates the counterexample. The last theorem says that if a symptom is not `G1` then in the future that symptom could be `G1`.

Safety properties will be proven by using forward reachability method in sal-smc, as a default [29]. A safety property asserts that nothing bad happens through execution of a system [32].

The fourth theorem represent liveness property. Liveness properties assert that something good eventually happens [32]. Our fourth theorem says that it is always the case that the present in G1 symptom indicates the infection of either A or C disease.

V. METHOD

Based on the expert system, we designed a Z specification for this system manually. It is emerged now the opposite way; to design formal models of systems, model inference is used which is combined with expert systems [33].

We declared diseases to be values for a global variable Decision. The same also applies to symptoms; they are declared values of a global variable, namely TypeS.

Rules in our expert system that relate symptoms with diseases are specified as predicates in our state schema. Each rule is presented with a symptom that implies diseases. The state schema declares several variables that represent a disease, a symptom and densities. As Z2SAL has not yet supported real numbers, we declare densities with a type of integer.

We declare initial values in the initialization schema. We include also in this schema the assigned values for densities, which are shown on Table 9 in [23].

The last schema is an operational schema. In this schema, we specify predicates that represent calculations of new densities of potential diseases. Another process here is to decide which disease is caught based on the calculation.

This Z specification was translated into the SAL specification by using Z2SAL. Then, we presented our theorems as explained above.

Following flow chart on Fig. 1 shows our works. As can be seen from Fig. 1, the first process is to design a Z specification of our rule-based expert system. We do not represent the expert system entirely in our Z specification. It is because our difficulty in representing some parts of the system. We designed three schemas in our Z specification: a state schema, an initial schema, and one operation schema. Our state schema, namely ruleBase, represents rules in our rule-based expert system. There are 20 rules, which indicates there are 20 symptoms in our rule-based expert system. The initial schema specifies initial values for the state variables. The operational schema, namely advise, shows how to conclude what sickness is somebody has based on the symptoms.

The second process is to translate our Z specification using Z2SAL translator. Fortunately, Z2SAL can translate our Z specification though there are manual modifications in some places. These modifications are discussed further in the next section, Result and Discussion.

After the SAL specification and some mathematical context files are generated by Z2SAL, we added some theorems in the resulted SAL specification (not in the mathematical context files). The explanation to these theorems can be read in other parts of this paper.

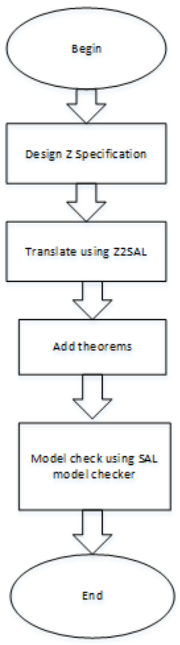

Fig. 1. Flow chart of our works

The last process is to model check the SAL specification. We use SAL model checker to perform this model checking. Again, the discussion on this process can be read in the next section.

Our result and discussion are given in the following section. It begins with the presentation of our Z specification.

VI. RESULT AND DISCUSSION

We present some of our Z specification as shown here:

Decision ::= A | B | C | D | E | F | clear

TypeS ::= G1 | G2 | G3 | G4 | G5 | G6 | G7 | G8 | G9 |G10 | G11 | G12 | G13 | G14 | G15 | G16 | G17 | G18 | G19 | G20 | nothing

The above declarations represent global variables that have enumerated values. These Z paragraphs have no box to declare free types. ::= symbol stands for a free type definition that has | to separate one element with other elements [34]. Decision has types of disease, where TypeS has types of symptoms. Thus, there are 20 symptoms in our system, G1 to G20. clear means there is no disease caught, as well as nothing means there is no symptom. A, B, C, D, E and F are names of disease modelled by our rule-based Expert System. We define names of disease as types for a global variable, Decision, to be able to use the variable as a type for other variables declared in schemas. The same reason is applied to another global variable, TypeS.

Our state schema is shown in top left of the next page. There are 20 symptoms which each of these is an antecedent of its implication of involved diseases. D1 to D20 are measures of belief or densities. In our expert system, these numbers are in real, but we changed them into integer types that suits the translator. However, we changed them back into real number in the SAL specification. Variables, which begin with dTheta are new densities of several combinations of symptoms.

`decision` is used to store the caught disease. By defining a schema, we can show a system's state and behavior of a computer system [34].

ruleBase

$\mathit{Infected}: \mathit{Decision}$

$\mathit{decision}: \mathbb{P}\mathit{Decision}$

$\mathit{symptom}: \mathit{TypeS}$

$D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12,$
$D13, D14, D15, D16, D17, D18, D19, D20: \mathbb{Z}$

$\mathit{dTheta}1, \mathit{dTheta}2, \mathit{dTheta}31, \mathit{dTheta}32, \mathit{dTheta}33,$
$\mathit{dTheta}34, \mathit{dTheta}4, \mathit{dTheta}51, \mathit{dTheta}51, \mathit{dTheta}52,$
$\mathit{dTheta}53, \mathit{dTheta}54, \mathit{dTheta}55, \mathit{dTheta}56, \mathit{dTheta}57, \mathit{dTheta}58: \mathbb{Z}$

$(\mathit{symptom} = G1 \Rightarrow (\mathit{Infected} = A \lor \mathit{Infected} = C))$
$(\mathit{symptom} = G2 \Rightarrow (\mathit{Infected} = A))$
$(\mathit{symptom} = G3 \Rightarrow (\mathit{Infected} = A))$
$(\mathit{symptom} = G4 \Rightarrow (\mathit{Infected} = A \lor \mathit{Infected} = B))$
$(\mathit{symptom} = G5 \Rightarrow (\mathit{Infected} = C \lor \mathit{Infected} = D \lor \mathit{Infected} = F))$
$(\mathit{symptom} = G6 \Rightarrow (\mathit{Infected} = C \lor \mathit{Infected} = D))$
$(\mathit{symptom} = G7 \Rightarrow (\mathit{Infected} = B \lor \mathit{Infected} = C))$
$(\mathit{symptom} = G8 \Rightarrow (\mathit{Infected} = C))$
$(\mathit{symptom} = G9 \Rightarrow (\mathit{Infected} = B \lor \mathit{Infected} = C))$
$(\mathit{symptom} = G10 \Rightarrow (\mathit{Infected} = E))$
$(\mathit{symptom} = G11 \Rightarrow (\mathit{Infected} = E))$
$(\mathit{symptom} = G12 \Rightarrow (\mathit{Infected} = E))$
$(\mathit{symptom} = G13 \Rightarrow (\mathit{Infected} = E))$
$(\mathit{symptom} = G14 \Rightarrow (\mathit{Infected} = E))$
$(\mathit{symptom} = G15 \Rightarrow (\mathit{Infected} = A))$
$(\mathit{symptom} = G16 \Rightarrow (\mathit{Infected} = F))$
$(\mathit{symptom} = G17 \Rightarrow (\mathit{Infected} = F))$
$(\mathit{symptom} = G18 \Rightarrow (\mathit{Infected} = B))$
$(\mathit{symptom} = G19 \Rightarrow (\mathit{Infected} = B))$
$(\mathit{symptom} = G20 \Rightarrow (\mathit{Infected} = B))$

It is usual to have two parts separated by a line (in a vertical style of schema). The part, which is over the line, is to declare variables. In a case it is a state schema, the variables are state variables, which can be called in other schemas. Other variable is global variable that has been declared in earlier part of our Z specification. The part, which under the line, is to define predicates of a schema. A predicate is an operation that could change variables' values. It can also constrain variables' values [34]. A set will be defined from the satisfied predicates [34].

Therefore, in the predicate part of our state schema, a value is defined for `Infected`. We implemented rules from our expert system as implication statements over all symptoms. Thus, each symptom relates to one or more sicknesses.

The initialization schema is shown below. This schema relates with post-operations of the state schema as implied by declaring the state schema's name with an apostrophe. It means that after the operation of the initialization schema, involved state schema's variables will be changed. This schema defines first values for all state variables.

Initial

$\mathit{ruleBase}'$

$\mathit{Infected}' = \mathit{clear}$

$\mathit{symptom}' = \mathit{nothing}$

$D1' = 9 \land D2' = 6 \land D3' = 6 \land D4' = 6 \land D5' = 9 \land D6' = 7$
$\land D7' = 8 \land D8' = 6 \land D9' = 8 \land D10' = 9 \land D11' = 8$
$\land D12' = 8 \land D13' = 9 \land D14' = 6 \land D15' = 6 \land D16' = 7$
$\land D17' = 6 \land D18' = 9 \land D19' = 9 \land D20' = 6 \land \mathit{dTheta}1' = 0$
$\land \mathit{dTheta}2' = 0 \land \mathit{dTheta}31' = 0 \land \mathit{dTheta}32' = 0 \land \mathit{dTheta}33' = 0$
$\land \mathit{dTheta}34' = 0 \land \mathit{dTheta}4' = 0 \land \mathit{dTheta}51' = 0 \land \mathit{dTheta}52' = 0$
$\land \mathit{dTheta}53' = 0 \land \mathit{dTheta}54' = 0 \land \mathit{dTheta}55' = 0 \land \mathit{dTheta}56' = 0 \land$
$\mathit{dTheta}57' = 0 \land \mathit{dTheta}58' = 0 \land \mathit{density}' = 0 \land \mathit{decision}' = \varnothing$

We do not give our operational schema, `advise`, because this schema is long enough. In this schema, we define predicates representing the calculation of Dempster Shafer method of our expert system. It can be read further in [22].

Our Z specification could accept a minimum of two symptoms and at most of three symptoms to ease the calculation of densities compared to our rules that could have more than three symptoms. The biggest density means that the sickness is recorded on its name.

We have not added medication for each disease in our Z specification due to time limitation. However, we think the medication could be added in the initialization schema.

As mentioned earlier, we have changed the real number into integer number, but changed back to real number in the SAL specification. It is because Z2SAL does not support real number in Z specifications. We change the integer type in the original SAL specification into real type. The original SAL specification before the manual change is as follows:

```
INT : TYPE = [-1..21];
LOCAL D1 : INT
D1' IN {x : REALL | TRUE};
```

We change also multiplication operations into summation with the same reason as above. We change the summation back into multiplication in the accompanied SAL specification.

One example of a summation operation in our Z specification is as follows:

```
dTheta31 = (1 - dTheta1) + (1 - dTheta2)
```

The original SAL part of that line is as follows:

```
dTheta31 = (1 - dTheta1) + (1 - dTheta2)
```

Then we changed it manually into following:

```
dTheta31 = (1 - dTheta1) * (1 - dTheta2)
```

We also modified several lines in the `TRANSITION` section, which represents advice, as these lines do not match with our Z specification. We did all of these changes manually.

Some parts of our SAL specification is presented here. This specification is generated by Z2SAL. However, we have modified some of this SAL specification.

The original SAL specification consists of 580 lines. Thus, we do not put this specification entirely in this paper.

```
homeMed : CONTEXT = BEGIN
REALL : TYPE = [-1..21];
Decision : TYPE = DATATYPE
 …
END;
TypeS : TYPE = DATATYPE
 …
END;
State : MODULE =
 BEGIN
  …
  DEFINITION
   invariant__ = (
…
  INITIALIZATION [
     …
  ]
  TRANSITION [
   advise :
     …
  ]
 END;
 …
END
```

We do not provide SAL parts of `advise`, which is an operation schema of our Z specification. Decision about the sickness will be calculated based on the symptoms. We specify three input variable of symptoms (`symptom1?` to `symptom3?`) in this paper. Each of these symptoms can be one of 20 symptoms and they cannot be the same. For example:

```
(symptom1? = G1 => (symptom2? /= G1 AND symptom3? /= G1 AND dTheta1 = 1 - D1 AND Infected = A AND decisionSet1_' = set {Decision;} ! insert(decisionSet1_', Infected) AND Infected = C AND decisionSet1_' = set {Decision;} ! insert(decisionSet1_', Infected)))
```

Theorems will be put at the bottom of the above SAL specification, between the second last `END` and the last `END`. We have discussed briefly about our theorems in Section IV.

The first theorem is as follows:

Theorem 1. th1: theorem State |- G(NOT(symptom = nothing AND Infected /= clear));

Other four theorems are as follows:

Theorem 2. th2: theorem State |- G(NOT(Infected = C) → (NOT(symptom = G1) OR NOT(symptom = G5) OR NOT(symptom = G6) OR NOT(symptom = G7) OR NOT(symptom = G8) OR NOT(symptom = G9)));

Theorem 3. th3: theorem State |- G(NOT(symptom = G1 AND X(symptom = G1)));

Theorem 4. th4: theorem State |- G(symptom = G1 → Infected = A OR Infected = C);

Theorem 5. th5: theorem State |- G(symptom /= nothing AND Infected = clear);

Theorem 6. th6: theorem State |- G(symptom = G1 AND X(symptom = G1));

They are defined by using LTL (see [35]) with SAL model checker notations [14][29]. `G` means it is applied globally or always the case. `X` means it is happened in the next state.

The first four and the last theorem are proved as VALID. However, the five theorem gives a counterexample because it is INVALID. The counterexample is as follows:

```
Counterexample for 'th5' located at [Context: homeMed, line(578), column(2)]:
========================
Path
========================
Step 0:
--- Input Variables (assignments) ---
symptom1? = G7
symptom2? = G14
symptom3? = G7
--- System Variables (assignments) ---
Infected = clear
decision(A) = false
...
invariant__ = true
```

The fifth theorem does not support the System Variables (assignments). A command to SAL model checker is:

```
$ sal-smc homeMed
```

`sal-smc` is the command to do symbolic model checking, where `homeMed` is the name of the SAL context.

Before we leave this section, we summarize our contribution as follows:

- design a Z specification which represents the rule-based expert system
- manual modification in the SAL specification to suit our Z specification and rule-based expert system
- specify theorems and add it to the SAL specification

VII. Conclusion

We have designed the Z specification for some parts of our expert system. We have also verified our SAL specification by adding six theorems. These theorems were model checked by SAL model checker. The results are as the same as the ones that we get from manual investigation. Thus, we argue that our Z specification along the SAL specification could represent some parts of our expert system. This means that we can model our expert system using specification languages; Z notation and SAL language. This model could show the functionalities of our rule-based expert system. The theorems could also represent properties of our expert system. Based on the model check's results, some parts of our expert system have been verified.

Acknowledgment

I want to thank John Derrick and Anthony Simons at the University of Sheffield for the discussion. I also want to thank Zarina Shukur at Universiti Kebangsaan Malaysia for the recommendation for making this paper more interesting.

References

[1] R. Knauf, A. Gonzalez, and K. P. Jantke, "Validating rule-based systems: a complete methodology," 1999, vol. 5, pp. 744–749 vol.5.

[2] M. Brezovan and C. Badica, "Using the Event-B Formal Method and the Rodin Framework for Verification the Knowledge Base of an Rule-Based Expert System," 2018, pp. 107–127.

[3] A. J. Gonzalez and V. Barr, "Validation and verification of intelligent systems - what are they and how are they different?," J. Exp. Theor. Artif. Intell., vol. 12, no. 4, pp. 407–420, Oct. 2000.

[4] N. Plat, J. van Katwijk, and H. Toetenel, "Application and benefits of formal methods in software development," Softw. Eng. J., vol. 7, no. 5, pp. 335–346, 1992.

[5] S. Stepney and S. P. Lord, "Formal specification of an access control system," Softw. Pract. Exp., vol. 17, no. 9, pp. 575–593, 1987.

[6] D. Jackson, "Abstract model checking of infinite specifications," in FME '94: Industrial Benefit of Formal Methods, 1994, pp. 519–531.

[7] B. Potter, D. Till, and J. Sinclair, An Introduction to Formal Specification and Z, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1996.

[8] M. M. West, "Issues in Validation and Executability of Formal Specifications in the Z Notation," University of Leeds, 2002.

[9] J. Woodcock and J. Davies, Using Z: Specification, Refinement, and Proof. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[10] A. E. Haxthausen, An Introduction to Formal Methods for the Development of Safety-critical Applications. 2010.

[11] C. Matthews, "Fuzzy concepts and formal methods: A sample specification for a fuzzy expert system," in IEEE International Conference on Fuzzy Systems, 2002, vol. 2, pp. 1150–1155.

[12] X. He, H. Yu, and Y. Deng, "Formal Methods for Specifying and Analyzing Complex Software Systems," in Modern Formal Methods and Applications, H. A. Gabbar, Ed. Dordrecht: Springer Netherlands, 2006, pp. 123–150.

[13] J. Derrick, S. North, and A. J. H. Simons, "Z2SAL: a translation-based model checker for Z," Form. Asp. Comput., vol. 23, no. 1, pp. 43–71, Jan. 2011.

[14] L. de Moura, S. Owre, and N. Shankar, "The SAL Language Manual," 2019.

[15] E. Pira, M. R. Z. Miralvand, and F. Soltani, "Verification of confliction and unreachability in rule-based expert systems with model checking," CoRR, vol. abs/1404.2, 2014.

[16] L. Lengyel, "Validating Rule-based Algorithms," Acta Polytech. Hungarica, vol. 12, no. 4, pp. 59–75, 2015.

[17] M. U. Siregar, A pre-processing tool for Z2SAL to broaden support for model checking Z specifications. 2018.

[18] J. C. Giarratano and G. D. Riley, Expert system: principles and programming. 2007.

[19] G. J. Nalepa, "Methodologies and Technologies for Rule-Based Systems Design and Implementation. Towards Hybrid Knowledge Engineering," in Knowledge-Driven Computing: Knowledge Engineering and Intelligent Computations, C. Cotta, S. Reich, R. Schaefer, and A. Lig\keza, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 183–198.

[20] D. Arias-Aranda, J. L. Castro, M. Navarro, J. M. Sánchez, and J. M. Zurita, "A Fuzzy Expert System for Business Management," Expert Syst. Appl., vol. 37, no. 12, pp. 7570–7580, 2010.

[21] L. A. Zadeh, "A Simple View of the Dempster-Shafer Theory of Evidence and Its Implication for the Rule of Combination," AI Mag., vol. 7, no. 2, pp. 85–90, 1986.

[22] M. Beynon, B. Curry, and P. Morgan, "The Dempster-Shafer theory of evidence: An alternative approach to multicriteria decision modeling," Omega, pp. 37–50, 2000.

[23] S. Abriani, "Sistem Rekomendasi Swamedikasi Penyakit Ringan Sistem Pencernaan dengan Metode Dempster Shafer." 2012.

[24] S. Abriani, K. Mukhoyyaroh, and M. U. Siregar, "Recommendation System of Self-Medication for Mild Digestive Diseases with Dempster Shafer Method," IJID, vol. 3, no. 1, 2014.

[25] E. M. Clarke and B.-H. Schlingloff, "Handbook of Automated Reasoning," A. Robinson and A. Voronkov, Eds. Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V., 2001, pp. 1635–1790.

[26] M. Kacprzak, A. Lomuscio, T. Łasica, W. Penczek, and M. Szreter, "Verifying Multi-agent Systems via Unbounded Model Checking," in Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), 2005, vol. 3228, pp. 189–212.

[27] R. Pelanek, "Reduction and Abstraction Techniques for Model Checking," Masaryk University, 2006.

[28] S. Bensalem, Y. Lakhnech, and S. Owre, "Computing Abstractions of Infinite State Systems Compositionally and Automatically," in Computer Aided Verification, 1998, pp. 319–331.

[29] B. Dutertre, Symbolic Analysis Laboratory. 2010.

[30] J. Derrick, S. North, and T. Simons, "Issues in Implementing a Model Checker for Z," in Formal Methods and Software Engineering, 2006, pp. 678–696.

[31] A. J. H. Simons, "Z2SAL: Translation-Based Tools for Z." 2012.

[32] "Safety & Liveness Properties," pp. 1–59.

[33] W. Durand and S. Salva, "Inferring Models with Rule-based Expert Systems," in Proceedings of the Fifth Symposium on Information and Communication Technology, 2014, pp. 92–101.

[34] T. Marris, "Z Notes."

[35] M. Huth and M. Ryan, Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press, 2004.