THE VERIFICATION OF TEMPORAL KNOWLEDGE BASED SYSTEMS
A Case-study on Power-systems

Jorge Santos, Zita Vale, Carlos Ramos
Instituto Superior de Engenharia do Porto, Rua Dr. Antonio Bernardino de Almeida, 431, 4200-072 Porto – Portugal

Carlos Serôdio
Departamento de Engenharias, Universidade de Trás–os–Montes e Alto Douro, 5001-801 Vila Real – Portugal

Keywords: Knowledge–based Systems Engineering, Verification, Temporal Reasoning, Power Systems.

Abstract: Designing KBS for dynamic environments requires the consideration of temporal knowledge reasoning and representation (TRR) issues. Although humans present a natural ability to deal with knowledge about time and events, the codification and use of such knowledge in information systems still pose many problems. Hence, the development of applications strongly based on temporal reasoning remains an hard and complex task. Furthermore, albeit the last significant developments in TRR area, there is still a considerable gap for its successful use in practical applications.

This paper presents a tool, named VERITAS, developed for temporal KBS verification. It relies in the combination of formal methods and heuristics, in order to detect a large number of knowledge anomalies. The underlying verification process addresses many relevant aspects considered in real applications, like the usage of rule triggering selection mechanisms and temporal reasoning.

1 INTRODUCTION

The methodologies proposed in software engineering showed to be inadequate for knowledge based systems (KBS) validation and verification, since KBS present some particular characteristics (Gonzalez and Dankel, 1993).

In last decades knowledge based systems became a common tool in a large number of power systems control centres (CC) (Kirschen and Wollenberg, 1992). In fact, the number, diversity and complexity of KBS increased significantly leading to important changes in KBS structure. Designing KBS for dynamic environments requires the consideration of TRR (Temporal Reasoning and Representation) issues. Although humans present a natural ability to deal with knowledge about time and events, the codification and use of such knowledge in information systems still pose many problems. Hence, the development of applications strongly based on temporal reasoning remains an hard and complex task. Furthermore, in despite of the last significant developments in TRR area, there is still a considerable gap for its successful use in practical applications.

This paper addresses the verification of knowledge based systems through the combination of formal methods and heuristics. The rest of the paper is organized as follows: the section 2 provides a short overview of the state-of-art of V&V and its most important concepts and techniques. Section 3 , describes the the study case, SPARSE, a KBS used to assist the Portuguese Transmission Control Centres operators in incident analysis and power restoration. Section 4 introduces the problem of verifying real world applications. Finally, section 5 presents VERITAS, describing the methods used to detect knowledge anomalies.

2 RELATED WORK

In the last decades several techniques were proposed for validation and verification of knowledge based systems, including for instance, inspection, formal proof, cross–reference verification or empirical tests (Preece, 1998), regarding that the efficiency of these techniques strongly depends on the existence of test cases or in the degree of formalization used on the specifications. One of the most used techniques is static verification, that consists of sets of logical

Santos J., Vale Z., Ramos C. and Serôdio C. (2007).
THE VERIFICATION OF TEMPORAL KNOWLEDGE BASED SYSTEMS - A Case-study on Power-systems.
In *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics*, pages 179-185
DOI: 10.5220/0001651801790185

tests executed in order to detect possible knowledge anomalies. An anomaly is a symptom of one, or more, possible error(s). Notice that an anomaly does not necessarily denotes an error (Preece and Shinghal, 1994).

Some well known V&V tools used different techniques to detect anomalies. The KB–Reducer (Ginsberg, 1987) system represents rules in logical form, then it computes for each hypothesis the corresponding labels, detecting the anomalies during the labeling process. Meaning, that each literal in the rule LHS (Left Hand Side) is replaced by the set of conditions that allows to infer it. This process finishes when all formulas became grounded. The COVER (Preece et al., 1992) works in a similar fashion using the ATMS (Assumption Truth Maintaining System) approach (Kleer, 1986) and graph theory, allowing to detect a large number of anomalies. The COVADIS (Rousset, 1988) successfully explored the relation between input and output sets.

The systems ESC (Cragun and Steudel, 1987), RCP (Suwa et al., 1982) and Check (Nguyen et al., 1987) and later, Prologa (Vanthienen et al., 1997) used decision table methods for verification purposes. This approach proved to be quite interesting, specially when the systems to be verified also used decision tables as representation support. These systems major advantage is that tracing reasoning path becomes quite clear. But in the other hand, there is a lack of solutions for verifying long reasoning inference chains.

Some authors studied the applicability of Petri nets (Pipard, 1989; Nazareth, 1993) to represent the rule base and to detect the knowledge inconsistencies. Colored Petri nets were later used (Wu and Lee, 1997). Although specific knowledge representations provide higher efficiency while used to perform some verification tests, arguably all of them could be successful converted to production rules.

Albeit there is no general agreement on the V&V terminology, the following definitions will be used for the rest of this paper.

Validation means building the right system (Boehm, 1984). The purpose of validation is to assure that a KBS will provide solutions with similar (or higher if possible) confidence level as the one provided by domain experts. Validation is then based on tests, desirably in the real environment and under real circumstances. During these tests, the KBS is considered as a *black box*, meaning that, only the input and the output are really considered important.

Verification means building the system right (Boehm, 1984). The purpose of verification is to assure that a KBS has been correctly designed and implemented and does not contain technical errors. During the verification process the interior of the KBS is examined in order to find any possible errors, this approach is also called *crystal box*.

3 SPARSE – A CASE STUDY

Control Centers (CC) are very important in the operation of electrical networks receiving real–time information about network status. CC operators should take, usually in a short time, the most appropriate actions in order to reach the maximum network performance.

Concerning SPARSE , in the beginning it started to be an expert system (ES) and it was developed for the Control Centers of the Portuguese Transmission Network (REN). The main goals of this ES were to assist Control Center operators in incident analysis allowing a faster power restoration. Later, the system evolved to a more complex architecture (Vale et al., 2002), which is normally referred as a knowledge based system (see Fig.1).

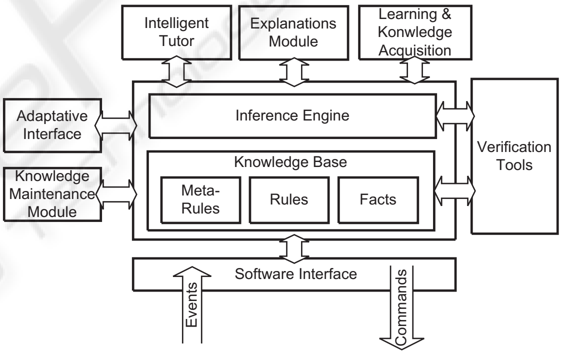

Figure 1: SPARSE Architecture.

One of the most important components of SPARSE is the knowledge base (KB) (see the formula 1):

$$\mathrm{KB} = \mathrm{RB} \cup \mathrm{FB} \cup \mathrm{MRB} \tag{1}$$

where:

- RB stands for rule base;
- FB stands for facts base;
- MRB stands for meta–rules base;

The rule base is a set of Horn clauses with the following structure:

```
RULE ID: 'Description':
[
 [C1 AND C2 AND C3]
 OR
```

```
  [C4 AND C5]
 ]
 ==>
 [A1,A2,A3].
```

The rule's LHS (Left Hand Side) is a set of conditions (C1 to C5 in this example) of the following types:

- A fact, representing domain events or status messages. Typically these facts are time–tagged;
- A temporal condition over facts;
- Previously asserted conclusions

The actions/conclusions to be taken in RHS (Right Hand Side) (A1 to A3 in this example) may be of one of the following types:

- Assertion of facts (conclusions to be inserted in the knowledge base);
- Retraction of facts (conclusions to be deleted from the knowledge base);
- Interaction with the user interface.

The meta–rule base is a set of triggers, used by rule selection mechanism, with the following structure:

$$trigger(Fact, [(R_1, Tb_1, Te_1), \ldots, (R_n, Tb_n, Te_n)])$$

standing for:

- *Fact* – the arriving fact (external alarm or a previously inferred conclusion);
- $(Rule_x, Tb_1, Te_1)$ – the tuple is composed by: $Rule_x$, the rule that should be triggered in when fact arrives; Tb_1 the delay time before rule triggering, used to wait for remaining facts needed to define an event; Te_1 the maximum time for trying to trigger the each rule.

The inference process relies on the cycle. In the first step, SPARSE collects one *message* from SCADA[1], then the respective *trigger* is selected and some rules are scheduled. The temporal window were the rule X could by triggered is defined in the interval $[Tb, Te]$. The *scheduler* selects the next rule to be tested, (the inference engines tries to prove its veracity). Notice that, when a rule succeeds, the conclusions (on the RHS) will be asserted and later processed in the same way as the SCADA messages.

4 THE VERIFICATION PROBLEM

Traditionally the verification problem through anomaly detection relies on the computation of all possible inference chains (expansions) that could be entailed during the reasoning process. Later, some logical tests are performed in order to detect if any constraints violation takes place (see Fig.2 core).

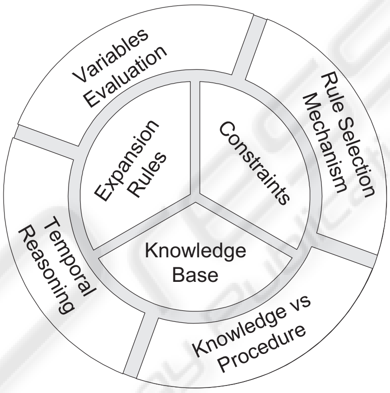

Figure 2: The verification problem.

SPARSE presents some features, namely, temporal reasoning and rule triggering mechanism (see Fig.2), that makes the verification work harder. These features demand the use of more complex techniques during anomaly detection and introduce significant changes in the number and type of anomalies to detect.

4.1 Rule Triggering Selection Mechanism

In what concerns SPARSE, this mechanism was implemented using both meta–rules and the inference engine. When a *message* arrives, some rules are selected and scheduled in order to be later triggered and tested.

In what concerns verification work, this mechanism not only avoids some run–time errors (for instance circular chains) but also introduces another complexity axis to the verification. Thus, this mechanism constrains the existence of inference chains and also the order that they would be generated. For instance, during system execution, the inference engine could be able to assure that shortcuts (specialists rules) would be preferred over generic rules.

[1] Supervisory Control And Data Acquisition, this systems collects messages from the mechanical/electrical devices installed in the network

4.2 Temporal Reasoning

This issue received large attention from scientific community in last two decades (surveys covering this issue can be found in (Gerevini, 1997; Fisher et al., 2005)). Despite the fact that *time* is ubiquitous in the society and the natural ability that human beings show dealing with it, a widespread representation and usage in the artificial intelligence domain remains scarce due to many philosophical and technical obstacles. SPARSE is an *alarm processing application* and its major challenge is to reasoning about events. Thus, it is necessary to deal with time intervals (*e.g.*, temporal windows of validity), points (*e.g.*, instantaneous events occurrence), alarms order, duration and the presence or/and absence of data (*e.g.*, messages lost in the collection/transmission system).

4.3 Variables Evaluation

In order to obtain comprehensive and correct results during the verification process, the evaluation of the variables present in the rules is crucial, especially in what concerns temporal variables, *i.e.*, the ones that represent temporal concepts. Notice that during anomaly detection (this type of verification is also called static verification) it is not possible to predict the exact value that a variable will have. Some techniques, concerning the variables domain and range, were considered in order to avoid the exponencial growth of the number of expansions during its computation.

4.4 Knowledge Versus Procedure

Languages like Prolog provide powerful features for knowledge representation (in the declarative way) but they are also suited to describe procedures, so, sometimes knowledge engineers encode rule using ”procedural” predicates. For instances, the following sentence in Prolog: `min(X,Y,Min)` calls a procedure that compares `X` and `Y` and instantiates `Min` with smaller value. So, is not an (pure) knowledge item, in terms of verification it should be evaluated in order to obtain the `Min` value. It means the verification method needs to consider not only the programming language syntax but also the meaning (semantic) in order to evaluate the functions. This step is particularly important for any variables that are updated during the inference process.

5 VERITAS

VERITAS’s main goal is to detect and report anomalies, allowing the users decide whether reported anomalies reflect knowledge problems or not. Basically, anomaly detection consists in the computation of all possible inference chains that could be produced during KBS performance. Later, the inference chains are tracked in order to find out if some constraint were violated.

After a filtering process, which includes *temporal reasoning analysis* and *variable evaluation* the system decides when to report an anomaly and eventually suggest some repair procedure. VERITAS is knowledge domain and rule grammar independent, due to these properties (at least theoretically), VERITAS could be used to verify any rule–based systems. Let’s consider the following set of rules:

$$\begin{aligned}
&r1 : st1 \wedge ev1 \rightarrow st2 \wedge st3\\
&r2 : st3 \wedge ev3 \rightarrow st6 \wedge ev5\\
&r3 : ev3 \rightarrow ev4\\
&r4 : ev1 \wedge ev2 \rightarrow ev4 \wedge st4\\
&r5 : ev5 \wedge st5 \wedge ev4 \rightarrow st7 \wedge st8
\end{aligned}$$

VERITAS can show the rule dependencies through a directed hypergraph type representation (see Fig.3). This technique allows rule representation in a manner that clearly identifies complex dependencies across compound clauses in the rule base and there is a unique directed hypergraph representation for each set of rules.

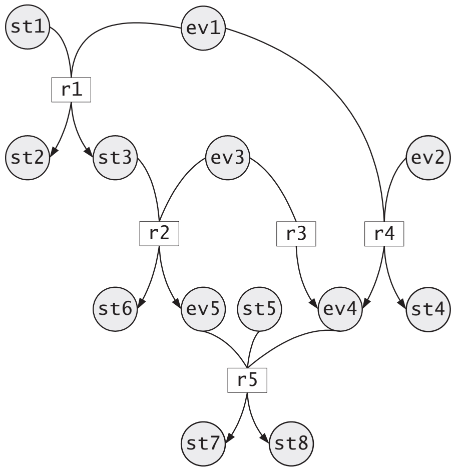

Figure 3: Hypergraph.

After the expansions calculation three different tuples are created, regarding ground facts, conclusions and circular chains, with the following structure, respectively, $f(Fact)$, $e(SL,SR)$ and $c(SL,SR)$, where, Fact stands for a ground fact, SL stands for a set of conclusions and SR stands for a set of supporting rules.

So considering the example previously provided, the following two data sets would be obtained, since there are two possible expansions that allow to infer the labels st7 and st8:

```
f(ev3) f(st5) f(ev3) f(ev1) f(st1)
e([ev4],[[r3]])
e([st2,st3],[[r1]])
e([st6,ev5],[[r1,r2]])
e([st7,st8],[[r3,r5],[r1,r2,r5]])
```

```
f(ev2) f(ev1) f(st5) f(ev3) f(ev1) f(st1)
e([ev4,st4],[[r4]])
e([st2,st3],[[r1]])
e([st6,ev5],[[r1,r2]])
e([st7,st8],[[r4,r5],[r1,r2,r5]])
```

After the expansion computation, VERITAS detects a large set of anomalies. These anomalies can be grouped in three groups, circularity, ambivalence and redundancy, the anomalies classification used is based on Preece classification (Preece and Shinghal, 1994) with some modifications. Next sections describe some of the special cases.

5.1 Circularity

A knowledge base contains circularity if and only if it contains a set of rules, which allows an infinite loop during rule triggering. During the circularity detection, VERITAS considers the matching values in rule analysis, meaning that a new set of anomalies will arise. Let us consider the rules *rc1* and *rc2*:

$$rc1: t(a) \wedge r(X) \rightarrow s(a)$$
$$rc2: s(a) \rightarrow r(a)$$

Since a is a valid value for the argument X, some inference engines could start an infinite loop, so such circular inference chain should be reported. Another situation concerns temporal analysis through the use of heuristics. The rules *rc3* and *rc4* describes the turning on and off of a device D.

$$rc3: st1(D,\mathsf{on},t1) \wedge ev1(D,\mathsf{turnoff}) \rightarrow st2(D,\mathsf{off},t2)$$
$$rc4: st2(D,\mathsf{off},t1) \wedge ev2(D,\mathsf{turnon}) \rightarrow st1(D,\mathsf{on},t2)$$

If nothing else is stated, these two rules could “configure” a circular chain. So, before reporting an anomaly, VERITAS checks if none of the following situations happens, first, if the instant t2 is later than t1. Second, if in each rule LHS’s appears an event (here represented by the ev label).

5.2 Redundancy

A knowledge base is redundant if and only if the set of final hypotheses is the same in the rule/literal presence or absence. A specific situation not considered in Preece anomaly classification concerns to redundancy between groups of rules. Consider the following example:

$$rr1: a \wedge b \wedge c \rightarrow z$$
$$rr2: \neg a \wedge c \rightarrow z$$
$$rr3: \neg b \wedge c \rightarrow z$$

since the rules $rr1$, $rr2$ and $rr3$ are equivalent to the following logical expression:

$$rrx: (a \wedge b \wedge c) \vee (\neg a \wedge c) \vee (\neg b \wedge c) \rightarrow z$$

Applying logical simplifications to the previous rule, it is possible to obtain the following one:

$$rrx: c \rightarrow z$$

These situations are detected with an algorithm for logical expressions simplification. Basically, this algorithm works as follows:

1. for each existing conclusion, compute the set of LHSs of all rules that allows to infer it;
2. try simplify disjunction of the set obtained in the previous step;
3. compare the original expression with the simplified one, if the former is simpler (less variables or less terms) then report a redundancy anomaly;

This algorithm allows multi–valued logic, for instances, consider the following example, regarding that there are just three valid values for the b parameter’s, respectively: open, closed and changing.

$$rr5: a \wedge b(\mathsf{open}) \rightarrow z$$
$$rr6: a \wedge b(\mathsf{closed}) \rightarrow z$$
$$rr7: a \wedge b(\mathsf{changing}) \rightarrow z$$

the rules $rr5$, $rr6$ and $rr7$ are equivalent to the following rule:

$$rry: a \wedge (b(\mathsf{changing}) \vee b(\mathsf{closed}) \vee b(\mathsf{open})) \rightarrow z$$

Applying logical simplifications to the previous rule, it is possible to obtain the following one:

$$rry: a \rightarrow z$$

Notice that, in fact, redundancy between groups of rules is a generalization of the unused literal situation already studied by Preece.

5.3 Ambivalence

A knowledge base is ambivalent if and only if, for a permissible set of conditions, it is possible to infer an impermissible set of hypotheses. For ambivalence detection, VERITAS considers some types of constraints (also referred as *impermissible sets*), representing sets of contradictory conclusions. The constraints can be one following types:

Semantic Constraints – formed by literals that cannot be present at the same time in the KB. For instance, the installation i cant be controlled remotely and locally at same time.

$$\forall i, t1, t2 : \bot \leftarrow Remote(\mathsf{d,On,t1}) \wedge \\ Local(\mathsf{d,Off,t2}) \wedge intersepts(t1, t2)$$

Single Value Constraints – formed by only one literal (Device) but considering different values of its parameters. For instance, the device d cant be on and off at same time.

$$\forall d, t1, t2 : \bot \leftarrow Device(\mathsf{d,On,t1}) \wedge \\ Device(\mathsf{d,Off,t2}) \wedge intersepts(t1, t2)$$

The relation *intersepts* checks wether two intervals have some instant in common. Later, VERITAS computes the various expansions for each item present in a constraint and later determines if there exist a minimal set of facts that allow to infer contradictory conclusions/hypothesis and in such case an anomaly is reported.

6 CONCLUSIONS

This paper described VERITAS, a verification tool that successful combines formal methods, as structural analysis, and heuristics, in order to detect knowledge anomalies and provide useful reports. During its evaluation, the SPARSE was used as study case.

ACKNOWLEDGEMENTS

This work is partially supported by the Portuguese MCT-FCT project EDGAR (POSI/EIA/61307/2004). We would like to thanks the anonymous reviewers for their useful and detailed feedback.

REFERENCES

Boehm, B. (1984). Verifying and validating software requirements and design specifications. *IEEE Software*, 1(1):75–88.

Cragun, B. and Steudel, H. (1987). A decision table based processor for checking completeness and consistency in rule based expert systems. *International Journal of Man Machine Studies (UK)*, 26(5):633–648.

Fisher, M., Gabbay, D., and Vila, L., editors (2005). *Handbook of Temporal Reasoning in Artificial Intelligence*, volume 1 of *Foundations of Artificial Intelligence Series*. Elsevier Science & Technology Books.

Gerevini, A. (1997). Reasoning about time and actions in artificial intelligence: Major issues. In O.Stock, editor, *Spatial and Temporal Reasoning*, pages 43–70. Kluwer Academic Pblishers.

Ginsberg, A. (1987). A new aproach to checking knowledge bases for inconsistency and redundancy. *In procedings 3rd Annual Expert Systems in Government Conference*, pages 10–111.

Gonzalez, A. and Dankel, D. (1993). *The Engineering of Knowledge Based Systems - Theory and Practice*. Prentice Hall International Editions.

Kirschen, D. S. and Wollenberg, B. F. (1992). Intelligent alarm processing in power systems. *Proceedings of the IEEE*, 80(5):663–672.

Kleer, J. (1986). An assumption-based TMS. *Artificial Intelligence Holland*, 2(28):127–162.

Nazareth, D. (1993). Investigating the applicability of petri nets for rule based systems verification. *IEEE Transactions on Knowledge and Data Engineering*, 4(3):402–415.

Nguyen, T., Perkins, W., Laffey, T., and Pecora, D. (1987). Knowledge Based Verification. *AI Magazine*, 2(8):69–75.

Pipard, E. (1989). Detecting inconsistencies and incompleteness in rule bases: the INDE system. In *Proceedigns of the 8th International Workshop on Expert Systems and their Applications, 1989, Avignon,France;*, volume 3, pages 15–33, Nanterre, France. EC2.

Preece, A. (1998). Building the right system right. In *Proc.KAW'98 Eleventh Workshop on Knowledge Acquisition, Modeling and Management*.

Preece, A., Bell, R., and Suen, C. (1992). Verifying knowledge-based systems using the cover tool. *Proccedings of 12th IFIP Congress*, pages 231–237.

Preece, A. and Shinghal, R. (1994). Foundation and application of knowledge base verification. *International Journal of Intelligent Systems*, 9(8):683–702.

Rousset, M. (1988). On the consistency of knowledge bases:the COVADIS system. In *Procedings of the European Conference on Artificial Intelligence (ECAI'88)*, pages 79–84, Munchen.

Suwa, M., Scott, A., and Shortliffe, E. (1982). An aproach to verifying completeness and consistency in a rule based expert system. *AI Magazine (EUA)*, 3(4):16–21.

Vale, Z., Ramos, C., Faria, L., Malheiro, N., Marques, A., and Rosado, C. (2002). Real-time inference for knowledge-based applications in power system control centers. *Journal on Systems Analysis Modelling Simulation (SAMS), Taylor&Francis*, 42:961–973.

Vanthienen, J., Mues, C., and Wets, G. (1997). Inter Tabular Verification in an Interactive Environment. *Proceedings of the European Symposium on the Verification and Validation of Knowledge Based Systems*, pages 155–165.

Wu, C.-H. and Lee, S.-J. (1997). Enhanced High-Level Petri Nets with Multiple Colors for Knowledge Verification/Validation of Rule-Based Expert Systems. *IEEE TRansactions on Systems. Man, and Cybernetics*, 27(5):760–773.