Testing the Integrity of Non-monotonic Knowledge Bases Containing Semi-normal Defaults

Neli P. Zlatareva

Department of Computer Science
Central Connecticut State University
1615 Stanley Street
New Britain, CT 06050
E-mail: zlatareva@ccsu.edu

Abstract

This paper presents a methodology for testing general non-monotonic knowledge bases for logical and semantic inconsistencies. It extends the CTMS-based verification framework introduced in our previous work with an additional ***integrity test***. This test aims to ensure that a logically consistent non-monotonic knowledge base is also free of semantic inconsistencies, resulted from violated integrity constraints. Generally, semantic inconsistencies are detected during dynamic validation of the KBS. We argue, however, that some types of semantic inconsistencies can be explicated during the formal V&V analysis, and we describe a logical procedure that carries out this validation task.

Introduction

In the last 10 years, numerous techniques and tools for verification and validation (V&V) of both traditional rule-based systems, and more complex KBSs utilizing hybrid representations were developed. For an extended summary and references on the former, see the report on EUROVAV'99 (Coenen et al. 2000) and (Wei-Tek, Vishnuvajjala, and Zhang 1999), while examples of the later are (Levy and Rousset 1996) and (Shiu, Liu, and Yeung, 1996). Most of these V&V frameworks, however, assume that a KBS employs a monotonic inference engine and thus the deductive closure of its knowledge base (the subject of the V&V analysis) is uniquely defined. This is not so if a KBS employs non-monotonic inference. The V&V task, in this case, is complicated due to the fact that there may be multiple, sometimes huge, number of extensions resulting from the way the non-monotonic inference works. Testing each extension as a separate entity, as suggested in earlier work (Autoniou 1995), (Zlatareva 1997) is not sufficient to ensure the overall integrity of the knowledge base. We show in this paper that there may be inconsistencies among rules, which are not manifested as structural errors in any given extension, but may lead to incorrect conclusions. Typically, such inconsistencies are caused by undetected errors in underlying domain specifications, and their detection and resolution is part of dynamic validation of the KBS.

The existence of multiple extensions is a core feature of non-monotonic theories. It is expected, however, multiple extensions to be originated by incomplete domain specifications, rather than by inconsistent specifications. To ensure that a knowledge base is free of such inconsistencies, we propose an additional ***integrity test***, which complements the set of verification tests for non-monotonic knowledge bases presented in (Zlatareva 1997). This test identifies semantic inconsistencies among default rules of a very general type by identifying contradictory conclusions supported by the same **monotonic** input data set. We present a formal procedure that carries out this test. The idea is the following. The original non-monotonic theory is converted into an equivalent set of monotonic sub-theories, which are more amenable to formal analysis. Each monotonic sub-theory is independently tested for logical inconsistencies as suggested in (Zlatareva and Preece 1994). After monotonic sub-theories are proved logically consistent, their grounded stable extensions are cross-examined for contradictory conclusions originated by the same input data set. If such contradictory conclusions are found, then rules involved in their derivation are semantically inconsistent. Clearly, semantic inconsistencies of any type can be confirmed and resolved only by a domain expert, but we argue that explicating them as part of the formal V&V process helps improve the efficiency of dynamic validation, which is still considered the "bottleneck" in KBS quality assurance.

The paper is organized as follows. First, we discuss some issues in verification and validation of non-monotonic KBSs that motivated the research presented in this paper. Next, we outline the proposed integrity test, and define the type of knowledge bases that we target. We then introduce a conversion procedure that allows a non-monotonic theory to be converted to a set of semantically equivalent monotonic theories, and show how the proposed integrity test is carried out. An extended example is used throughout the paper to illustrate the presentation.

Issues in V&V of non-monotonic KBS

Verification and validation of non-monotonic theories was sparingly addressed in the V&V literature, although related topics, such as belief revision and knowledge refinement, are widely studies and discussed by AI theoreticians and logicians (see Proceedings of the 8th International Workshop on Non-Monotonic Reasoning for references). This is partly due to the fact that non-monotonic KBSs are still at early development stages, and partly due to the wide spread belief that these systems are incomplete and inconsistent by nature, which is why they cannot be satisfactory validated. On the other hand, the behavior of non-monotonic systems is far more complex than that of monotonic systems. This steams from the fact that non-monotonic inference is flexible enough to continue in the presence of missing data, which considerably reduces the size of the knowledge base, but introduces the well known multiple extension problem. The complexity of the V&V process jumps according to the number of extensions. Each extension now must be tested to ensure its structural correctness and validity, and also the theory as a whole must be tested to ensure that there are no inconsistencies among extensions resulted from violated integrity (semantic) constraints.

Testing extensions for structural anomalies and errors (such as logical inconsistencies, circularities, redundancies, etc.) is not hard because extensions are, in fact, monotonic theories. Any verification technique for monotonic KBSs can be applied to them as well. The validation task, on the other hand, is to ensure that every extension contains all and only correct conclusions. To fully solve this task we need test cases with known solutions. As it is well known, this is a high complexity task even for monotonic KBSs, where we deal with a single extension. Assume, however, that we limit the validation task to only identifying extensions, which possibly contain invalid conclusions. This can be done by cross-examining extensions in search for contradictory formulas implied by the same input data set or its monotonic supersets. If such contradictory formulas are found, then there are semantic inconsistencies among rules involved in their derivation. To identify the "wrong" rule (or rules), we can use test cases already contained in the grounded versions of the extensions involved. Grounded stable extensions are operational theories, which are generated by computing the transitive closure of the formulas comprising stable extensions (deductive closures) of the original theory. In a grounded stable extension, each conclusion is defined in terms of the underlying input data set, which, in turn, defines a test case for that conclusion. To validate extensions, we can compare test cases associated with suspicious conclusions to real-world test cases. This simple idea is easy to implement for monotonic knowledge bases; however, its implementation for non-monotonic knowledge bases turned out to be not trivial. One solution is discussed next.

Definition of the validation task: a case study

Let a knowledge base consists of rules of the following type:

Ri: (premise_1, …, premise_n) (assumption_1,…, assumption_m) → conclusion_i.

Here (premise_1,…,premise_n) is the set of monotonic supporters for conclusion_i, called for short *In-list*, (terminology adapted from (Doyle, 1979)), and (assumption_1,…,assumption_m) is the set of non-monotonic supporters for conclusion_i, called for short *Out-list*. Conclusion_i holds if and only if all of its monotonic supporters hold, and none of its non-monotonic supporters holds. Conclusions may serve as both monotonic and non-monotonic supporters for other conclusions.

Rules, such as Ri, are very general semi-normal defaults. Normal defaults and monotonic rules, respectively, can be expressed in this form as follows:

Rj: (premise_1,..,premise_n) (¬conclusion_i) →
→ conclusion_i

Rk: (premise_1,...,premise_n) ( ) → conclusion_i

Consider the following example theory, which is used throughout the paper to illustrate the proposed technique:

R1: (A1, A2) ( ) → C1
R2: (C1, A3) ( ) → C2
R3: (A4, A5) (A8, A9) → C2
R4: (C2, A6) (C1) → C3
R5: (C3, A7) ( ) → C4
R6: (C4, C2) ( ) → C5
R7: (C5, A8) ( ) → C6

R8: (C6, A9) ( ) → C7
R9: (A7) (C3) → C4

For the purposes of the V&V analysis, we treat all input data (A1 through A9) as assumptions. Conclusions are denoted as C1, C2, etc.

Assumptions may or may not have values. For example, A8 and A9 may be *true*, *false*, or *unknown*. If they are *unknown* (or *false*), R3 will fire if its monotonic supporters hold. In this sense, A8 and A9 are different from A4 and A5, which must be *true* for R3 to fire.

In general, there may be a huge number of specific assignments of values to assumptions, where each assignment set defines a real situation in the problem domain (test case). In verification analysis, we are not interested in any particular assignment; rather, we study interactions between rules assuming that all input data is simultaneously passed to the system. This explicates all potential anomalies (real and false) without running test cases.

When we deal with a non-monotonic theory, a difficulty arises from the existence of *unknown* data (assumptions). These data cannot be given values at the start of the verification process, because this would change the meaning of the rules containing them. This makes it impossible to compute grounded versions of non-monotonic theory extensions, which are the ones examined for structural anomalies. To get around this difficulty, we suggest original non-monotonic theory to be converted into a set of monotonic sub-theories, stable extensions of which are semantically equivalent to extensions of the original theory. What we gain from this conversion is that now we deal with monotonic extensions, grounded versions of which are easy to compute. Moreover, the verification framework introduced in (Zlatareva and Preece 1994) can be applied directly to test these theories for a variety of anomalies and errors, not just logical inconsistencies.

Assuming that each one of these monotonic theories was tested and proved to be structurally correct, the next step towards validating the original non-monotonic theory is to ensure that the union of derivative monotonic theories do not contain contradictory conclusions underlined by the same input data set. If this is the case, then the original non-monotonic theory contains inconsistencies among rules that result from violated integrity constraints, which may lead to functional errors (wrong conclusions reported by the KBS).

In summary, the following three steps outline the V&V process that we have described.

1. Convert the original non-monotonic theory into an equivalent set of monotonic sub-theories, such that stable extensions of monotonic sub-theories are semantically equivalent to possible extensions of the original theory.

2. Test each monotonic extension for logical inconsistencies and other structural anomalies and errors as described in (Zlatareva and Preece 1994).

3. Once the internal consistency of each extension is ensured, cross-examine grounded stable extensions of monotonic sub-theories for contradictory conclusions extending the same input data set, to test for violated integrity constraints.

The rest of this paper deals with steps 1 and 3 of the outlined V&V process. Next, we show how the conversion of a non-monotonic theory into an equivalent set of monotonic sub-theories is carried out.

Conversion of a non-monotonic knowledge base into an equivalent set of monotonic theories

The purpose of converting a non-monotonic theory to a set of monotonic theories is that the later are more amenable to formal V&V analysis than the original theory. The following definition states the conditions under which a non-monotonic theory and a set of monotonic theories are considered **semantically equivalent**.

Definition. A non-monotonic theory Th_{nm} is semantically equivalent to a set of monotonic theories $Th_{m1}, Th_{m2}, \ldots, Th_{mi}$ iff:

a) The number of extensions of Th_{nm} is exactly equal the number of monotonic theories in the set.

b) For every monotonic theory, there exists a non-monotonic theory extension such that the stable extension of the monotonic theory contains all of the formulas from the non-monotonic theory extension, i.e. $EXT_{Thnm} \subseteq EXT_{Thmi}$.

The suggested transformation is carried out in two steps:

1. Each non-monotonic rule is converted into a set of monotonic ones.

2. Consistent subsets of assumptions that originate possible extensions of the non-monotonic theory are identified. The monotonic theory, resulted from the first step, is divided into a number of theories each containing only those rules of the

initial theory that are activated by identified assumption subsets.

Converting non-monotonic rules into monotonic ones turned out to be surprisingly subtle. The semantics of default rules must be carefully accounted for, which is why the following three cases are considered separately.

Case 1. The original rule has an empty *Out-list* and its conclusion does not belong to the *Out-list* of any other conclusion. An example of such rule is R2 (see example theory above). These rules are the easiest to convert, because they are, in fact, monotonic rules. R2 can be recorded as:

R2*: C1 ∧ A3 → C2

Case 2. The original rule has an empty *Out-list*, but its conclusion serves as a non-monotonic supporter for another conclusion. R1 is an example of such rule. Its conclusion, C1, is a non-monotonic supporter for C3 (the conclusion of R4). Consistent with the semantics of default rules, the stronger (monotonic) version of R4 will include ¬C1 as a premise. Therefore, there must be an additional rule (or rules) stating when ¬C1 holds. This results in the following set of monotonic rules that R1 translates to:

R1*: A1 ∧ A2 → C1
R1A*: ¬A1 → ¬C1
R1B*: ¬A2 → ¬C1

Case 3: The original rule has a non-empty *Out-list*. Example of such rule is R3. It will not fire if any of its non-monotonic supporters holds. This translates into following implications:

R3A*: A8 → ¬C2
R3B*: A9 → ¬C2

R3, itself, translates into:

R3*: A4 ∧ A5 ∧ ¬A8 ∧ ¬A9 → C2

Note that explicitly deriving ¬C2 affects all rules having C2 as a monotonic supporter. For each such rule, we must add an implication of the following type:

Rj*: ¬C2 → ¬conclusion_j

Here is the final result of the conversion process for our example theory:

R1*: A1 ∧ A2 → C1
R1A*: ¬A1 → ¬C1
R1B*: ¬A2 → ¬C1
R2*: C1 ∧ A3 → C2
R3*: A4 ∧ A5 ∧ ¬A8 ∧ ¬A9 → C2
R3A*: A8 → ¬C2
R3B*: A9 → ¬C2
R4*: C2 ∧ A6 ∧ ¬C1→ C3
R4A*: C1 → ¬C3
R4B*: ¬C2 → ¬C3
R5*: C3 ∧ A7 → C4
R6*: C4 ∧ C2 → C5
R7*: C5 ∧ A8 → C6
R8*: C6 ∧ A9 → C7
R9*: A7 ∧ ¬C3 → C4
R9A*: C3 → ¬C4

Next step is to identify monotonic sub-theories, whose stable extensions are semantically equivalent to non-monotonic theory extensions. These are comprised by the rules fired by consistent subsets of assumptions originating possible extensions of the non-monotonic theory. To compute these subsets of assumptions, we must:

1. Identify all violated logical constraints in the "extended" input data set, which accounts for all assumptions explicitly accounted for in the resulting monotonic theory.

2. Divide the "extended" input data set into subsets consistent with respect to violated logical constraints.

For our example theory, the extended input data set is the following one:

{A1, ¬A1, A2, ¬A2, A3, A4, A5, A6, A7, A8, ¬A8, A9, ¬A9}

There are four violated logical constraints in this set:

(A1, ¬A1), (A2, ¬A2), (A8, ¬A8), (A9, ¬A9)

Therefore, the number of extensions of the original non-monotonic theory is 2^4. For each one of them a semantically equivalent monotonic theory containing a subset of rules of the converted theory, is defined next. To illustrate how this works, consider the following input subset originating one of the non-monotonic theory extensions:

{A1, A2, A3, A4, A5, A6, A7, A8, ¬A9}

Simple forward chaining on the resulting monotonic theory computes the extension originated by this input set:

{A1, A2, A3, A7, A8, ¬A9, C1, C2, C4, C5, C6}

Note that A4, A5 and A6 can be removed from the input data set without causing any changes in conclusions inferred. Therefore, the resulting monotonic theory is the following one:

R1*: A1 $\wedge$ A2 $\rightarrow$ C1
R2*: C1 $\wedge$ A3 $\rightarrow$ C2
R3A*: A8 $\rightarrow$ ¬C2
R4A*: C1 $\rightarrow$ ¬C3
R4B*: ¬C2 $\rightarrow$ ¬C3
R6*: C4 $\wedge$ C2 $\rightarrow$ C5
R7*: C5 $\wedge$ A8 $\rightarrow$ C6
R9*: A7 $\wedge$ ¬C3 $\rightarrow$ C4

Testing this theory for anomalies and errors involves computing its stable extension (deductive closure). For this, we use the CTMS-based verification technique described in (Zlatareva and Preece 1994). Stable extensions contain all formulas that follow from a given input data set. CTMS-based verification technique identifies all potential inconsistencies among formulas in a given extension; however, more processing is needed to decide whether an inconsistency is "real" or "false". Tracing backwards from conclusions to the input data sets results in the so-called grounded stable extension (GSE), where each formula is described in terms of assumptions and rules contributing to its derivation.

The following is the GSE of the example monotonic theory:

GSE = {C1: (A1, A2, R1*)( ),
C2: (A1, A2, A3, R1*, R2*)( ), ¬C2: (A8, R3A*) ( ),
¬C3: (A1, A2, R4A*) ( ), ¬C3: (A8, R4B*) ( ),
C4: (A1, A2, A7, R9*, R4A*),
C4: (A7, A8, R9*, R4B*),
C5: (A1, A2, A3, A7, R9*, R4A*, R6*, R1*, R2*),
C5: (A1,A2,A3,A7,A8, R9*, R4B*, R6*, R1*, R2*),
C6: (A1,A2,A3,A7, R8,R9*, R4A*, R6*, R1*, R2*),
C6: (A1,A2,A3,A7,A8, R9*, R4B*, R6*, R1*, R2*)}

Note that detected contradiction (C2, ¬C2) signals an error only if there exists a real world test case involving A1, A2, A3, and A8. By analyzing the support information for contradictory formulas in GSEs, we can identify real logical and semantic inconsistencies in the underlying theory as described next.

Testing GSEs for semantic inconsistencies

To ensure that the original non-monotonic theory is free of inconsistencies (both, logical and semantic), we must:

1. Show that each GSE of the resulting monotonic theories is consistent.
2. Cross-examine GSEs for contradictory conclusions originated by the same input. If such contradictory conclusions are detected, then there are inconsistencies in the original theory, resulted from violated integrity constraints.

We show how these tests are carried out on examples below.

The GSE originated by input set {A1, A2, A3, A4, A5, A6, A7, A8, A9} includes the following contradictory formulas:

C2: (A1, A2, A3, R1*, R2*)( ),
¬C2: (A8, R3A*) ( ),
¬C2: (A9, R3B*) ( ).

Although the GSE contains both, C2 and ¬C2, this is only a potential contradiction, which may occur if A1, A2, A3, and either A8 or A9 are members of some real-world test case. That is, if contradictory conclusions are originated by disjoint input data sets, then real-world test cases are needed to decide whether the detected contradiction may occur in a real domain setting.

The GSE originated by input set {A1, ¬A2, A3, A4, A5, A6, A7, ¬A8, ¬A9} contains the following contradictory formulas:

C4: (¬A2, A4, A5, A6, A7, ¬A8, ¬A9, R1B*, R3*, R4*, R5*),
¬C4: (¬A2, A4, A5, A6, ¬A8, ¬A9, R1B*, R3*, R4*, R9A*).

Here the data set of ¬C4 is a sub-set of the data set of C4. Therefore, we have a real inconsistency that will occur in any super-set of {¬A2, A4, A5, A6, ¬A8, ¬A9}. It is easy to see where this inconsistency is coming from. GSEs maintain records of both assumptions and rules involved in each derivation. Comparing the rule sets of the contradictory formulas allows us to identify inconsistent rules. In our example, the rules-culprits are R5* and R9A*, which are derivatives of R5 and R9. The former rule suggests that conclusion C4 holds if A7 holds and C3 holds, while the later rule suggests that C4 holds if A7 holds, but C3 does not hold. Clearly, there is a semantic inconsistency between these two rules. Resolution of such inconsistencies is a prerogative of a human expert (contrary to logical inconsistencies, which are amenable to automated refinement).

Consider R2 and R4 from the original theory, and compare them to R5 and R9. It may seem that R2 and R4 are inconsistent for a similar reason: C2 holds if C1 holds, while C3 holds if C2 holds and C1 does not hold. This inconsistency only seems to appear possible, but there is no GSE where both C2 and C3 (a corresponding semantic constraint must be defined as part of the domain specification) are originated by the same input set.

Conclusion

In this paper, we have discussed a methodology for testing general non-monotonic knowledge bases for logical and semantic inconsistencies. The CTMS-based verification framework presented in (Zlatareva 1997) was extended with an additional ***integrity test***, which aims to ensure that a logically consistent knowledge base is also free of semantic inconsistencies. Clearly, semantic inconsistencies can be confirmed and resolved only by a domain expert during KBS dynamic validation. We have argued, however, that explicating semantic problems that resulted from violated integrity constraints, can be done as part of the formal V&V analysis, and we have presented a procedure to carry out this task. We have tested the proposed procedure on a small knowledge base (about 100 rules implemented in LISP), and we believe that it can be of practical interest for real-world applications. More work remains to be done to see to what extend traditional dynamic validation can be substituted by formal validation procedures such as the one presented here.

Acknowledgement

This research was supported by a CCSU-AAUP research grant.

References

Antoniou, G. 1995. Integrity and rule checking in non-monotonic knowledge bases. In Working Notes of IJCAI'95 Verification and Validation of Knowledge-Based Systems Workshop, Montreal, Canada.

Coenen, F.P., Bench-Capon, T., Boswell, R., Dibie-Barthelemy, J., Eaglestone, B., Gerrits, R., Gregoire, E., Ligeza, A. Laita, L., Owoc, M., Sellini, F., Spreeuwenberg, S., Vanthienen, J., Vermesan, A. and Wiratunga, N. 2000. Validation and Verification of Knowledge-based systems: report on EUROVAV99. *Knowledge Engineering Review*, Vol. 5, No 2.

Doyle, J. 1979. Truth Maintenance Systems. *Artificial Intelligence,* Vol. 12, North Holland.

Levy, A. and Rousset M.-Ch. 1996. Verification of Knowledge Bases based on Containment Checking. In Proceedings of ECAI'96 Validation, Verification and Refinement of KBS Workshop, Budapest, Hungary.

Proceedings of the 8th International Workshop on Non-Monotonic Reasoning, 2000. Breckenridge, Colorado, April 9 – 11.

Shiu, S., J. Liu and Yeung D. 1996 An Approach towards the Verification of Hybrid Rule/Frame-based Expert Systems using Coloured Petri Nets. In Proceedings of ECAI'96 Validation, Verification and Refinement of KBS Workshop, Budapest, Hungary.

Wei-Tek T., Vishnuvajjala R., and Zhang D. 1999. Verification and Validation of Knowledge-Based Systems. *IEEE Trans. of Knowledge and Data Engineering*, Vol. 11, No. 1.

Zlatareva, N., and Preece A. 1994. An Effective Logical Framework for Knowledge-Based Systems Verification. *International Journal of Expert Systems: Research and Applications*, Vol. 7, No 3.

Zlatareva, N. 1997. Verification of non-monotonic knowledge bases. *Decision Support Systems*, Vol. 21, Elsevier Publ.