Knowledge-Base Reduction: A New Approach to Checking Knowledge Bases for Inconsistency & Redundancy

Allen Ginsberg
Knowledge Systems Research Department
AT&T Bell Laboratories
Holmdel, NJ 07733

Abstract

This paper presents a new approach, called *knowledge-base reduction*, to the problem of checking knowledge bases for inconsistency and redundancy. The algorithm presented here makes use of concepts and techniques that have recently been advocated by de Kleer [deKleer, 1986] in conjunction with an assumption-based truth maintenance system. Knowledge-base reduction is more comprehensive than previous approaches to this problem in that it can in principle detect *all* potential contradictions and redundancies that exist in knowledge bases (having expressive power equivalent to propositional logic). While any approach that makes such a guarantee must be computationally intractable in the worst case, experience with *KB-Reducer* - a system that implements a specialized version of knowledge-base reduction and is described in this paper - has demonstrated that this technique is feasible and effective for fairly complex "real world" knowledge bases. Although KB-Reducer is currently intended for use by expert system developers, it is also a first step in the direction of providing safe "local end-user modifiability" for distant "sites" in a nationwide network of expert systems.

1 Introduction

This paper presents a new technique, called *knowledge-base reduction* (KB-reduction) that can be used, among other things [Ginsberg, 1988b], for checking knowledge bases (rule sets) for inconsistency and redundancy. KB-reduction is based upon recent advances in thinking about problems of truth maintenance for problem-solving systems due mainly to de Kleer [deKleer, 1986], and is also related to the notion of "operationalization" found in the literature on explanation-based learning [Mitchell *et al.*, 1986]. While each of these procedures is dynamic - de Kleer's ATMS, for example, processes justifications for conclusions only when the problem solver passes it the results of some step in reasoning - KB-reduction involves a complete prior analysis of the knowledge base. If one views an expert system as implicitly specifying a (partial) function, having all possible input sets as domain and all possible sets of conclusions as range, then KB-Reduction may be seen as a transformation of this implicit function to a function $\mathcal{E}$ more amenable to analysis: for each conclusion c, $\mathcal{E}(c)$ is a *minimal sum-of-products* (or minimal disjunctive normal form) expression in which each product term (disjunct) consists solely of symbols representing possible input data; following the terminology of de Kleer, each product term is said to be an *environment* for c, and $\mathcal{E}(c)$ is said to be the *label* for c. Intuitively, $\mathcal{E}(c)$ represents all the possible minimal input sets which cause the knowledge base (KB) to assert c. By scrutinizing the intermediate results produced in the process of constructing $\mathcal{E}$, one can, in principle, detect *all* potential inconsistencies and redundancies in the knowledge base. This result is an advance over previous approaches [Nguyen *et al.*, 1985; Suwa, *et al.*, 1982] which focused on analyses involving *pairs* of rules with conflicting or identical conclusions; such pairwise analyses cannot guarantee detection of all inconsistencies and redundancies in a KB [Ginsberg, 1987].

KB-Reducer is a system that implements a special version of the knowledge base reduction technique. KB-Reducer has been shown to be an effective tool on "real world" knowledge bases of various sizes and degrees of complexity. This paper focuses on the version of KB-Reduction implemented in KB-Reducer.

1.1 Motivation: Safe Local End-User Programmability

Imagine that a group of expert system developers - with the assistance of a single expert - are designing a system, distinct copies of which may eventually be *used* at more than 100 hundred sites across the nation. Each site is a local end-user environment that is likely to differ in significant ways from the others. Moreover, imagine that each site possesses one or more "local" experts who want the ability to modify their local copy of the system so that it will do things in ways they are accustomed to, or that is more appropriate for the local environment. This is a scenario that is on its way to becoming a reality in the AT&T telecommunications network [Callahan, 1988; Khan and Dube, 1987].

Although KB-Reducer is currently intended as a tool for expert system developers, the work presented here represents part of the foundation for providing safe local end-user programmability. A modification that introduces a potential inconsistency into a knowledge base is dangerous and should be regarded as being unsafe (until proven otherwise) even if its immediate effect is to produce some desired change in behavior. A modification that introduces a redundancy into the knowledge base cannot, by definition (see section 3), achieve any change in the system's behavior and should be flagged in order to help avoid user frustration. Moreover, redundancies that go undetected may lead to future problems when attempting to effect desired changes in the behavior of the expert system.

2 KB-Reducer's Model of Inference

One cannot talk about consistency or irredundancy of a knowledge base in precise terms without characterizing the deductive apparatus used by the expert system. KB-Reducer is currently based upon an abstract model of inference that emphasizes the close relationship between certain forms of expert system reasoning and natural deduction in propositional logic.

The three most important features of KB-Reducer's current inference model are the following. First, *monotonicity*: if a certain set of input data, α causes the inference engine to conclude C, then any *superset* of α leads to C being asserted; moreover, there is no possibility of "retracting" conclusions or data from working memory. Secondly, the inference engine is *non-selective*: there is no conflict-resolution strategy [Forgy and McDermott, 1977], every satisfied rule is "fired" exactly once and its conclusions are deposited into working memory. Finally, the inference engine may be said *strongly data-driven* in the following sense. It is assumed that all the input data for any case in the problem domain is deposited into working memory in advance of any rule evaluation. Initially all rules whose left hand sides "match" against the input data are fired. Rules which are satisfied by matching against the input data together with the other contents of working memory are now fired. This process is repeated until no further rules can be fired.

Additional assumptions underlying KB-Reducer, and certain subtleties concerning KB-Reducer's inference model will be discussed as the need arises. There are two important points to make here. The first is that experience has shown that KB-Reducer can provide useful analyses of knowledge bases that are designed for inference engines that do not entirely conform to KB-Reducer's model. Secondly, partly as a result of this experience, it is now clear that the essential idea behind knowledge-base reduction - transformation of a knowledge-base into a form more amenable to analysis - can in principle be applied to systems that violate any or all of the assumptions made by the current version of KB-Reducer. Elaboration of some of these claims may be found in [Ginsberg, 1988a; Ginsberg, 1988b]

3 Inconsistency and Redundancy Defined

While KB-Reducer's inference model is intended to reflect a preference for a formal view of inference, it is important to point out a difference in the notion of consistency as employed in formal logic and as employed in the rule-based systems paradigm. In terms of formal logic, we may say that a set of propositions $\mathcal{P}$ is consistent iff there is *some* way of interpreting the propositional symbols of $\mathcal{P}$ so that a contradiction is not entailed. Thus in terms of formal logic the set of propositions

$$p \rightarrow Q$$
$$p\&r \rightarrow \neg Q$$

is consistent: if p is assigned false, for example, both propositions are true and no contradiction is entailed. (Note that throughout this paper propositional symbols in lower case represent "input" variables, while symbols in upper case represent possible conclusions.)

Clearly, however, a knowledge base containing the preceding propositions as rules is inconsistent: if p and r were to be given as an input, the knowledge base would be ready to assert both Q and $\neg Q$. Consistency of knowledge bases is a more stringent requirement than consistency from the formal logic point of view. Roughly speaking, a knowledge base is consistent iff there is *no* way of reaching contradictory assertions from "valid input data". The notion of "valid input data" will be discussed below.

The notion of redundancy of a knowledge base, as employed here, is nearly identical to the notion of *independence* in formal logic. A set of propositions $\mathcal{P}$ is said to be *independent* if no $p \in \mathcal{P}$ follows from the other members of $\mathcal{P}$. A set of rules $\mathcal{R}$ is said to be *irredundant* if no $r \in \mathcal{R}$ follows from the other members of $\mathcal{R}$, and if no $r \in \mathcal{R}$ is such that r can never be satisfied by any valid input set. Thus, in the example above, if the set $\{p, r\}$ turns out to be an invalid input combination, the second rule will be redundant (although the knowledge base would not be inconsistent).

3.1 Valid Input Sets and Semantic Constraints

We say that an input set $\mathbf{E}$ to a KB is *valid* iff $\mathbf{E}$ does not violate any of the *semantic constraints* that exist for the domain in question. For example, the propositions *John is a male human being*, and *John is pregnant* represent invalid input since a male cannot be pregnant, i.e., it violates customary usage of these terms to assert that one and the same person is both male and pregnant. An important example of a general type of semantic constraint is *single-valuedness* of object attributes, e.g., a particular person has one and only weight (at a given time). Domain-independent constraints, such as logical constraints, e.g., a person cannot both be a male and a non-male, and mathematical constraints, e.g., an object cannot both weigh more than 50 ounces and less than 20 ounces, must also be satisfied for an input set to be valid.[1]

While logical and mathematical constraints must always hold, and single-valuedness constraints may generally be assumed by default, the domain-specific semantic constraints known to a knowledge-base developer will usually grow as a function of time. Thus, some invalid input sets may initially appear to be valid since the appropriate semantic constraints are unknown. "Inconsistencies" discovered in the knowledge base at this time, may latter vanish as a result of additional semantic constraints being posted. Indeed, the discovery of interesting semantic constraints may be a direct consequence of the discovery of such "inconsistencies." We therefore say that a KB is *potentially inconsistent* for some agent γ iff there is at least one set of inputs $\mathbf{E}$ that does not violate any semantic constraints *known to* γ and which is such that the KB will assert conflicting conclusions if $\mathbf{E}$ is given as input. To be

[1] The issue of how to design rule-based systems that can recognize invalid input data and take appropriate action, while an important one, is not directly related to the issue of the internal consistency of knowledge bases.

KB

$p \vee q \rightarrow R$
$a \rightarrow B$
$B\&R \rightarrow S$
$\neg S\&c \rightarrow \neg R$

Findings: p, q, a, c
Hypotheses: R, $\neg R$, B, S, T
Default-Hypotheses: $\neg S$
Label for R: $p \vee q$
Label for B: a
Label for S: $a\&p \vee a\&q$
Label for ¬ S: $\neg a \vee (\neg p\&\neg q)$
Label for ¬ R: $\neg a\&c \vee (\neg p\&\neg q\&c)$

Figure 1: Findings, Hypotheses, Default-Hypotheses, and Labels

absolutely certain that a knowledge base will never be in a position to assert contradictory conclusions (or use them in its reasoning) one must be able to reject all potential inconsistencies on the basis of semantic constraints.

4 KB-Reducer

KB-Reducer analyzes KB's written in a canonical rule representation language that is based on literals having an object-attribute-value type of syntax. For example (Person Gender Male) and (NOT (Person Weight > 200)) are possible rule literals. A rule consists of a left hand side in conjunctive normal form (*or's* of *and's*), and the right hand side is a conjunction of literals. In the current implementation KB-Reducer uses a purely syntactic criterion for identifying inputs versus hypotheses versus default-hypotheses. A *finding* [Weiss and Kulikowski, 1979] (input) is any literal that appears only on the left hand side of rules *and* is not the logical negation of any literal on the right hand side of any rule. (The second proviso distinguishes findings from default-hypothesis.) A *hypothesis* is any literal that either occurs solely on the right hand side of rules or on the right hand side of some rules and the left hand side of others. A default-hypothesis is any literal that only occurs on the left hand sides of rules and is the logical negation of some hypothesis. If D is a default-hypothesis we refer to the hypothesis that it negates as its *counter-hypothesis*. See figure 1 for an example.

Since default-hypotheses, by definition, are not asserted by any rules in a KB, an environment **E** for a default-hypothesis D is interpreted to be a set of findings which will prevent the KB from concluding the counter-hypothesis of D. If the label, $\mathcal{L}$, for the counter-hypothesis of D is known, the label for D itself may be computed by negating $\mathcal{L}$ and converting the resulting expression to disjunctive normal form (being sure to minimize the resulting expression). Figure 1 illustrates these concepts.

This manner of calculating the labels of default-hypotheses may seem troubling in view of the following consideration. Suppose, referring to figure 1, that c is the sole input to this KB. Even though no environment in the label for $\neg S$ is satisfied by this input, one might argue that $\neg S$ should be assumed to be true - note that no environment in the label for S is satisfied - and therefore $\neg R$ should be concluded. *A fortiori*, the input $\{p, c\}$ should lead to both R and $\neg R$ being asserted. This argument is cogent, but it is wrong to view it as an objection to our technique for handling default-hypotheses. The reason is that a sort of "closed-world assumption" [Reiter, 1980] with respect to the input data is clearly at work in this argument. The tacit assumption is being made - at least for certain findings - that if they are not given in the input then their negations may be assumed. Thus in the example just stated, since a is not given in the input set one tends to assume that $\neg a$ is true, the later being an environment for $\neg S$. This is a particularly reasonable assumption to make with respect to a since the literal $\neg a$ does not explicitly occur in the knowledge base in figure 1, i.e., presumably the literal $\neg a$ is not expected to ever occur as an explicit input. KB-Reducer can be directed to make the "closed world" assumption in cases such as this, and the results of its analyses will reflect this point of view.

4.1 Ordering The Knowledge Base

KB-reducer currently requires that the rules in the knowledge base form an acyclic network in a sense to be defined below. The relation that is used to define this network is called *the depends-on* relation. Speaking somewhat loosely, a rule r depends-on a rule r' iff r' asserts a literal l, such that either l or its negation appears in the left hand side of r [Ginsberg, 1987]. If for any rule r, the pair $< r, r >$ is in the *transitive closure* of the depends-on relation, the KB has a cycle, otherwise the KB is acyclic.

Assuming that the KB is acyclic, it is possible to partition its rules into *levels* which are determined according to the following recursive definition:

Level of r = 0, if all the literals on left hand side of r are findings
otherwise, = 1 + max of level of rules that r depends on.

4.2 Partial Labels and Rule Labels

If a KB is acyclic, in the sense defined above, it can always be reduced in one pass over the rules by first processing all level 0 rules, then all level 1 rules, etc., working up to the highest level. Only after all the rules have been processed will the labels of all the hypotheses and default-hypotheses be known. The incomplete "labels" generated as rules are processed one-by-one, will be called *partial labels*. As a rule is processed the current partial label for every hypothesis that it asserts is updated, in a manner to be described below, and, in addition, checks are done for redundancy and contradiction. Partial labels for default-hypotheses are updated when rules on whose left hand sides they occur are processed. Note that if a KB is acyclic, then by processing rules in this order we will never process a rule containing a default-hypothesis D on its left hand side until every rule asserting D's counter-hypothesis has been processed.

Partial labels are updated as follows. Suppose that we are currently processing rule r, and that r asserts hypothesis H. We first compute the complete set of minimal environments that lead to satisfaction of r's left hand side by taking the logical conjunction of the labels of the literals on its left hand side and minimizing the resulting expression.

We call the set of minimal environments so generated the *rule-label* for rule r. The new partial label for H is then determined by taking the logical disjunction of r's rule-label and the current partial label for H, and minimizing the resulting expression. It is provable that, if the computation is done in the order described, then the rule-label for a rule will indeed be computable *at the time it is "added" to the KB*, i.e., the rule-label for a rule need never be updated as the computation proceeds to other rules. This is due to the fact that rules at higher levels cannot effect the satisfaction conditions of rules at lower levels. Similar remarks apply to the computation of labels for default-hypotheses. Detailed discussion of the procedures for label computation are described in [Ginsberg and Rose, 1987] and discussion of these issues may also be found in [deKleer, 1986].

4.3 Checking for Redundancy

Suppose that we have just computed the rule-label for rule r which asserts hypothesis H. At this time we will check for redundancy by determining whether i) the rule-label of r consists solely of inconsistent (or invalid) environments, ii) the rule-label of r is implied by (more general than) the current partial label for H, i.e., every environment in H is a superset of some environment in the rule-label of r or iii) the rule-label of r implies the current partial label for H, i.e., every environment in the rule-label of r is a superset of some environment in H.

In the first case the KB is redundant because r can never be satisfied. In case (ii), since rule r concludes H in every case concluded by one or more previously processed rules, one or more of the latter may be removed from the KB. In the last case the reverse is true, i.e., rule r may be removed from the KB since every case in which it is satisfied is already covered by one or more previously processed rules.

4.4 Checking for Contradictions

Suppose that we have just computed the rule-label for rule r which asserts hypothesis H (and have completed the redundancy check described above). We now update the partial label for H using r's rule-label. Except for hypotheses that are explicit negations of each other, KB-Reducer currently requires a list of conflicting hypotheses to be given in advance by the knowledge engineer or domain expert. For every hypothesis X that conflicts with H we do the following *subset/superset* and *union* tests. The subset/superset test determines whether there is any environment in the partial label for H that is a subset or superset of any environment in the partial label for X. Any such an environment represents a set of inputs that will lead the KB to assert both X and H; the environment is flagged and note is made of the rule that was processed when it was discovered. Let E_1 and E_2 be environments from the partial labels for H and X respectively such that neither environment is a subset of the other. For each such pair, the union test determines whether the "combined environment" $E_1 \cup E_2$ violates some domain-independent or domain-specific semantic constraint. If $E_1 \cup E_2$ does *not* violate one of these constraints then it represents a combination of inputs that may cause the KB to assert contradictory hypotheses.

It can happen that an environment flagged by the subset/superset test may violate some semantic constraint. While this means that the danger of inconsistency is avoided, it also means that a flaw in the knowledge base exists, such as a rule that can never be satisfied, or more likely, an unsatisfiable disjunct in a rule component. Potential contradictions that are flagged by the union check, however, do not carry this implication: such a "combined environment" indicates a need for revision of the KB only if it does not violate any semantic constraints.

Figure 2 provides illustrations of the concepts and algorithm described above. The level and rule-label of each rule is displayed, as well as the partial-labels that would exist for various hypotheses and default-hypotheses just after processing the rule in the corresponding row. Points at which potential contradictions and redundancies would be flagged are indicated by asterisks.

5 Discussion

5.1 Experimental Results

KB-reducer has been used to analyze several knowledge bases. Running on an Explorer II[2], knowledge bases of approximately 50, 150, and 370 rules in size were reduced in 40 cpu seconds, 5 cpu minutes, and 10 cpu hours, respectively. The total numbers of environments produced was approximately 700, 4000, and 35,000, respectively. In several cases it found redundant rules and contradictions - of the "combined environment" variety - that the knowledge base developers had missed. It should be noted that once a KB has been reduced, it is generally not necessary to reprocess the entire KB in order to determine the effect of making certain modifications to it.

5.2 Complexity Considerations

The complexity of ordering the knowledge base is proportional to the number of rules in the knowledge base. The complexity of the reduction step is proportional to the number of environments that are generated. In the worst case, a knowledge base may generate 3^n environments, where n is the number of findings (for any finding f, either f, $\neg f$, or neither may be contained in an environment). However, this will occur only when every distinct combination of findings is included in the label for some hypothesis. Recalling that a label, by definition, contains only *minimal*, i.e., non-subsumable environments, even for a relatively small number of findings, say 20, it is hard to imagine that any human being could formulate, comprehend, or use, a knowledge base that postulates relevant distinctions among over one million data combinations. Such a "theory" would simply lack the requisite degree of compactness and generality to ever be learned by anyone in the first place. [3] Good estimates of the number of environments generated by knowledge bases must be generated on a case-by-case basis, and can be gotten as a "side-effect" of the ordering phase of the reduction procedure [Ginsberg, 1987; Ginsberg, 1988a].

[2] Explorer is a trademark of Texas Instruments Incorporated.

[3] de Kleer gives a similar argument for the ATMS [deKleer, 1986, p. 153].

Rule	Level	Rule-Label	Partial-Labels (After Processing Rule)
$p \vee q \rightarrow A$	0	$p \vee q$	**A**: $p \vee q$
$q \vee r \rightarrow B$	0	$q \vee r$	**A**: $p \vee q$, **B**: $q \vee r$
$s\&\neg q \rightarrow \neg B$	0	$s\&\neg q$	**A**: $p \vee q$, **B**: $q \vee r$, $\neg$ **B**: $s\&\neg q$*
$A \rightarrow D$	1	$p \vee q$	**A**: $p \vee q$, **B**: $q \vee r$, $\neg$ **B**: $s\&\neg q$, **D**: $p \vee q$
$B \rightarrow \neg D$	1	$q \vee r$	**A**: $p \vee q$, **B**: $q \vee r$, $\neg$ **B**: $s\&\neg q$, **D**: $p \vee q$, $\neg$ **D**: $q \vee r$**
$D\&\neg A \rightarrow T$	2	$(p\&\neg p\&\neg q) \vee (q\&\neg p\&\neg q)$***	**A**: $p \vee q$, $\neg$ **A**: $\neg p\&\neg q$, **B**: $q \vee r$, $\neg$ **B**: $s\&\neg q$, **D**: $p \vee q$, $\neg$ **D**: $q \vee r$

* Flag $s\&\neg q\&r$ as leading to potential contradiction: **B** and $\neg$ **B**
** Flag q as leading to potential contradiction: **D** and $\neg$ **D**
*** Flag $D\&\neg A \rightarrow T$ as unsatisfiable (redundant) rule

Figure 2: Detecting Inconsistency and Unsatisfiable Rules

5.3 Future Directions

Knowledge-base reduction is also a first step in a threefold procedure for providing expert systems with the capability to automatically reliably modify their knowledge bases in order to adapt to varying local environments. Initial results for this approach to supervised learning are encouraging [Ginsberg, 1988b]. A version of KB-Reducer that will allow for cyclic KB's and inference models that are selective and not strongly data-driven (see section 2) - for example, the same literal may be used as both an input and a hypothesis - is the subject of current investigation. Finally, while no general algorithm can exist that solves the problems of inconsistency and redundancy for knowledge-based systems having expressive power equivalent to first-order logic, we hope to develop an efficient implementation of KB-reduction that will handle occurrences of individual variables in rules and other features of predicate logic not currently accounted for in KB-Reducer.

6 Acknowledgments

I am grateful to Paul Callahan and Rajesh Dube for their interest and support of this research. Lincoln Rose's contribution in implementing the first version of KB-Reducer is gratefully acknowledged, as is Richard Reed's contribution to the current technology transfer effort. I thank David Etherington for useful comments on an earlier version of this paper. Finally, I thank Tom London, Bill Ninke, and Arno Penzias for their recognition and support of this work.

References

[Callahan, 1988] P. Callahan. Expert Systems for AT&T Switched Network Maintenance *AT&T Technical Journal*, 67:93–103,1988.

[deKleer, 1986] Johan de Kleer. An assumption-based tms. *Artificial Intelligence*, 28:127–162, 1986.

[Forgy and McDermott, 1977] C. Forgy and J. McDermott. OPS, a domain-independent production system language. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 933–939, 1977.

[Ginsberg, 1987] A. Ginsberg. A new approach to checking knowledge bases for inconsistency and redundancy. In *Proceedings of The Third Annual Expert Systems in Government Conference*, Washington, D.C., 1987.

[Ginsberg, 1988a] A. Ginsberg. Theory reduction: operationalization as a prelude to learning. In *Proceedings of AAAI Spring Symposium Series*, Stanford U., 1988.

[Ginsberg, 1988b] A. Ginsberg. *Theory Revision via Prior Operationalization.* In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 1988.

[Ginsberg and Rose, 1987] A. Ginsberg and L. Rose. *KB-REDUCER: A System That Checks for Inconsistency and Redundancy in Knowledge Bases.* Technical Report, AT&T Bell Laboratories, 1987.

[Khan and Dube, 1987] N. Khan and R. Dube. The gems trunk-trouble analyser: a knowledge based expert system for trunk maintenance. In *Proceedings of IEEE INFOCOM*, pages 459–465, San Francisco, CA, 1987.

[Mitchell *et al.*, 1986] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: a unifying view. *Machine Learning*, 1:47–80, 1986.

[Nguyen *et al.*, 1985] T. Nguyen, W. Perkins, T. Laffey, and D. Pecora. Checking an expert systems knowledge base for consistency and completeness. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 375–378, 1985.

[Reiter, 1980] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Suwa, *et al.*, 1982] M. Suwa, A. Scott, and E. Shortliffe. An approach to verifying completeness and consistency in a rule-based expert system. *The AI Magazine*, 3(3):16–21, Fall 1982.

[Weiss and Kulikowski, 1979] S. Weiss and C. Kulikowski. Expert: a system for developing consultation models. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 942–947, Tokyo, Japan, 1979.