KNOWLEDGE REFINEMENT DURING DEVELOPMENTAL AND FIELD VALIDATION OF EXPERT SYSTEMS

Neli P. Zlatareva
Department of Computer Science
Central Connecticut State University
New Britain, CT 06050
e-mail: zlatareva@ccsua.ctstateu.edu

Abstract

To ensure that Expert System (ES) performance remains above the level of acceptance throughout the entire life cycle of the system, its knowledge base must periodically be updated and upgraded. In this paper, we review refinement activities taking place during ES development and exploitation, and outline a domain-independent refinement framework intended to support them.

Introduction

Expert System (ES) validation aims to ensure that the system is error free and performs above certain level of acceptance throughout its entire life cycle. This suggests a two-step validation process, where the first step is often referred to as *developmental* validation, and the second step - as *field* validation. Three fundamental tasks involved in this process are testing, refinement, and maintenance of the Knowledge Base (KB). So far, most of the work in ES validation was concerned with testing the KB for structural and functional anomalies, little attention was paid to automated knowledge base refinement, and almost no attention was paid to knowledge base maintenance once the system is placed into exploitation under the assumption that it is similar to the maintenance of conventional software. This assumption is true only if we expect KB maintenance to be limited to fixing software bugs during system operation. Maintenance, however, also assumes updating the knowledge base to accommodate changes in the domain theory, and upgrading it as new knowledge become available. At present, activities related to ES maintenance are most often carried out by a maintainer who is not an expert in the problem domain and thus not capable of extending the maintenance process beyond fixing software bugs. In many cases, this results in a decreased acceptance of the ES over time.

In this paper, we advocate the idea that a refinement tool capable of assisting the maintainer with updating and upgrading of the KB can substantially expand the scope of his activities. Furthermore, we believe that refinement tasks involved in ES maintenance are similar to those taking place at the development stage, and thus a uniform refinement framework can be designed to carry out both activities. Next, we review refinement tasks required at different stages of ES life cycle, and present a refinement theory intended to support them.

Refinement Tasks: definitions and characterization

During ES development, knowledge base refinement is mainly concerned with knowledge revision and knowledge restructuring, while at the exploitation stage the two major refinement tasks are knowledge update and knowledge upgrade.

Knowledge Revision

Knowledge revision aims to improve the inferential accuracy of the KB-theory. Given the initial KB-theory $Th(DB, R)$, where DB is a set of ground facts describing the problem domain, and R is a set of first-order Horn clauses involving elements from DB, a set of positive test cases with known solutions E^+ (optional), and a set of negative test cases E^- (optional), the knowledge revision problem consists in finding a refined theory $ref\{Th(DB, R)\}$, which contains no inconsistencies and incompleteness with respect to the set of

final hypotheses defined in DB, and which can correctly classify all of the positive and negative test cases (if such are provided).

It is important to note that KB revision must be carried out even when no test cases are available. The need for this arises at early stages of ES development. Among the errors which can be detected and resolved at this point are overgeneralization errors resulting in logical or semantic inconsistencies and overspecialization errors caused by redundant knowledge.

Theory revision has been extensively studied in theoretical AI [Gardenfords, 1992, Wrobel, 1996] and machine learning
[Ourston and Mooney, 1994]. Although some of the results from these areas are applicable to KB refinement, there are important differences between them such as:

- Theory revision requires changes made to the theory to be minimal. This requirement does not hold in KB refinement. However, it is expected that a KB refinement tool will address *all* of the detected structural and functional errors, whereas theory revision can be limited to resolving only *some* of the errors.

- Theory revision systems often use inductive learning techniques, which require a significant number of test cases. In KB refinement, it is unrealistic to expect a large number of test cases to be available which makes these techniques inappropriate for KB refinement.

- Although some machine learning techniques do need fewer test cases (for example, apprenticeship learning), a common requirement for theory revision and machine learning techniques is to have all of the test cases available before the revision process is initiated. In KB refinement, test cases are accumulated throughout the development process and the refinement algorithm should be able to utilise them as they arrive.

- Limited KB refinement must be carried out even if no test cases are available. Syntax-independent theory revision techniques [Gardenfords, 1992] address this problem by means of postulates for expansion, contraction and revision. Implementation of these techniques, however, is computationally very expensive which makes them inappropriate for KB refinement.

To summarize, a KB refinement tool is expected to perform in two different settings depending on the availability of test cases. If considerable number of test cases with known solutions is available, existing empirical learning techniques [Ourston and Mooney, 1994] can be used to implement knowledge base revision. If no or few test cases are available, certain types of revisions such as those originated by logical or semantic inconsistencies, must still be carried out. Further in this paper we introduce a small number of domain independent transformation rules intended to locate culprits for detected inconsistencies and suggest possible revisions of the existing KB-theory.

Knowledge Restructuring

Knowledge restructuring aims to eliminate redundancies, subsumptions and circularities from the KB-theory in order to assure its convergency and efficiency. The knowledge restructuring task can be defined as follows. Given initial KB-theory $Th(DB, R)$, find a refined theory $ref\{Th(DB, R)\}$, where no alternative explanations exist for any element from DB.

Although restructuring of the knowledge base is not intended to change the set of conclusions generated by the ES (which is why no test cases are required for this activity), it is important for the following reasons:

1. Guarantees the confluence of the rule set. As stated in [Schmolze and Snyder, 1995], confluence is an important property of the rule set, which ensures the determinism and convergency of the knowledge base *independently* from a particular conflict resolution strategy. Furthermore, it simplifies the testing of the KB-theory, because for proving the correctness of a confluent rule set it is enough to show that each rule participates in one valid path.

2. Improves the run-time efficiency of the system. Redundant or subsumed rules may cause the system to perform too slow without providing any benefits in terms of understandability, while circular rules may cause intractable rule chains. Although in some cases (especially if the ES utilizes uncertain or incomplete knowledge) alternative paths do provide useful information, it is important that redundant, subsumed and circular rules be analyzed for potential performance inefficiencies during ES validation.

3. Guaranties the completeness of subsequent refinements by eliminating the possibility that only one of the alternative paths leading to the wrong conclusion is revised.

One way to improve the efficiency of the KB-theory is provided by Explanation-Based Learning (EBL) [DeJong and Mooney, 1986]. EBL builds new operational rules from the explanations produced by the existing theory in its attempt to explain presented test cases, given that the KB does not contain circular rules. In this process, redundant and subsumed rules are automatically eliminated. The resulting theory, however, is in an operational form, which reduces its understandability and accessibility. A KB refinement tool must go one step further and "decompile" the restructured KB-theory back in the original language.

Knowledge Update

Knowledge update is required when the ES is expected to perform in an evolving environment. In contrast to KB revision, which is intended to improve the KB-theory "internally" (i.e. to get rid of structural or functional errors in the KB-theory itself), KB update accounts for "external" changes, where new knowledge must be taken into account [Lea Sombe, 1994]. The KB update task can be defined as follows. Given initial KB-theory $Th(DB, R)$, new ground facts and/or rules, or new sets of positive/negative test cases, E^+_{new} and E^-_{new} respectively (where $E^+_{new} \cup E^+ \Rightarrow \perp$ or $E^-_{new} \cup E^- \Rightarrow \perp$), find revised, internally consistent sets of test cases E^+_{rev} and E^-_{rev} and a refined theory $ref\{Th(DB, R)\}$, which contains no inconsistencies, all test cases from E^+_{rev} are successfully classified by the refined theory, and none of the negative test cases E^-_{rev} holds in any identifiable real-world situation.

At this stage, we limit our attention to the case where new knowledge is represented as rules or facts. Incorporating these in the existing KB-theory may introduce new logical or semantic inconsistencies the detection of which requires a new validation - refinement cycle.

Knowledge Upgrade

To keep ES adequacy above the level of acceptance, an upgrade of the KB-theory may be required at the exploitation stage. This change can be viewed as a simple monotonic extension of the existing KB-theory, which is why it is not expected to originate new inconsistencies. However, new redundancies and subsumptions are possible if existing rules are affected in this process. Therefore, a subsequent restructuring of the upgraded knowledge base may be required.

The KB upgrade task can be defined as follows. Given initial KB-theory $Th(DB, R)$, new ground facts and/or new rules, new set of positive test cases E^+_{new} (where $E^+_{new} \cup E^+$ is consistent), or new set of negative test cases E^-_{new} (where $E^-_{new} \cup E^-$ is consistent), find a refined theory $ref\{Th(DB, R)\}$ which contain no redundancies or subsumptions, and which correctly classifies all test cases from E^+_{new} and E^-_{new}. Note that KB upgrade can be viewed as a machine learning task if new knowledge is defined by means of test cases, or as a knowledge integration task, if new knowledge is available in a form of rules or facts. In the former case, some of the existing machine learning techniques (for example, [Paszani and Brunk, 1991]) can be used for acquiring new rules from the new test cases. Here we are concerned with the later case, and we assume that new knowledge is available in a rule or data format. Integration of the new knowledge requires search for relevant knowledge in the existing KB-theory to ensure that necessary restructuring of the resulting KB-theory takes place.

Refinement Theory Using Problem-Independent Transformation Rules

In the previous sections, we have outlined refinement tasks required at different stages of ES life cycle. We have assumed that KB-theories are described as first-order Horn clauses. Such theories may contain structural or functional errors, which can be revealed by means of the DIVER tool [Zlatareva and Preece, 1994]. During each validation cycle, DIVER identifies potential logical and semantic inconsistencies defined in terms of data and rules involved. Detected inconsistencies can be sorted into levels depending on how early in the operationalization process a given inconsistency was detected, and earliest inconsistencies should be resolved first. Also, inconsistencies detected at the same level are likely to be caused by independent anomalies, which is why they must be handled concurrently. Inconsistencies at higher levels may depend on inconsistencies detected earlier, which is why their resolution should be postponed until inconsistencies at the lower levels are resolved.

Let $C_{F_i/F_j} : (F_i, F_j)(C_{F_i/F_j})$ be the inconsis-

tency currently examined (for logical inconsistencies, $F_j = \neg F_i$), and SE_k be a stable extension generated by DIVER at the current step of the validation process. The refinement procedure searches SE_k for formulas with heads[1] F_i or F_j. The union of the *T-sets* of these formulas provides an explanation of F_i relative to SE_k. Similarly, the union of the *T-sets* of the formulas with head F_j provides an explanation for F_j. To construct the explanation for inconsistency C_{F_i/F_j}, the refinement tool computes the intersection of the explanations for F_i and F_j, namely
$(E1_{F_i} \vee E2_{F_i} \vee ... \vee En_{F_i}) \wedge (E1_{F_j} \vee E2_{F_j} \vee ... \vee Em_{F_j})$.

This can be represented in the following equivalent form:
$E1_{F_i} \wedge E1_{F_j}$, ..., $E1_{F_i} \wedge Em_{F_j}$, $E2_{F_i} \wedge E1_{F_j}$, ..., $E2_{F_i} \wedge Em_{F_j}$, ... $En_{F_i} \wedge E1_{F_j}$, ..., $En_{F_i} \wedge Em_{F_j}$.

Note that each of these formulas provides an independent explanation for C_{F_i/F_j}. These explanations may explicate independent anomalies in the KB-theory causing the same error. This is why each explanation is searched for recognizable sources of potential errors by means of the tests described next.

Search for duplicated data

If the T-sets of F_i and F_j contain the same datum, then this datum will be encountered twice in the explanation for C_{F_i/F_j}. Such a datum is likely to be either redundant (and therefore, it should be removed from the affected rules), or the culprit for an inconsistency.

Consider the following two rules: $A \wedge B \rightarrow H_i$ and $C \wedge B \rightarrow \neg H_i$. There are two obvious reasons as to why these rules may be inconsistent: (i) A and C are inconsistent, in which case B is redundant, and (ii) B is the culprit for the inconsistency, in which case it must be removed from the left hand side of one of the two rules. Note that the latter case is equivalent to specializing a predecessor rule with the negation of B, if we want to preserve the current explanation for the corresponding conclusion. This is captured by the following *transformation rules* (TRs) intended to suggest possible refinements of the KB-theory rules containing a duplicated datum.

- **TR 1**: Generalize the rules containing the duplicated datum by removing that datum from their left hand sides.
- **TR 2**: If the duplicated datum participates as a premise in same-level conflicting rules, then select (arbitrary) one of these rules for generalization, and remove the duplicated datum from the left hand side of that rule.
- **TR 3**: If the duplicated datum participates as a premise in conflicting rules fired at different levels of the operationalization process, select (according to TR 6 below) one of the conflicting rules for specialization with the negation of the duplicated datum.

Another type of an error caused by a duplicated datum is revealed by checking if one of the conflicting rules is a specialization of the other. Then, the more general rule may fire inappropriately if nothing prevents it from firing when the more specific rule fires. For example, consider $A \wedge B \rightarrow H_i$ and $B \rightarrow \neg H_i$. Here B will be encountered twice in the explanation for $C_{H_i/\neg H_i}$, and the former rule is a specialization of the latter. To get rid of this inconsistency, the latter rule must be specialized with $\neg A$. This case is captured by the following transformation rule:

- **TR 4**: If one of the conflicting rules is a specialization of the other, then the more general rule must be specialized with the negation of the premise serving as a specialization condition for the other rule.

A special case of a duplicated datum is the one where the head of the formula is an element of its own T-set. This case suggests a circularity in the KB-theory. To reveal the rule causing it, the underlying set of rules is checked for rules whose conclusions are declared as elements of the input data set. This is captured by the following transformation rule:

- **TR 5**: A rule whose conclusion is defined as an input datum is likely to cause a circular rule chain, which is why this rule must be removed from the KB-theory.

[1] Stable extensions consist of two types of formulas: $F_i : (B_1, ..., B_m)()$ and $F_j : (A_1, ..., A_n)(C_{F_k}, ...C_{F_k})$, where F_i and F_j are KB-theory statements (facts or hypotheses) called the *head* of the corresponding formula; $\{B_1, ..., B_m\}$ and $\{A_1, ..., A_n\}$ represent the evidence for F_i and F_j, respectively, and are called the *T-sets*; and $\{C_{F_k}, ..., C_{F_k}\}$ represents underlying inconsistencies upon which F_j depends, and is called the *U-set*.

Search for conflicting rules

Rules involved in a given explanation are ordered into levels depending on how early in the operationalisation process the rule has been fired. It is reasonable to assume that "level 0" rules (directly acquired from the domain expert) are most likely to be correct. Under this assumption, the following transformation rule defines the possible candidate for specialization among a given set of conflicting rules:

- **TR 6:** If conflicting rules belong to different levels, then the highest level rule is selected for specialization.

Search for redundant and subsumed rules

If the T-sets of F_i and F_j are the same or the data set of one T-set is a subset of another, then there are redundant or subsumed rules among those comprising the rule parts of the T-sets. Such rules are identified by the following transformation rules:

- **TR 7:** If a rule contains a subsumed term, then this rule must be generalized by substituting the subsumed term with its generalisation.

- **TR 8:** A rule whose conclusion's T-set is comprised by data which is a superset of the T-set of the same conclusion inferred by a different set of rules is redundant, and it must be removed from the KB-theory.

It is easy to see that the number of refinements suggested by transformation rules TR 1 to TR 8 is very small, and it is feasible to exhaustively test them by validating the revised theory for each subset of independent refinements.

Conclusion

We have shown in this paper that refinement tasks involved in ES development and maintenance are similar, and therefore a uniform refinement framework can be designed to support both activities. We have reviewed four refinement tasks required at different stages of ES life cycle: revision, restructuring, upgrade and update of the rule set, and have outlined a refinement framework capable of supporting them. The proposed refinement theory was tested on relatively large examples, and it was able to correct about 72% of the detected performance errors. Some performance errors, however, require additional knowledge (in a form of test cases, for example) in order to be corrected. We believe that incorporating a learning from test cases component in the proposed refinement framework will improve its ability to handle a larger variety of performance errors. We plan to tackle this aspect of knowledge base refinement in our future work.

References

[DeJong and Mooney, 1986] DeJong, G. and Mooney, R. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176.

[Gardenfords, 1992] Gardenfords, P. Belief revision: an introduction. In *Belief revision* (P. Gardenfords, Ed.), Cambridge University Press, Cambridge, UK.

[Lea Sombe, 1994] Lea Sombe. A Glance at Revision and Updating in Knowledge Bases. *International Journal of Intelligent Systems*, 9(1):1–27.

[Ourston and Mooney, 1994] Ourston, D. and Mooney, R. Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, 66:273–309.

[Pazzani and Brunk, 1991] Pazzani, M. and Brunk, C. Detecting and correcting rule-based expert systems: an integration of empirical and explanation-based learning. *Knowledge Acquisition*, 3:157–173.

[Schmolze and Snyder, 1995] Schmolze J. and W. Snyder. A Tool for Testing Confluence of Production Rules. In *Working Notes of IJCAI'95 Workshop on Verification and Validation of Knowledge-Based Systems*, pages 77–84, Montreal, Canada.

[Wrobel, 1996] Wrobel, S. First Order Theory Refinement. In *Advances in Inductive Logic Programming* (ed. L.De Raedt), IOS Press, pages 14–38.

[Zlatareva and Preece, 1994] Zlatareva, N. and Preece, A. An effective logical framework for knowledge-based systems verification. *International Journal of Expert Systems: Research and Applications*, 7.