Towards a Quantitative, Platform-Independent Analysis of Knowledge Systems†

Noah S. Friedland[1], Paul G. Allen[1], Michael Witbrock[2], Gavin Matthews[2], Nancy Salay[2], Pierluigi Miraglia[2], Jurgen Angele[3], Steffen Staab[3], David Israel[4], Vinay Chaudhri[4], Bruce Porter[5], Ken Barker[5], Peter Clark[6]

[1]Vulcan Inc., 505 5th Ave S, Seattle, WA 98104
[2]Cycorp Inc., Suite 100, 3721 Executive Center Drive, Austin, TX 78721
[3]ontoprise GmbH, Amalienbadstraße 36, 76227 Karlsruhe, Germany
[4]SRI International, 330 Ravenswood Ave., Menlo Park, CA 94025
[5]Computer Science, University of Texas at Austin, Austin, TX 78712
[6]Boeing Phantom Works, The Boeing Company, Seattle, WA 98124

Abstract

The Halo Pilot, a six-month effort to evaluate the state-of-the-art in applied Knowledge Representation and Reasoning (KRR) systems, collaboratively developed a taxonomy of failures with the goal of creating a common framework of metrics against which we could measure inter- and intra- system failure characteristics of each of the three Halo knowledge applications. This platform independent taxonomy was designed with the intent of maximizing its *coverage* of potential failure types; providing the necessary *granularity* and *precision* to enable clear categorization of failure types; and providing a *productive* framework for short and longer term corrective action.

Examining the failure analysis and initial empirical use of the taxonomy provides quantitative insights into the strengths and weaknesses of individual systems and raises some issues shared by all three. These results are particularly interesting when considered against the long history of assumed reasons for knowledge system failure. Our study has also uncovered some shortcomings in the taxonomy itself, implying the need to improve both its granularity and precision. It is the hope of Project Halo to eventually produce a failure taxonomy and associated methodology that will be of general use in the fine-grained analysis of knowledge systems.

Introduction

Since the first expert systems were first developed forty years ago, a great many knowledge-representation and reasoning (KR&R) systems have been fielded. Some – very few – have been carefully evaluated; these evaluations have typically yielded data on the systems' overall performance, and occasionally have drawn comparisons with other systems or with the performance of people. Our goal is to go beyond *evaluations* of KR&R systems to an *analysis* of them. We seek to understand *why* these systems fail when they do, the relative frequency of each type of failure, and the ways these failures might be avoided or mitigated.

This is a major undertaking, and we have taken only the initial steps. First, we have designed a taxonomy of failures that fielded KR&R systems might experience. This step is necessarily speculative, since we have not studied a large sample of systems or surveyed their developers, but it is based on the authors' collective experience building many systems using a variety of different technologies. Second, we have built three KR&R systems using state-of-the-art technologies and carefully evaluated their performance in a pilot study. Although the systems did quite well overall, they nevertheless exhibited many shortcomings, yielding a large corpus of failures. Third, we analyzed each of these failures and attempted to place it within the taxonomy. We studied the resulting data to draw lessons about the taxonomy, the systems, and (by extrapolation) the current state of KR&R technologies for building fielded systems.

A Class of Knowledge-based Systems: The Halo Pilot

This effort to analyze KR&R systems and to better understand the causes of their failures arises in the context of Project Halo, a multi-stage effort funded and managed by Vulcan Inc. to develop a "Digital Aristotle", an application that will encompass a substantial amount of scientific knowledge and be capable of answering unanticipated questions using advanced problem-solving. Vulcan sees two primary functions for the Digital Aristotle: first, as a tutor capable of instructing students in the sciences; and second, as a research assistant with broad interdisciplinary skills able to help scientists in their work.

The data for our study was produced by the pilot phase of Project Halo. This was a six-month effort to evaluate the state-of-the-art in fielded KR&R systems performing deep

† Full support for this research was provided by Vulcan Inc. as part of Project Halo

reasoning. Three teams were contracted to participate in the evaluation: a team led by SRI International with substantial contributions from Boeing Phantom Works and the University of Texas at Austin; Cycorp; and Ontoprise. The objective of the evaluation was to determine whether current KR&R technologies were capable of correctly answering novel (previously unseen) questions and of providing concise, readable answer justifications.

Significant attention was given to domain selection for the evaluation. It was important, given the limited scope of this phase of the project, to adapt an existing, well-known evaluation methodology with easily understood and objective standards. Several standardized test formats were examined. A 70-page subset of Advanced Placement (AP) chemistry was selected because it was reasonably self-contained and did not require solutions to other hard AI problems, such as spatial or uncertain reasoning, or understanding diagrams. Topics included: stoichiometry calculations with chemical formulas; aqueous reactions and solution stoichiometry; and chemical equilibrium. This scope was large enough to produce many novel, and hence unanticipated, question types. One analysis of the syllabus identified nearly 100 distinct rules, suggesting that it was rich enough to require complex inference. It was also small enough to be represented in four months, the time allocated to the teams for knowledge formulation.

Each team developed their KR&R systems using their existing, and very different, technologies. Team SRI's implementation was based upon their SHAKEN system, [1] which uses the frame-based KM language and inference engine [2]. They employed an existing component library (CLIB) of representations of reusable, generic events, entities, roles, and relations to facilitate rapid development of knowledge [3]. Cycorp built the OpenHalo chemistry knowledge base upon their public OpenCyc technology, extended as needed by constructs from the main Cyc engine. The Cyc system is designed with the intention of representing the broad range of knowledge required by a general Artificial Intelligence, and achieves a partial functional partition into consistent functional domains using hierarchies of ontologies called microtheories. The knowledge in all Cyc systems is represented in CycL, Cycorp's formal language, which includes first-order logic and some second-order and modal constructs [4]. Ontoprise built their OntoNova system on top of their OntoBroker® technology, [5, 6] which uses F-Logic [7], a logic-programming language similar to Prolog but with an object-oriented syntax. This implementation was constructed without the benefit of any pre-existing knowledge infrastructure like that represented by the SRI CLIB or the Cycorp upper ontology. The final SRI and Ontoprise knowledge bases were on the order of 500 concepts, rules and relations, while Cycorp's openHALO included about 15,000 concepts, of which approximately 14,000 were preexisting general terms from OpenCyc, and approximately 1000 were added to support the AP Chemistry task.

The three teams also employed very different approaches to answer justification. Cycorp used its generative English capabilities to produce English language explanations from its proof trees. Meta-reasoning was used to remove explanation components that would be extremely obvious to the domain expert or that addressed Cyc's internal inference methodology and would not be easily understood by a domain expert. Ontoprise used a dual inference process in its question answering. The first process attempted to derive the answer. If successful, the second process used the first proof tree along with rule specific human-authored explanation templates to produce the explanation. SRI also relied on human-authored explanation templates associated, in this case, with chemical "methods" built into their knowledge representation.

Upon completion of the knowledge formulation, all three systems were sequestered on identical servers. Then the challenge exam was released to the teams, who were given two weeks to encode its questions in their respective formal languages. The exam consisted of three sections: 50 multiple-choice questions and two sets of 25 multipart questions. Upon completion of the encoding effort, the formal question encodings of each team where evaluated by a program-wide committee to guarantee high *fidelity* to the original English. Once the encodings were evaluated, Vulcan personnel submitted them to the respective sequestered systems. The evaluations ran in batch mode. The Ontoprise system completed its processing in two hours, the SRI system in five hours and the Cycorp system in a little over 12 hours.

Three chemists were engaged to evaluate the exams. Adopting an AP-style evaluation methodology, they graded each question both for correctness and the quality of its explanation. The exam encompassed 168 distinct gradable components consisting of questions and question sub-parts. Each of these received marks—ranging from 0 to 1 point each for correctness and explanation quality for a maximum high score of 336. All three experts graded all three exams. The scoring of all three chemistry experts was aggregated for a maximum high score of 1008. The graded exams were distributed to the Halo teams to serve as the basis for their failure analysis. The guidelines for the analysis included producing written explanations for every point loss on a question-by-question basis and association of every point loss to a category in the taxonomy of types of failures. Details about the Halo Pilot – including the exam, the systems' answers and explanations, and the graders' scores and comments – are available at the Project's Web site: http://www.projecthalo.com.

A Taxonomy of Types of Failures

Our goal was to design an implementation neutral taxonomy of failure types, exhibiting these qualities:

Coverage: The taxonomy must be broad enough to account for virtually every type of failure that a fielded KR&R system might experience.
Precision: The categories in the taxonomy should be clearly defined so that every failure can be unambiguously classified into one or more categories.
Granularity: The categories should be defined at a fine enough level so that they capture distinctions among interesting classes of problems. If all errors fall into one category, the categories are too coarse. If each category has only one instance of a problem, the categories are probably too fine-grained.
Productivity: The categories should be defined in a way that they clearly suggest an action that could be taken to address it. It should also be clear how each type of failure, if uncorrected, would affect system performance.
To meet the requirement of coverage, we created top-level categories in the taxonomy for the primary issues in building a question-answering system for the chosen subset of AP chemistry – one that receives previously unseen queries in a formal language and generates answers and justifications appropriate to the user. These categories are:
(MOD) Knowledge Modeling: the ability of the knowledge engineer to model information or write the needed axioms.
(IMP) Knowledge Implementation/Modeling Language: the ability of the representation language to accurately and adequately express the axioms.
(INF) Inference and Reasoning: the ability of the inference engine to do the reasoning required to compute correct answers.
(KFL) Knowledge Formation and Learning: the ability of the system (KB + inference engine) to acquire and merge knowledge through automated and semi-automated techniques
(SCL) Scalability: the ability of the KB to scale.
(MGT) Knowledge Management: the ability of the system to maintain, track changes, test, organize, document its current state; the ability of the knowledge engineer to inspect and revise knowledge.
(QMN) Query Management: the ability of the system to robustly answer queries.
(ANJ) Answer Justification: the ability of the system to provide justifications for answers in the correct context and at the appropriate level of detail.
(QMT) Quality Metrics: the ability of the developers to evaluate the knowledge base throughout its development.
(MTA) Meta Capabilities: the system's ability to employ meta-reasoning or meta-knowledge.
To meet the precision and granularity requirements, we refined the top-level categories into 24 more specific ones, grouped under the ten original ones. For example, for failures due to the implementation/modeling language (IMP), we created three sub-categories, including failures due to the language being insufficiently expressive (B-IMP-1) or being overly expressive (B-IMP-2). Although increased expressiveness has obvious benefits for knowledge engineering, it can, in the limit at least, come at the expense of tractable inference [8]. Distinguishing between failures of these two types might enable analysts to measure the costs and benefits of such enhancements. See Table 1.

Table 1: A Taxonomy of Types of Failure

Category	Type	Name and Description
Modeling	B-MOD-1	**Modeling Error Failure**. The knowledge engineer fails to model domain knowledge properly (the act of writing the axiom).
	B-MOD-2	**Modeling Assumption Failure.** Implicit "context" assumptions are not articulated, making it difficult for knowledge engineers to model/extend/modify information. Designers working from disparate assumed "context models" introduce conflicts into the KB. Resolving multiple contexts creates large, unwieldy rule sets.
	B-MOD-3	**Modeling Primitive Failure**. Limitations of the KR language make straightforward representation difficult resulting in errors or complex representations.
	B-MOD-4	**"Islands of Knowledge" Failure**. The knowledge engineer fails to make explicit connections between the domain model and the existing ontology/KB. The system cannot take advantage of existing knowledge to achieve the desired reasoning performance.
Implementation, Language	B-IMP-1	**Under-expressive Language Failure.** The KR language is not expressive enough to model the domain knowledge. The resulting convoluted representations or approximations give unexpected or undesirable results.
	B-IMP-2	**Over-expressive Language Failure**. The KR language is overly expressive. Certain representations make inference intractable.
	B-IMP-3	**External Module Interface Failure**. The KR language allows representations that do not readily translate to the representation states of external modules.

Category	Type	Name and Description
Management	B-MGT-1	**Large KB Learning Failure**. The knowledge engineer has difficulty learning the existing ontology/KB due to its size and complexity. Poor search and documentation tools compound this problem.
	B-MGT-2	**Large KB Extension Failure.** The knowledge engineer has difficulty extending the existing large, highly interconnected KB. The number of modeling errors (B-MOD-1) increases with KB magnitude and connection factor.
	B-MGT-3	**Large Team Failure.** The development team fails to communicate modeling assumptions, track versions, coordinate changes, etc. among team members. The assumptions, errors or conflicts lead to unpredictable system performance.
Formation, Learning	B-KFL-1	**Information Extraction Failure.** Information Extraction techniques over unstructured data produce insufficiently deep models of domain knowledge. The system is unable to reason adequately over shallow domain representations.
	B-KFL-2	**Knowledge Mapping Failure.** The knowledge engineer fails to merge structured knowledge from multiple sources appropriately (either due to merging errors or irreconcilable representational differences). The system either cannot take advantage of knowledge from multiple sources or suffers from inconsistencies.
Inference, Reasoning	B-INF-1	**Inference Engine Conceptualization Failure.** The knowledge engineer models domain knowledge based on faulty understanding of the inference algorithms. The inference engine produces unexpected results.
	B-INF-2	**Inference Engine Bug Failure.** Errors in the implementation of the inference engine cause unexpected or undesirable results.
	B-INF-3	**"Practical Incompleteness" Failure.** The resource challenges of deep KBs prevent exhaustive search. The system fails to return an answer even though the information exists in the KB. Sensitivity to initial conditions makes search success unpredictable.
	B-INF-4	**Consistency Failure.** Hard contradictions cause deductive reasoning systems to fail. Large KBs that encompass many topics are susceptible to contradictions.
	B-INF-5	**Numeric Instability Failure.** Failure to factor numerical aspects of computation into query responses leads to incorrect or inappropriate answers.
Query Management	B-QMN-1	**Query Scoping Failure.** The query encoding misses implicit assumptions or incorrectly includes irrelevant information from the query. The missing or extraneous information prevents the system from answering the query successfully.
	B-QMN-2	**Query Encoding Failure.** Sensitivity to the query encoding leads to unexpected or undesirable results.
Answer Justification	B-ANJ-1	**Exposition Failure.** Answer justifications are overly dependent on idiosyncrasies of the reasoning steps and/or proof tree. The resulting explanations may contain irrelevant, redundant or out-of-sequence information, making them unintuitive to a human reader.
	B-ANJ-2	**Answer Template Failure.** Manually created answer justification templates produce static justifications at fixed resolution independent of context.
	B-ANJ-3	**Context Justification Failure.** The answer justification mechanism is unable to produce user- and context-appropriate justifications.
Quality Metrics	B-QMT-1	**Quality Metrics Failure.** The KB quality metrics fail to provide needed feedback on the knowledge engineering process. The knowledge engineers cannot accurately determine coverage and completeness, resulting in gaps in the KB.
Meta-capabilities	B-MTA-1	**Meta Capabilities Failure.** The KB lacks required meta-knowledge (either due to omission or KR language insufficiency). The system performs poorly on questions requiring meta-reasoning.
Other	OTHER	Failure for reasons other than the above.

To meet the requirement of being productive, we elaborated the descriptions of each type of failure with the following attributes:

A list of influences: the high-level influences that typically contribute to failures of this type

An example: the symptoms (in terms of system behavior) that this type of failure might cause

Mitigating factors: technologies and methods that might mitigate failures of this type

Long-term research: research directions that might reduce or eliminate failures of this type

Table 2: Complete Taxonomy Entry for B-MOD-1

Failure	Influences	Description	Example	Mitigation	Future Research
B-MOD-1	MOD, QMT	**Modeling Error Failure** The knowledge engineer fails to capture domain information properly in their modeling (the act of writing the axiom).	Classifying chemical as an acid independent of the reaction.	Review processes to validate that domain-specific information is captured correctly; SME testing of the system; SME involvement throughout	Tools to better facilitate knowledge modeling by domain experts; Automated techniques to vet completeness and coverage of KB formation

Table 2 gives the complete entry for the failures of type B-MOD-1, a prevalent and intriguing type of failure of the Halo Pilot systems.

Evaluation of the Taxonomy of Failure

In this section we summarize the results of the failure analysis for the three systems. The numbers against each slice of the pie charts represent the points lost that could be attributed of the corresponding category. For example, in Figure 1.a, Cycorp's loss of 64.63 points could be ascribed to B-ANJ-1 in the system's ability to produce readable answer expositions appropriate to the context and user (AP chemistry exams, and their graders, respectively). The process of ascribing points of failure to positions in the taxonomy may not have been uniform, since it was performed by different groups for each system, working independently. The Cycorp analysis reported significant problems in 7 categories, and the SRI and Ontoprise systems reported significant problems in four categories each. Each of the three teams attributed a significant number of failures to the "other" category, meaning that the performance problem could not be attributed to any of the categories in the taxonomy. In SRI's case, most of these points reflected failures due to gaps in knowledge attributed to lack of implementation time. Most of Cycorp's "other" scores were attributed to points lost on answer justifications for questions that were not scored as having been answered correctly. Ontoprise reported a number of reasons for classifying points lost in the "other" category. Some of these were related to disputes over the details of question grading and whether the final questions were within the design scope of the pilot evaluation[1].

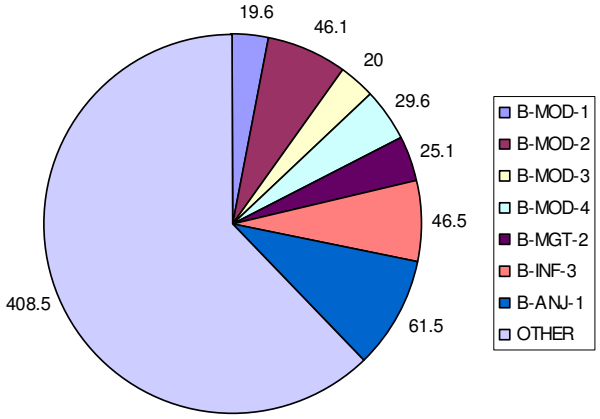

Figure 1.a: Cycorp Failure Analysis. Most of the points lost due to failure were the result of a failure to represent some element of chemistry knowledge (B-MOD-1), difficulty in producing justifications in the form expected by domain experts (B-ANJ-1), and a lack of inference completeness with respect to the available knowledge (B-INF-3).

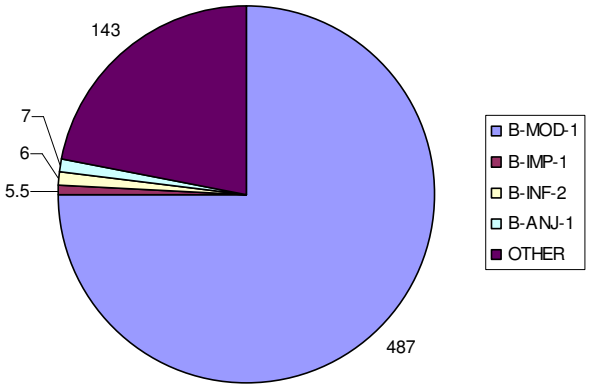

Figure 1.b: Ontoprise Failure Analysis. Most of the points lost due to failure were the result of a failure to represent some element of chemistry knowledge (B-MOD-1).

[1] These disputes were not confined to Ontoprise, and were surprisingly many in number in light of the standardized nature of the test. For the purposes of the pilot evaluation, these were resolved by simply accepting the scoring produced by the judges.

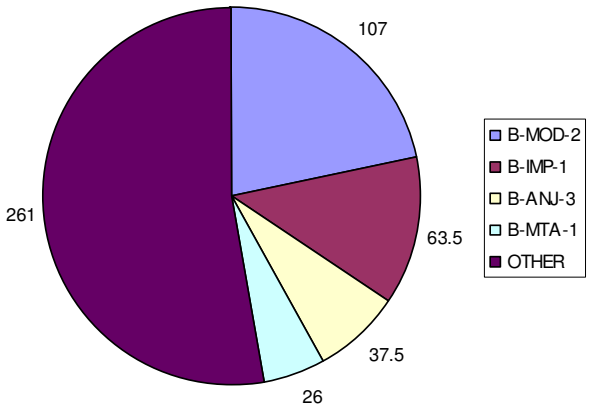

Figure 1.c: SRI Failure Analysis. Most of the points lost due to failure were the result of inappropriate modeling assumptions (B-MOD-2), difficulty in modeling knowledge due to the expressiveness of the KR language (B-IMP-1), and inflexible justification generation (B-ANJ-3).

Table 3, below, displays the categories used by each of the three systems. Interestingly, there is no category that was used in analysing the errors of all three systems, even though there are several that were used in the analysis of two of the three systems. In general, modeling problems affected all three systems. B-MGT-2 and B-INF-3, which are associated with the size of the knowledge base, primarily affected the Cyc system. Ontoprise and SRI were both affected by B-IMP-1, which represents problems due to lack of expressiveness.

	Cycorp	Ontoprise	SRI
B-MOD-1	✖	✖	
B-MOD-2	✖		✖
B-MOD-3	✖		
B-MOD-4	✖		
B-MGT-2	✖		
B-IMP-1		✖	✖
B-IMF-2		✖	
B-IMF-3	✖		
B-ANJ-1	✖	✖	
B-ANJ-3			✖
B-MTA-1			✖

Table 3: Failure Category Usage

Given the above observations, it is worth reviewing the original goals of the failure taxonomy to see how well they were met. Since the taxonomy was designed with the functional components of early versions of the various Halo systems in mind, it should have been expected to display good coverage of forms of failure exhibited by those systems. The substantial use of the "other" category and some possible overuse of the B-MOD-1 category suggest that the proposed taxonomy does not have enough precision; given a failure, it is not always possible to clearly and precisely attribute it to a failure category. Many of these problems could have been remedied procedurally during the project by establishing a reconcilliation process to ensure that the taxonomy was employed correctly and consistently by all reporting teams. Clearly such a process will need to be tested, and suggested modifications to the taxonomy will need to be applied before it is ready for more widespread use.

Nevertheless, the taxonomy did prove to be suggestive in indicating functional areas of weakness in the three systems. Our failure analysis suggests three broad lessons that can be drawn across the board for the three systems:

Modeling: A common theme in the modeling problems across the systems was that the incorrect knowledge was represented, or some domain assumption was not adequately factored in, or the knowledge was not captured at the right level of abstraction. Addressing these problems requires us to have direct involvement of the domain experts in the knowledge engineering process. The teams involved such experts to different extents, and at different times during the course of the project. The SRI team, which involved professional chemists from the beginning of the project, appeared to benefit substantially. This presents a research challenge, since it suggests that the expositions of chemistry in current texts are not sufficient for building or training knowledge-based systems. Instead, a high-level domain expert must be involved in formulating the knowledge appropriately for system use. Two approaches to ameliorating this problem that are being pursued by participants are: 1) providing tools that support direct manipulation and testing of KRR systems by such experts, and 2) providing the background knowledge required by a system to make appropriate use of specialised knowledge as it is presented in texts.

Answer Justification: Explanation, or, more generally, response interpretability, is fundamental to the acceptence of a knowledge base system, yet for all three state-of-the-art systems, it proved to be a substantial challenge. A part of the reason for this was a failure to fully grasp the vital role played by the explanation mechanism in a deployed question-answering system. Since the utility of the system will be evaluated end-to-end, it is to a large degree immaterial whether its answers are correct, if they cannot be understood. Reaching the goals of projects like the Digital Aristotle will require an investment of considerably more resources into this aspect of systems to realize robust gains in their competence. Exactly what direction this research should take is not clear; deriving explanations automatically from the system's proof strategy is neither straightforward nor particularly sucessful, especially if that strategy has not been designed with explanation in mind. On the other hand, designing explanation templates that are tightly bound to specific problem types is also likely to engender brittleness when Halo systems apply their knowledge to a wider, more heterogeneous set of scientific questions. One approach to this problem, that is being pursued by all three teams in different ways, is to develop a meta-reasoning capability

that applies to the proof tree to construct a readable explanation.

Scalability for Speed and Reuse: There has been substantial work in the literature on the tradeoff between expressiveness and tractability, yet managing this tradeoff, or even predicting its effect in the design of fielded systems over real domains is still not at all straightforward. To move from a theoretical to an engineering model of scalability, the KR community would benefit from a more systematic exploration of this area driven by the empirical requirements of problems at a wide range of scales. For example, the three Halo systems, and more generally, the Halo development and testing corpora, can provide an excellent test bed to enable KR&R researchers to pursue experimental research in the tradeoff between expressiveness and tractability.

In addition to the primary sources of failure experienced by the system, it is important to note that several types of failure did not occur, even though they were predicted by previous research. Davis and King, in their seminal paper on rule-based systems [9], predicted that systems might fail because of unforeseen and incorrect rule interactions, and that the risk of these failures would increase with the size of the rule base. In our taxonomy this type of failure is B-MGT-2. Very few of these errors were experienced by the Halo systems.

The "uncertainty in AI" community predicts that KR&R systems might fail due to the use of axiomatic rules in uncertain environments. In general, this prediction might well be right. However, the Halo pilot systems ignored uncertainty, used only axiomatic rules, and did not fail because of it. It is important to note, however, that the system did not have to deal with raw data from the environment, which would necessarily introduce an important source of uncertainty.

Finally, the "commonsense reasoning" community predicts that KR&R systems might fail if they are unable to "fall back" on general principles. For example, this prediction has been an explicit motivation for the Cyc project [10]. Two of the the Halo Pilot systems (the ones built by Cycorp and the SRI team) had access to some degree of commonsense knowledge (i.e. general knowledge outside the domain of chemistry), but gained little benefit from it in this domain, and very few failures were attributed to lack of commonsense knowledge.

Undoubtedly, "hard science" domains, such as AP level chemistry, largely avoid issues of uncertainty and commonsense, but we were surprised that these issues scarcely arose at all. It is important and encouraging for the Digital Aristotle project that this class of KR&R systems can be at least somewhat successful in the absence of immediate solutions for these difficult problems in AI.

Additional Related Work

The cycle of fielding implemented systems, analyzing their performance, learning from that performance, and fielding the augmented version reflects the maturity of any scientific discipline. Not surprisingly, in the software engineering community, several aspects of system building have benefited by characterizing the system failures, and learning from them [11] [12]. Of course, there have been several efforts at documenting and analyzing the experience of implemented systems, [13], [14, 15]. DARPA's recent HPKB and RKF have made a pioneering effort to analyze and document the performance of the knowledge base performance [16, 17]. The present paper improves upon the HPKB and RKF evaluations by being more thorough and systematic, and by adopting an evaluation standard, such as an AP test, that is independent, objective and extensive enough to support a coherent, long-term development program.

Summary

Although many knowledge-based systems have been fielded and some have been evaluated, few have been analyzed to determine why they fail, the relative frequency of each type of failure, and the ways these failures might be avoided or mitigated. That is the goal of our work. We presented a taxonomy of failures that fielded KR&R systems might experience, and we have used the taxonomy to analyze failures of three question-answering systems built using state-of-the-art technologies for the challenging domain of AP chemistry. This analysis revealed important shortcomings of the KR&R technologies, as well as several weaknesses in the taxonomy itself.

References

1. Clark, P., et al., Knowledge Entry as the Graphical Assembly of Components, in *Proc 1st Int Conf on Knowledge Capture (K-Cap'01)*. 2001. p. 22-29.
2. Clark, P. and B. Porter, *KM - The Knowledge Machine: Users Manual.* 1999.
3. Barker, K., B. Porter, and P. Clark, A Library of Generic Concepts for Composing Knowledge Bases, in *Proc. 1st Int Conf on Knowledge Capture (K-Cap'01).* 2001. p. 14-21.
4. Guha, R.V. and D.B. Lenat, Cyc: A Mid-term report. *AI Magazine,* 1990. **11**(3).
5. Angele, J., *Operationalisierung des Modells der Expertise mit KARL*, . 1993, DISKI, Infix Verlag.
6. Decker, S., et al., eds. Ontobroker: Ontology-based Access to Distributed and Semi-Structured Information. *Database Semantics: Semantic Issues in Multi-media Systems,* ed. R. Meersmann. 1999, Kluwer Academics.
7. Kifer, M., G. Lausen, and J. Wu, Logical Foundations of Object Oriented and Frame Based Languages. *Journal of the ACM,* 1995. **42**: p. 741--843.
8. Levesque, H.J. and R.J. Brachman, Expressiveness and Tractability in Knowledge Representation and Reasoning. *Computational Intelligence,* 1987. **3**(2): p. 78-93.

9. Davis, R. and J. King, *The Origin of Rule-based Systems in AI, in Rule-Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project,* B.G. Buchanon and E.H. Shortliffe, Editors. 1984, Addison-Wesley: Reading, MA.
10. Lenat, D.B. and R.V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project.* 1990: p. 336.
11. Young, M. and R.N. Taylor. Rehinking the Taxonomy of Falult Detection Techniques. in *11th International Conference on Software Engineering*. 1989. Pittsburgh.
12. Perry, D.E., *An Empirical Study of Software Interface Faults,* 1985, AT&T Bell Laboratories: Murray Hill.
13. Brachman, R.J., et al., Reducing CLASSIC to ``Practice'': Knowledge Representation Theory Meets Reality. *Artificial Intelligence Journal,* 1999. **114**: p. 203-237.
14. Keyes, J., Why Expert Systems Fail? *IEEE Expert,* 1989. **4**: p. 50-53.
15. Batanov, D. and P. Brezillon, eds. *First International Conference on Successes and Failures of Knowledge-based Systems in Real World Applications.* 1996, Asian Institute of Technology: Bangkok, Thailand.
16. Cohen, P., et al., The DARPA High Performance Knowledge Bases Project. *AI Magazine,* 1998. **19**(4): p. 25--49.
17. Cohen, P., et al., Does Prior Knowledge Facilitate the Development of Knowledge-based Systems, in *Proceedings of the AAAI-99*. 1999. p. 221-226.