Applications of Abduction #1: Intelligent Decision Support Systems

Tim Menzies *

August 7, 1995

Abstract

We discuss using a single inference procedure (abduction) for implementing the various modules of an intelligent decision support systems.

1 Introduction

We define intelligent decision support systems (IDSSs) as model-based software systems that support *management comfort* in *vague domains*. In terms of a computational architecture, the main requirements for such a system are the ability to:

- *Validate* models;
- Perform inference over those models using *assumptions*;
- Managing mutually exclusive assumptions in separate *worlds*;
- Support domain specific criteria for finding the *best* worlds.

We find that a single inference procedure (*abduction*) satisfies all these requirements. Hence, we propose the use of abduction as a framework for IDSS.

All the above terms in *italics* are defined below.

Section 1 describes in detail our definition of IDSS. Section 2 describes abduction. Section 3 discusses using abduction for IDSS. Section 4 discuss the practicality of our proposal.

Note that this work is part of our *abductive reasoning project*. We believe that abduction provides a comprehensive picture of declarative KBS inference. Apart from IDSS, we argue elsewhere [25, 24] that abduction is useful also for prediction, classification, explanation, planning, qualitative reasoning, verification, diagrammatic reasoning, and multiple-expert knowledge acquisition. Further, abduction could model certain interesting features of human cognition [26]. Others argue elsewhere that abduction is also a framework for natural-language processing [28], design [34], visual pattern recognition [35], analogical reasoning [9], financial reasoning [15], machine learning [16, 30] and case-based reasoning [19].

*Dept. of Software Development, Monash University, Melbourne, Australia; `timm@insect.sd.monash.edu.au`

2 What is an IDSS?

Henri Fayol suggested in 1916 that managers plan, organise, co-ordinate and control. This lead to a view of managers as agents systematically assessing all relevant factors to generate some optimum plan. Sometime in the sixties, it was realised that electronic computers could automatically and routinely generate the information required for the Fayol model. This lead to the era of the management information system (MIS) and the wastage of a lot of paper. Managers found themselves overloaded with more information than they could manage.

Mintzberg's classic empirical fly-on-the-wall tracking of managers in the day-to-day work demonstrated that the Fayol model was normative, rather than descriptive. For example, a study of 56 U.S. foreman found that they averaged 583 activities in an eight-hour shift (one every 48 seconds). Another study of 160 British middle and top managers found that they worked for half an hour or more without interruption only once every two days [14]. This frantic pace for decision making does not match with Fayol's model of managers as systematic agents.

The lesson of MIS was that management decision making was not inhibited by a lack of information. Rather, it is confused by an excess of irrelevant information [1]. Modern decision-support systems (DSS) evolved to filter useless information to deliver relevant information (a subset of all information) to the manager. Our preferred definition of a decision-support system is based on Brookes who developed it from Mintzberg's model [3]. The goal of a DSS is *management comfort*, i.e. a subjective impression that all problems are known and under control. More specifically, managers need to seek out problems, solve them, then install some monitoring routine to check that the fix works. A taxonomy of tasks used in that process is shown in Figure 1

Other DSS workers have a similar view. Boose, Bradshaw, Koszaek, and Shema (BBKS) [2] discuss DSS architectures suitable for groups. Portions of the BBKS and the Brookes' models overlap. The BBKS system lets groups manipulate their group model, its inter-relationships, and the group's criteria for selecting the best alternative. BBKS stress that:

> The process of generating and scoring alternatives are at the heart of most decision processes. [2]

In the typical business situation, this process occurs domains containing much guess work. We have previously characterised [23] such *vague domains* as being:

- *Poorly measure*: i.e. known data from that domain is insufficient to confirm or deny that some inferred state is valid;
- *Hypothetical*: i.e. the domain lacks an authorative oracle that can declare knowledge to be "right" or "wrong". Note that in a well-measured domain, the authorative oracle could be a database of measurements.
- *Indeterminate*: i.e. inferencing over a knowledge base could generate numerous, mutually exclusive, outcomes. For example, consider the qualitative model [17] of Figure 2. In that figure:

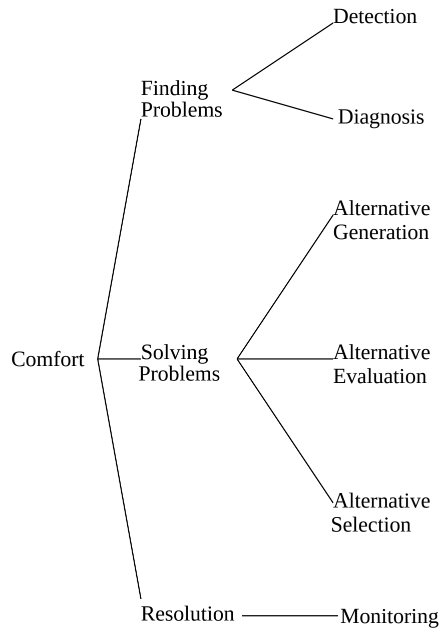

Figure 1: Components of management comfort

- $X \stackrel{++}{\rightarrow} Y$ denotes that Y being *UP* or *DOWN* could be explained by X being *UP* or *DOWN* respectively;
- $X \stackrel{--}{\rightarrow} Y$ denotes that Y being *UP* or *DOWN* could be explained by X being *DOWN* or *UP* respectively.

Note that the results of this model may be uncertain; i.e. it is indeterminate. In the case of both A and B going *UP*, then we have two competing influences of C and it is indeterminate whether C goes *UP*, *DOWN*, or remains *STEADY*.

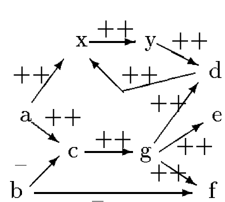

Figure 2: An indeterminate qualitative model.

Models developed for vague domains have two properties. Firstly, inference requires making guesses or assumptions and mutually exclusive assumptions must be managed separately. Secondly, since vague domains lack an authorative oracle, their models may be widely inaccurate. Modeling in vague domains therefore requires a validation engine. We have argued previously that such a validation engine should not be based on internal syntactic criteria (e.g. detection of subsumption or loops) [25]. We know of examples of working expert systems that contain these anomalies, yet still satisfy their day-to-day operational criteria [36]. We argue that the definitive test for any model is "can a model of X reproduce known behaviour of X". That is, external test suite validation is more important that internal syntactic verification.

We define an *intelligent decision support system* as a model-based software device that can execute in vague domains to support both external test suite validation and the tasks of Figure 1. Note that, by this definition, an IDSS must manage the assumption space of a problem.

3 Abduction

In this section, we discuss an inference procedure called abduction. In the next section we will argue that this procedure can implement IDSS.

Informally, abduction is inference to the best explanation [29]. Given α , β, and the rule $R_1 : \alpha \vdash \beta$, then *deduction* is using the rule and its preconditions to make a conclusion ($\alpha \wedge R_1 \Rightarrow \beta$); *induction* is learning R_1 after seeing numerous examples of β and α; and *abduction* is using the postcondition and the rule to assume that the precondition could be true ($\beta \wedge R_1 \Rightarrow \alpha$) [20].

More formally, abduction is the search for assumptions $\mathcal{A}$ which, when combined with some theory $\mathcal{T}$ achieves some goal $\mathcal{G}$ without causing some contradiction [8]. That is:

EQ_1: $\mathcal{T} \cup \mathcal{A} \vdash \mathcal{G}$

EQ_2: $\mathcal{T} \cup \mathcal{A} \not\vdash \perp$

In order to understand abduction in more detail, we describe our HT4 abductive inference engine [25]. To execute HT4, the user must supply a theory $\mathcal{T}$ comprising a set of uniquely labeled statements $\mathcal{S}_x$. For example, from Figure 2, we could say that:

```
s[1] = plus_plus(a,b).
s[2] = minus_minus(b,c).
etc.
```

The dependency graph $\mathcal{D}$ connecting literals in $\mathcal{T}$ is an and-or graph comprising $<< \mathcal{V}^{and}, \mathcal{V}^{or} >, \mathcal{E}, \mathcal{I} >$; i.e. a set of directed edges $\mathcal{E}$ connecting vertices $\mathcal{V}$ containing invariants $\mathcal{I}$. $\mathcal{I}$ is defined in the negative; i.e. $\neg\mathcal{I}$ means that no invariant violation has occurred (e.g. $\neg\mathcal{I}(p, \neg p)$). Each edge $\mathcal{E}_x$ and vertex $\mathcal{V}_y$ is labeled with the $\mathcal{S}_z$ that generated it.

For example, returning to the theory $\mathcal{T}$ of Figure 2, let us assume that (i) each node of that figure can take the value *UP*, *DOWN*, or *STEADY*; (ii) the conjunction of an *UP* and a *DOWN* can explain a *STEADY*; and (iii) no change can be explained in terms of a *STEADY* (i.e. a *STEADY* vertex has no children). With these assumptions, we can expand Figure 2 into Figure 3. In that figure, $\mathcal{V}^{and}$ vertices are denoted (e.g.) *&002* while all other vertices are $\mathcal{V}^{or}$ vertices. Note that in practice, the assumptions used to convert $\mathcal{T}$ into $\mathcal{D}$ are contained in a domain-specific *model-compiler*.

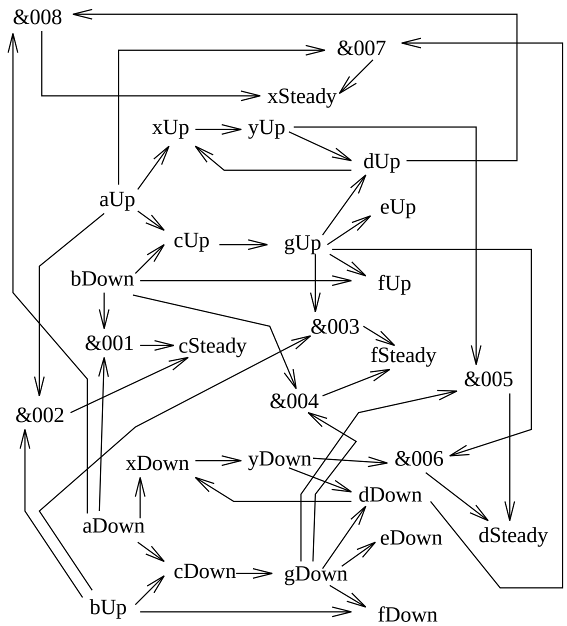

Figure 3: $\mathcal{D}$ calculated from the $\mathcal{T}$ of Figure 2

Not shown in Figure 3 are the invariants. For a qualitative domain, where nodes can have one of a finite number of mutually exclusive values, the invariants are merely all pairs of mutually exclusive assignments; e.g.:

```
i(aUp, aSteady). i(aSteady, aUp).
i(aUp, aDown).   i(aDown,   aUp).
i(bUp, bSteady). i(bSteady, bUp).
i(bUp, bDown).   i(bDown,   bUp).
etc.
```

HT4 extracts subsets of $\mathcal{E}$ which are relevant to some user-supplied $\mathcal{TASK}$. Each $\mathcal{TASK}_x$ is a triple $< \mathcal{IN}, \mathcal{OUT}, \mathcal{BEST} >$. Each task comprises some $\mathcal{OUT}$puts to be reached, given some $\mathcal{IN}$put ($\mathcal{OUT} \subseteq \mathcal{V}$ and $\mathcal{IN} \subseteq \mathcal{V}$). For the rest of this paper we will explore the example where:

```
IN  = {aUp, bUp}
OUT = {dUp, eUp, fDown}
```

$\mathcal{IN}$ can be either be a member of the known $\mathcal{FACTS}$ or a $\mathcal{DEFAULT}$ belief which we can assume if it proves convenient to do so. Typically, $\mathcal{FACTS} = \mathcal{IN} \cup \mathcal{OUT}$. If there is more than one way to achieve the $\mathcal{TASK}$, then the $\mathcal{BEST}$ operator selects the preferred way(s).

To reach a particular output $\mathcal{OUT}_z \in \mathcal{OUT}$, we must find a proof tree $\mathcal{P}_x$ using vertices $\mathcal{P}_x^{used}$ whose single leaf is $\mathcal{OUT}_z$ and whose roots are from $\mathcal{IN}$ (denoted $\mathcal{P}_x^{roots} \subseteq \mathcal{IN}$). All immediate parent vertices of all $\mathcal{V}_y^{and} \in \mathcal{P}_x^{used}$ must also appear in $\mathcal{P}_x^{used}$. One parent of all $\mathcal{V}_y^{or} \in \mathcal{P}_x^{used}$ must also appear in $\mathcal{P}_x^{used}$ *unless* $\mathcal{V}_y^{or} \in \mathcal{IN}$ (i.e. is an acceptable root of a proof). No subset of $\mathcal{P}_x^{used}$ may contradict the $\mathcal{FACTS}$; e.g. for invariants of arity 2:

$$\neg(\mathcal{V}_y \ \in \ \mathcal{P}_x^{used} \ \wedge \ \mathcal{V}_z \ \in \ \mathcal{FACTS} \ \wedge \ \mathcal{I}(\mathcal{V}_y, \mathcal{V}_z))$$

For our example, the proofs are:

```
p(1) = {aUp, xUp,   yUp,   dUp}
p(2) = {aUp, cUp,   gUp,   dUp}
p(3) = {aUp, cUp,   gUp,   eUp}
p(4) = {bUp, cDown, gDown, fDown}
p(5) = {bUp, fDown}
```

The union of the vertices used in all proofs that are not from the $\mathcal{FACTS}$ is the HT4 assumption set $\mathcal{A}_{all}$; i.e.

$$\mathcal{A}_{all} = \left(\bigcup_{\mathcal{V}_y} \left\{ \mathcal{V}_y \ \in \ \mathcal{P}_x^{used} \right\} \right) - \mathcal{FACTS}$$

The proofs in our example makes the assumptions:

```
a = {xUp, yUp, cUp, gUp, cDown, gDown}
```

The union of the subsets of $\mathcal{A}_{all}$ which violate $\mathcal{I}$ are the *controversial assumptions* $\mathcal{A}_C$:

$$\mathcal{A}_C = \bigcup_{\mathcal{V}_x} \{\mathcal{V}_x \in \mathcal{A}_{all} \wedge \mathcal{V}_y \in \mathcal{A}_{all} \wedge \mathcal{I}(\mathcal{V}_x, \mathcal{V}_y)\}$$

The controversial assumptions of our example are:

```
ac = {cUp, gUp, cDown, gDown}
```

Within a proof $\mathcal{P}_y$ the *preconditions* for $\mathcal{V}_y \in \mathcal{P}_x^{used}$ are the transitive closure of all the parents of $\mathcal{V}_y$ in that proof. The *base controversial assumptions* ($\mathcal{A}_B$) are the controversial assumptions which have no controversial assumptions in their preconditions (i.e. are not downstream of any other controversial assumptions). The base controversial assumptions of our example are:

```
ab = {cUp, cDown}
```

Maximal consistent subsets of $\mathcal{P}$ (i.e. maximal with respect to size, consistent with respect to $\mathcal{I}$) are grouped together into worlds $\mathcal{W}$ ($\mathcal{W}_i \subseteq \mathcal{E}$). Each world $\mathcal{W}_i$ contains a consistent set of beliefs that are relevant to the $\mathcal{TASK}$. The union of the vertices used in the proofs of $\mathcal{W}_i$ is denoted $\mathcal{W}_i^{used}$. In terms of separating the proofs into worlds, $\mathcal{A}_B$ are the crucial assumptions. We call the maximal consistent subsets of $\mathcal{A}_B$ the *environments* $\mathcal{ENV}$ ($\mathcal{ENV}_i \subset \mathcal{A}_B \subseteq \mathcal{A}_C \subseteq \mathcal{A}_{all} \subseteq \mathcal{V}$). The environments of our example are:

```
env(1) = {cUp}
env(2) = {cDown}
```

The union of the proofs that do not contradict $\mathcal{ENV}_i$ is the world $\mathcal{W}_i$. In order to check for non-contradiction, we compute the exclusions set $\mathcal{X}$. $\mathcal{X}_i$ are the base controversial assumptions that are inconsistent with $\mathcal{ENV}_i$. The exclusions of our example are:

```
x(1) = {cDown}
x(2) = {cUp}
```

A proof $\mathcal{P}_j$ belongs in world $\mathcal{W}_i$ if it does not use any member of $\mathcal{X}_i$ (the excluded assumptions of that world); i.e.

$$\mathcal{W}_i = \bigcup_{\mathcal{P}_j} \left\{ \mathcal{P}_j^{used} \cap \mathcal{X}_i = \emptyset \right\}$$

Note that each proof can exist in multiple worlds. The worlds of our example are:

```
w(1) = {p(1), p(2), p(3), p(5)}
w(2) = {p(1), p(4), p(5)}
```

$\mathcal{W}_1$ is shown in Figure 4 and $\mathcal{W}_2$ is shown in Figure 5.

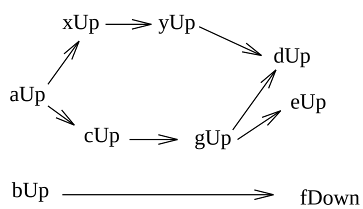

Figure 4: $\mathcal{W}_1$

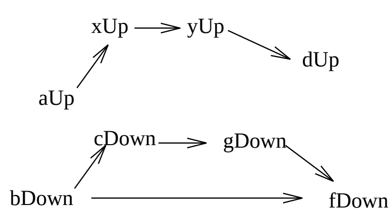

Figure 5: $\mathcal{W}_2$

For any world $\mathcal{W}_i$, $\mathcal{W}_i^{causes}$ are the members of $\mathcal{IN}$ found in $\mathcal{W}_i$ ($\mathcal{W}_i^{causes} = \mathcal{W}_i^{used} \cap \mathcal{IN}$). The achievable or *covered* goals $\mathcal{G}$ in $\mathcal{W}_i$ are the members of $\mathcal{OUT}$ found in that world ($\mathcal{W}_i^{covered} = \mathcal{W}_i^{used} \cap \mathcal{OUT}$). Continuing our example:

```
causes(w(1)) = {aUp, bUp}
causes(w(2)) = {aUp, bUp}

cover(w(1))  = {dUp, eUp, fDown}
cover(w(2))  = {dUp, fDown}
```

Note that, in our example, we have generated more than one world and we must now decide which world(s) we prefer. This is done using the $\mathcal{BEST}$ criteria. Numerous $\mathcal{BEST}$s can be found in the literature; e.g. the $\mathcal{BEST}$ worlds are the one which contain:

1. the most specific proofs (i.e. largest size) [12];
2. the fewest *causes* [37];
3. the greatest *cover* [23];
4. the most number of specific concepts [32];
5. the largest subset of $\mathcal{E}$ [28];
6. the largest number of *covered* outputs [27];
7. the most number of edges that model processes which are familiar to the user [31];
8. the most number of edges that have been used in prior acceptable solutions [19];

Our view is that $\mathcal{BEST}$ is domain specific; i.e. we believe that their is no best $\mathcal{BEST}$.

4 Using Abduction for IDSS

In order to satisfy our definition of an IDSS, abduction must be able to support validation, problem detection, diagnosis, alternative generation and assessment, and monitoring in vague domains. In this section, we argue that this is indeed the case.

HT4 can run with a minimum of information about the model at hand. During execution, mutually exclusive assumptions are maintained in separate worlds. HT4 was originally developed as a validation algorithm for external test suite assessment. Given a library of known behaviours (i.e. a set of pairs $< \mathcal{IN}, \mathcal{OUT} >$), abductive validation uses a $\mathcal{BEST}$ that favours the worlds with largest number of covered outputs (i.e. maximise $\mathcal{IN} \cap \mathcal{W}_x$) [27]. Returning to our above example, we see that there exists a world `w(1)` whose cover is all of $\mathcal{OUT}$; i.e. there exists a set of assumptions by which all of the known behaviour can be explained. Since HT4 can handle validation and inference in poorly measured, indeterminate domains overly possible incorrect model; it is suitable for vague domains and IDSS.

IDSS diagnosis can be implemented using either *set-covering* or *consistency-based* diagnosis. Parsimonious *set-covering diagnosis* [37] uses a $\mathcal{BEST}$ that favors worlds that explain the most things, with the smallest number of diseases (i.e. maximise $\mathcal{W}_x \cap \mathcal{OUT}$ and minimise $\mathcal{W}_x \cap \mathcal{IN}$). Set-covering diagnosis is best for fault models and causal reasoning [18]. Set-covering diagnosis could implement IDSS detection. Experts can add alarm points into the model. If we can reach certain vertices (the alarm points), then the system can detect potential problems.

The opposite of set-covering diagnosis is *consistency-based diagnosis* [5, 7, 13, 33, 38] in which all worlds consistent with the current observations are generated. In this abductive framework, this is implemented by calling HT4 with $\mathcal{OUT} \subseteq \mathcal{V} - \mathcal{IN}$; i.e. find all vertices we can reach from the inputs. The $\mathcal{FACTS}$ is restricted to being empty (i.e. all vertices are possible) or just $\mathcal{IN}$ (i.e. only the inputs cannot be contradicted). This is a non-naive implementation of prediction since mutually exclusive predictions will be found in different worlds. Note that in the special case where:

- $\mathcal{IN}$ are all root vertices in the graph
- $\mathcal{FACTS} = \emptyset$
- $\mathcal{OUT} = \mathcal{V} - \mathcal{IN}$

then our abductive system will compute ATMS-style [6] total envisionments; i.e. all possible consistent worlds that are extractable from the theory. A more efficient case is that $\mathcal{IN}$ is smaller than all the roots of the graph and some *interesting subset* of the vertices have been identified as possible reportable outputs (i.e. $\mathcal{OUT} \subset \mathcal{V} - \mathcal{IN}$). This process can be varied slightly. For example, in Reiter's variant on consistency-based diagnosis [38], all predicates relating to the behaviour of a model component $\mathcal{V}_x$ assume a test that $\mathcal{V}_x$ in not acting $\mathcal{AB}$normally; i.e. $\neg\mathcal{AB}(\mathcal{V}_x)$. $\mathcal{BEST}_{Reiter}$ is to favour the worlds that contain the least number of $\mathcal{AB}$ assumptions.

IDSS assessment generation and selection is merely the world generation and selection process described above. Experts can specify their preference criteria using $\mathcal{BEST}$.

Lastly, we can use abductive for IDSS monitoring. In the case where $\mathcal{BEST}$ returns us N worlds, we can pass these worlds to a monitoring process which reviews the possible worlds as new data comes to light. Worlds that use literals which are inconsistent with new data are rejected. The remaining worlds represent the space of possible ways to achieve the desired goals in the current situation. If all worlds are rejected, then HT4 is run again using all the available data.

5 Practicality

Abduction has a reputation of being impractically slow [8]. Selman & Levesque show that even when only one abductive explanation is required and $\mathcal{T}$ is restricted to an acyclic theories, then abduction is NP-hard [39]. Bylander *et. al.* make a similar pessimistic conclusion [4].

In practice these theoretical restrictions may not limit application development. The core computational problem of HT4 is the search for $\mathcal{X}_i$. Earlier versions of HT4 [10, 11, 22] computed the $\mathcal{BEST}$ worlds $\mathcal{W}$ via a basic depth-first search chronological backtracking algorithm (DFS) with no memoing. These systems took days to terminate [25]. Mackworth [21] and DeKleer [6] warn that DFS can learn features of a search space, then forget it on backtracking. Hence, it may be doomed to waste time re-learning those features later on. One alternative to chronological backtracking is an algorithm that caches what it learns about the search space as it executes. HT4 runs in four "sweeps" which learn and cache features of the search space as it executes. For details, see [24, 25]. In experiments with 94 models run 1991 times, HT4 proved to be practical for models of up to 800 vertices in $\mathcal{D}$ [25, 27].

In those runtime experiments, a *worlds-level* $\mathcal{BEST}$ was used for worlds assessment. Such *worlds-level* $\mathcal{BEST}$ operators have the drawback that they cannot be used by a local propagation algorithm to cull the search space. There is no reason why certain $\mathcal{BEST}$s could not applied earlier; e.g. during proof generation. For example, if it is known that $\mathcal{BEST}$ will favour the worlds with smallest path sizes between inputs and goals, then a beam-search style $\mathcal{BEST}$ operator could cull excessively long proofs within the generation process.

More generally, we characterise $\mathcal{BEST}$s into the information they require before they can run:

- *Vertex-level* assessment operators can execute at the local-propagation level; e.g. use the edges with the highest probability.
- *Proof-level* assessment operators can execute when some proofs or partial proofs are known; e.g. beam search. Ng & Mooney report reasonable runtimes for their abductive system using a beam-search proof-level assessment operator [28].
- Favoring the world(s) that cover (e.g.) the greatest number of outputs (i.e. the validation process described above) is a *worlds-level assessment operator* which cannot execute till all the worlds are generated.

While the complexity of $\mathcal{BEST}$ is operator specific, we can make some general statements about the computational cost of $\mathcal{BEST}$. *Vertex* or *proof-level* assessment reduce the complexity of proof generation (since not all paths are explored). *Worlds-level* assessment is a search through the entire space that could be relevant to a certain task. Hence, for fast runtimes, do not use worlds-level assessment. However, for some tasks (e.g. the validation task), worlds-level assessment is unavoidable.

6 Conclusion

A single inference procedure (abduction) can support many of the sub-routines required for an intelligent decision support system; i.e. validation, detection, diagnosis, alternative generation and assessment, and monitoring. Hence, we propose the use of abduction as a framework for IDSS.

References

[1] R.L. Ackoff. Management Misinformation Systems. *Management Science*, pages 319–331, December 1967.

[2] J.H. Boose, J.M. Bradshaw, J.L. Koszareck, and D.B. Shema. Knowledge Acquisition Techniques for Group Decision Support. In B.R. Gaines, M.A. Musen, and J.R. Boose, editors, *Proceedings of the 7th Knowledge Acquisition for Knowledge-Based Systems Workshop*, pages 2.1–2.22, 1992.

[3] C.H.P. Brookes. Requirements Elicitation for Knowledge Based Decision Support Systems. Technical Report 11, Information Systems, University of New South Wales, 1986.

[4] T. Bylander, D. Allemang, M.C. M.C. Tanner, and J.R. Josephson. The Computational Complexity of Abduction. *Artificial Intelligence*, 49:25–60, 1991.

[5] L. Console and P. Torasso. A Spectrum of Definitions of Model-Based Diagnosis. *Computational Intelligence*, 7:133–141, 3 1991.

[6] J. DeKleer. An Assumption-Based TMS. *Artificial Intelligence*, 28:163–196, 1986.

[7] J. DeKleer and B.C. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32:97–130, 1 1987.

[8] K. Eshghi. A Tractable Class of Abductive Problems. In *IJCAI '93*, volume 1, pages 3–8, 1993.

[9] B Falkenhainer. Abduction as Similarity-Driven Explanation. In P. O'Rourke, editor, *Working Notes of the 1990 Spring Symposium on Automated Abduction*, pages 135–139, 1990.

[10] B.Z. Feldman, P.J. Compton, and G.A. Smythe. Hypothesis Testing: an Appropriate Task for Knowledge-Based Systems. In *4th AAAI-Sponsored Knowledge Acquisition for Knowledge-based Systems Workshop Banff, Canada, October 1989*, 1989.

[11] B.Z. Feldman, P.J. Compton, and G.A. Smythe. Towards Hypothesis Testing: JUSTIN, Prototype System Using Justification in Context. In *Proceedings of the Joint Australian Conference on Artificial Intelligence, AI '89*, pages 319–331, 1989.

[12] C.L. Forgy. RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, pages 17–37, 19 1982.

[13] M.R. Genesereth. The Use of Design Descriptions in Automated Diagnosis. *Artificial Intelligence*, 24:411–436, 1984.

[14] Mintzberg H. The Manager's Job: Folklore and Fact. *Harvard Business Review*, pages 29–61, July-August 1975.

[15] W. Hamscher. Explaining Unexpected Financial Results. In P. O'Rourke, editor, *AAAI Spring Symposium on Automated Abduction*, pages 96–100, 1990.

[16] K. Hirata. A Classification of Abduction: Abduction for Logic Programming. In *Proceedings of the Fourteenth International Machine Learning Workshop, ML-14*, page 16, 1994.

[17] Y. Iwasaki. Qualitative physics. In P.R. Cohen A. Barr and E.A. Feigenbaum, editors, *The Handbook of Artificial Intelligence*, volume 4, pages 323–413. Addison Wesley, 1989.

[18] K. Konoligue. Abduction versus Closure in Causal Theories. *Artificial Intelligence*, 53:255–272, 1992.

[19] D.B. Leake. Focusing Construction and Selection of Abductive Hypotheses. In *IJCAI '93*, pages 24–29, 1993.

[20] H. Levesque. A Knowledge-Level Account of Abduction (Preliminary Version). In *IJCAI '89*, volume 2, pages 1061–1067, 1989.

[21] A.K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8:99–118, 1977.

[22] T. Menzies, A. Mahidadia, and P. Compton. Using Causality as a Generic Knowledge Representation, or Why and How Centralised Knowledge Servers Can Use Causality. In *Proceedings of the 7th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop Banff, Canada, October 11-16*, 1992.

[23] T. J. Menzies and P. Compton. The (Extensive) Implications of Evaluation on the Development of Knowledge-Based Systems. In *Proceedings of the 9th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge Based Systems*, 1995. Available from http://www.sd.monash.edu.au/ ~timm/pub/ docs/papersonly.html.

[24] T.J. Menzies. An Overview of Abduction as a General Framework for Knowledge-Based Systems. Technical Report TF95-5, Department of Software Development, Monash University, Caulfield East, Melbourne, Australia, 3145, 1995. Available from http://www.sd.monash.edu.au/ ~timm/pub/ docs/papersonly.html.

[25] T.J. Menzies. *Principles for Generalised Testing of Knowledge Bases*. PhD thesis, University of New South Wales, 1995.

[26] T.J. Menzies. Situated Semantics is a Side-Effect of the Computational Complexity of Abduction. In *Australian Cognitive Science Society, 3rd Conference*, 1995. Available from http://www.sd.monash.edu.au/ ~timm/pub/ docs/papersonly.html.

[27] T.J. Menzies and W. Gambetta. Exhaustive Abduction: A Practical Model Validation Tool. In *ECAI '94 Workshop on Validation of Knowledge-Based Systems*, 1994. Available from http://www.sd.monash.edu.au/ ~timm/pub/ docs/papersonly.html.

[28] H.T. Ng and R.J. Mooney. The Role of Coherence in Constructing and Evaluating Abductive Explanations. In *Working Notes of the 1990 Spring Symposium on Automated Abduction*, volume TR 90-32, pages 13–17, 1990.

[29] P. O'Rourke. Working notes of the 1990 spring symposium on automated abduction. Technical Report 90-32, University of California, Irvine, CA., 1990. September 27, 1990.

[30] M. Pagnucco, A.C. Nayak, and N.Y. Foo. Abductive Expansion: Abductive Inference and the Process of Belief Change. In J. Debenham C. Zhang and D. Lukose, editors, *AI '94, Australia*, 1994.

[31] C.L. Paris. The Use of Explicit User Models in a Generation System for Tailoring Answers to the User's Level of Expertise. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 200–232. Springer-Verlag, 1989.

[32] D. Poole. On the Comparison of Theories: Preferring the Most Specific Explanation. In *IJCAI '85*, pages 144–147, 1985.

[33] D. Poole. Normality and Faults in Logic-Based Diagnosis. In *IJCAI '89*, pages 1304–1310, 1989.

[34] D. Poole. Hypo-Deductive Reasoning for Abduction, Default Reasoning, and Design. In P. O'Rourke, editor, *Working Notes of the 1990 Spring Symposium on Automated Abduction.*, volume TR 90-32, pages 106–110, 1990.

[35] D. Poole. A Methodology for Using a Default and Abductive Reasoning System. *International Journal of Intelligent Systems*, 5:521–548, 1990.

[36] A.D. Preece and R. Shinghal. Verifying knowledge bases by anomaly detection: An experience report. In *ECAI '92*, 1992.

[37] J. Reggia, D.S. Nau, and P.Y Wang. Diagnostic Expert Systems Based on a Set Covering Model. *Int. J. of Man-Machine Studies*, 19(5):437–460, 1983.

[38] R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32:57–96, 1 1987.

[39] B. Selman and H.J. Levesque. Abductive and Default Reasoning: a Computational Core. In *AAAI '90*, pages 343–348, 1990.