# Applications of Abduction: Knowledge-Level Modeling

Tim Menzies *

November 3, 1996

## Abstract

A single inference procedure (abduction) can operationalise a wide variety of knowledge-level modeling problem solving methods; i.e. prediction, classification, explanation, tutoring, qualitative reasoning, planning, monitoring, set-covering diagnosis, consistency-based diagnosis, validation, and verification. This abductive approach offers a uniform view of different problem solving methods in the style proposed by Clancey and Breuker. Also, this abductive approach is easily extensible to validation; i.e. using this technique we can implement both inference tools and testing tools. Further, abduction can execute in vague and conflicting domains (which we believe occur very frequently). We therefore propose abduction as a framework for knowledge-level modeling.

## 1 Introduction

In the 1970s and early 1980s, several high-profile expert system successes were documented: e.g. MYCIN [86], CASNET [82], PROSPECTOR [7, 20], and XCON [1]. However, despite careful attempts to generalise this work (e.g. [78]), expert systems construction remains a somewhat hit-and-miss process. By the end of the 1980s, it was recognised that our design concepts for knowledge-based systems were incomplete [5].

A new expert system design approach (which has come to dominate the knowledge acquisition field) is the the search for reusable abstract domain-independent problem-solving strategies. We call this approach $\mathcal{KL}_B$ since it is a variant of Newell's *knowledge level* (KL) modeling approach [46, 53, 55, 54]. The fundamental premise of $\mathcal{KL}_B$ is that a knowledge base should be divided into domain-specific facts and domain-independent abstract problem solving inference procedures (e.g. Clancey's model construction operators [12], Steels' components of expertise [77], Chandrasekaran's task analysis [8], SPARK/ BURN/ FIREFIGHTER [42] and KADS [84]). $\mathcal{KL}_A$ refers to Newell's research on the knowledge level [53, 55, 54] and the SOAR project [73, 85]. $\mathcal{KL}_A$ does not explicitly model problem-solving procedures. The observation that a $\mathcal{KL}_A$ system such as SOAR is performing (e.g.) classification is a user-interpretation of

1. The application of domain-specific knowledge controlling...
2. ... a single inference procedure (operator selection over a problem space traversal) [85].

This paper argues for a variant on the $\mathcal{KL}_A$ approach. Like $\mathcal{KL}_A$ , we will use a single inference procedure (abduction). However, we take a graph-theoretic approach rather than the production-system approach used by SOAR (see section 6.3. for a comparison of our approach and SOAR). We find that a wide-variety of problem solving strategies are merely different types of calls to the same abduction procedure. Such uniformity simplifies the construction of interfaces between the inputs and outputs of different problem solving types. Breuker argues that such interfacing is essential since most problem solving types are used in combination to perform some task [4].

Far from being radical proposal, we find that our abductive process directly operationalises the *theory subset extraction* process that Breuker [4] and Clancey [11, 12] argue is at the core of expert systems. Clancey offers a two-layered extraction process (qualitative model to situation-specific model) while Breuker offers a four-layered view (generic

*Address*: Dept. of Software Development, Monash University, Caulfield East, Melbourne, Melbourne, Australia, 3185. *email*: timm@insect.sd.monash.edu.au.
*Past papers* available from http://www.sd.monash.edu.au/~timm/pub/ docs/papersonly.html. This paper is Dept. of Software Engineering technical report TR95-23. WpRef: 95/abkl

| | |
|---|---|
| $\|X\|, X \cap Y$ | Size of the set $X$, the intersection of the sets $X$ and $Y$ |
| $\mathcal{S}$ | A statement provided by an expert. |
| $\mathcal{T}$ | A theory comprising a set of statements; e.g. Figures 2 & 9. |
| $\mathcal{D}$ | A dependency graph showing connections between literals in $\mathcal{T}$; e.g. Figures 3 & 10. |
| $< \mathcal{V}, \mathcal{E} >$ | Vertices and edges in $\mathcal{D}$. Vertices are either and-vertices $\mathcal{V}^{and}$ or or-vertices $\mathcal{V}^{or}$. |
| $\mathcal{I}$ | An invariants predicate reporting pairs of incompatible vertices in $\mathcal{D}$. |
| $\mathcal{NOGOODS}$ | Sets of incompatible vertices; generated using $\mathcal{I}$. |
| *model compiler* | A translator from $\mathcal{T}$ to $\mathcal{D}$. |
| $\mathcal{F}$ | The fanout of $\mathcal{D}$; i.e. average number of edges from a vertex. $\mathcal{F} = \frac{\|\mathcal{E}\|}{\|\mathcal{V}\|}$. |
| $\mathcal{OUT}$ | The subset of $\mathcal{V}$ we are trying to explain. |
| $\mathcal{IN}$ | The subset of $\mathcal{V}$ which are acceptable starting-points of an explanation. |
| $\mathcal{FACTS}$ | Vertices we cannot doubt. |
| $\mathcal{DEFAULTS}$ | $\mathcal{IN}$ vertices that are not $\mathcal{FACTS}$. |
| $\mathcal{P}$ | Proof trees connecting $\mathcal{OUT}$ to $\mathcal{IN}$. Each proof $\mathcal{P}_i$ using vertices $\mathcal{V}_i^{used}$, and avoids the vertices $\mathcal{V}_i^{forbid}$. |
| $\mathcal{A}$ | Assumptions made by $\mathcal{P}$; i.e. $\mathcal{P}_i^{used} - \mathcal{FACTS}$. |
| $\mathcal{A}_C$ | Assumptions which $\mathcal{I}$ tells us are contradictory. |
| $\mathcal{A}_B$ | The most upstream controversial assumptions. |
| $\mathcal{ENV}$ | Maximal (with respect to size) consistent (defined using $\mathcal{I}$) subsets of $\mathcal{A}_B$. |
| $\mathcal{W}_i$ | A world: the set of proofs that are consistent with $\mathcal{ENV}_i$; e.g. Figures 4, & 5. |
| *cover, causes* | Outputs and inputs in a world. $cover = \|\mathcal{OUT} \cap \mathcal{W}_i\|$; $causes = \|\mathcal{IN} \cap \mathcal{W}_i\|$. |
| $\mathcal{BEST}$ | Competing worlds are judged by the $\mathcal{BEST}$ assessment operator. |
| $\mathcal{TASK}$ | The goal of an inference procedure: $\mathcal{TASK} = < \mathcal{BEST}, \mathcal{IN}, \mathcal{OUT} >$ |

Figure 1: Summary of terms.

domain model to case model to conclusion to argument structure). We take theory subset extraction to be a literal description of the internals of expert systems inference. Our research goal is the description of the minimal architecture necessary to perform this process.

This paper is organised as follows. A summary of the terms introduced in this article is given in Figure 1. Section 2 describes the *theory subset extraction* described by Clancey and Breuker. Section 3 describes our *abductive* framework. Section 4 discusses the use of abduction for a variety of $\mathcal{KL}_B$ tasks; i.e. prediction, classification, explanation, tutoring, qualitative reasoning, planning, monitoring, set-covering diagnosis, consistency-based diagnosis, validation, and verification. Section 5 discusses the practicality of our proposal. Section 6 discusses some related work and issues.

Note that this work is part of our *abductive reasoning project*. We believe that abduction provides a comprehensive picture of declarative knowledge-based systems (KBS) inference. Apart from the problem solving methods discussed here, we also believe that abduction is a useful framework for intelligent decision support systems [44], diagrammatic reasoning [51], single-user knowledge acquisition, and multiple-expert knowledge acquisition [48]. Further, abduction could model certain interesting features of human cognition [49]. Others argue elsewhere that abduction is also a framework for natural-language processing [56], design [61], visual pattern recognition [62], analogical reasoning [24], financial reasoning [32], machine learning [33] and case-based reasoning [39].

# 2 Clancey & Breuker

In this section, we argue that the common theme between Clancey's and Breuker's view of expert systems inference is the extraction of a sub-theory from a super-theory.

## 2.1 Model Construction Operators

Clancey characterises expert system inference as model construction operators that create a *situation-specific model* (SSM) from a general *qualitative model* (QM) in the knowledge base (KB). Clancey's QM is like a first-order theory whose relations model causality, sub-types, and temporal relations. At runtime, portions of this theory are accessed and the variables are bound. This ground subset of the full theory is the SSM; i.e. "the specific model the program is constructing of the particular system it is (processing)" [12]. This specific model is the subset of the QM that is relevant to the task at hand.

Clancey argues that there are two basic problem-solving methods used by expert systems: *heuristic classification* and *heuristic construction* [12]. By heuristic classification, Clancey means that the inference engine merely *selects* a pre-existing inference path. In heuristic classification, this pathway would include:

- Inference to an abstracted description of the problem at hand;
- A partial match of this problem to an abstracted solution;
- An inference that specialises the abstracted solution to a solution relevant to the current problem.

By heuristic construction, Clancey means that the inference engine *constructs* its conclusions from partial inferences supplied in the knowledge base. Construction is much harder than mere selection. Literals in different partial proofs may be mutually exclusive; i.e. while we can believe $A \vee B$, it may not be true that we can believe $A \wedge B$. The constructed SSM must be built with care in order to take into account these *cancelation interactions*. Multiple, mutually exclusive, SSMs may be possible and these must be managed separately. Extra architecture is required to handle conflicts and dependencies within the SSM.

## 2.2 Components of Solutions

Breuker explores the relationships between problem solving techniques used in expert systems (i.e. modeling, planning, design, assignment, prediction, assessment, monitoring and diagnosis) [4]. He offers an abstract description of the "components of a solution" generated by these techniques which, he argues, are of four types:

- A *case model* (equivalent to Clancey's SSM) that represents some understanding of a problem;
- A *conclusion*, which is some answer to a question posed by the problem definition;
- An *argument structure*, which is supporting evidence for the conclusion generated.
- The case model which is generated from some *generic domain model* (equivalent to Clancey's QM).

An argument structure is extracted from the case model. The conclusion is the portion of an argument structure that is relevant to the user. In the case where all the solution components are represented as a ground propositional theory whose dependency graph has edges $\mathcal{E}$, then:

$$edges(answer) \subseteq$$
$$edges(argument\ structure) \subseteq$$
$$edges(case\ model) \subseteq$$
$$(edges(generic\ domain\ model) = \mathcal{E})$$

where *edges(X)* denotes the edges of the dependency graph present in $X$.

## 2.3 Theory Subset Extraction: an Example

We now describe theory subset extraction in detail using a theory of vertices $\mathcal{V}$ and edges $\mathcal{E}$. This example will informally introduce many of the concepts we will return to later. In summary, we will search our theory for a subset of its edges that are relevant to some problem. The found subset must be internally consistent (i.e. we have to check for cancelation effects between mutually exclusive assumptions).

Consider the qualitative theory [35] of Figure 2.

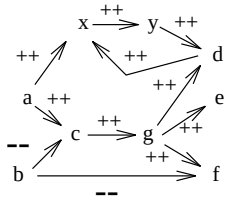


Figure 2: An indeterminate qualitative theory.

In that figure:

- All vertices can take one of three values: *UP*, *DOWN*, or *STEADY*.
- $X \stackrel{++}{\rightarrow} Y$ 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.

Let us make some qualitative reasoning assumptions about Figure 2:

- The conjunction of an *UP* and a *DOWN* can explain a *STEADY*;
- 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. That figure contains one vertex for each possible state of the vertices of Figure 2. It also contains *and* vertices that models combinations of influences (for example, `aUp` and `bUp` leads to `cSteady`).

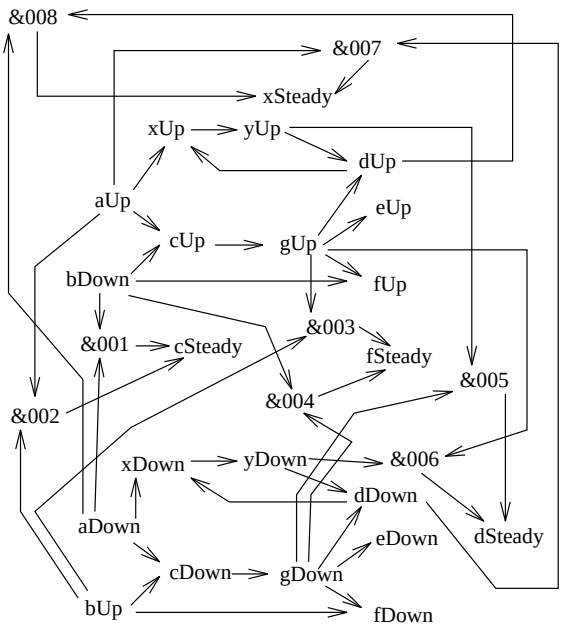


Figure 3: The edges $\mathcal{E}$ tacit in Figure 2.

Figure 3 represents the superset of all explanations possible from Figure 2; i.e. it is an explicit ground version of Clancey's QM and Breuker's generic domain model. Given some inputs (denoted $\mathcal{IN}$) and some desired goals (denoted $\mathcal{OUT}$), then we can use Figure 3 to generate a set of explanatory proofs (denoted $\mathcal{P}$). For example, if

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

then all the proofs which can link members of $\mathcal{IN}$ to members of $\mathcal{OUT}$ across Figure 3 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`

Some of these proofs are contradictory since that make conflicting *assumptions*. An assumption is a literal that is not one of the known $\mathcal{FACTS}$ (typically, $\mathcal{FACTS} = \mathcal{IN} \cup \mathcal{OUT}$). Our assumptions are {`xUp, yUp, cUp, gUp, cDown, gDown`}. If we assume that an entity can't be in two different states at the same time, then the following assumptions are conflicting and controversial: {`cUp, gUp, cDown, gDown`}. Note that, in Figure 2, `g` is fully determined by `c`. Therefore, in terms of sorting out the various possibilities, the key controversial assumptions are {`cUp`} or {`cDown`}.

Depending on which controversial assumptions we adopt, we can believe different things. In this example, we have two possibilities: one for {`cUp, dUp`} and one for {`cDown, dDown`}. The proofs that are consistent with {`cUp, dUp`} are $\{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_5\}$ and the proofs that are consistent with {`cDown, dDown`} are $\{\mathcal{P}_1, \mathcal{P}_4, \mathcal{P}_5\}$. The union of the proofs that we can believe at the same time are the Clancey SSM or the Breuker case model (we will call them *worlds* below). There are two such case models, shown in Figure 4 and Figure 5.

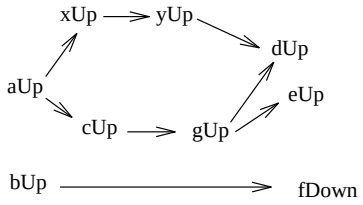


Figure 4: Case model #1: The union of proofs that are consistent with {`cUp, gUp`}.

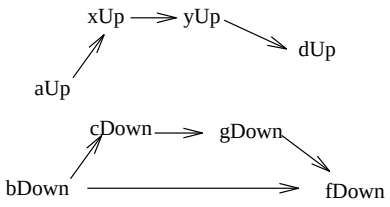


Figure 5: Case model #2: The union of proofs that are consistent with {`cDown, gDown`}.

Queries can be executed over each case model to

generate Breuker's argument structures or conclusions. For example, in case model#1 (Figure 4), an argument structure for the conclusion `dUp` could be `aUp` $\rightarrow$ `xUp` $\rightarrow$ `yUp` $\rightarrow$ `dUp`.

# 3 Abduction

We believe that abduction is a powerful framework for describing the above theory subset extraction process. In this section, we repeat the above example in terms of HT4 [48], our preferred abductive framework. In the next section, we will argue that many $\mathcal{KL}_B$ tasks are just different ways of calling HT4.

## 3.1 Definitions

Informally, abduction is typically defined as inference to the best explanation (e.g. [57]). Given $\alpha$ , $\beta$, 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 $\beta$ and $\alpha$; and *abduction* is using the postcondition and the rule to assume that the precondition could explain the postcondition $(\beta \wedge R_1 \Rightarrow \alpha)$ [40]. Abduction is not a certain inference and its results must be checked by an inference assessment operator (which we call $\mathcal{BEST}$ and Bylander *et. al.* [6] call the plausibility operator *pl*).

## 3.2 HT4 and Abduction

More formally, abduction is the search for assumptions $\mathcal{A}$ which, when combined with some theory $\mathcal{T}$ achieves some set of goals $\mathcal{OUT}$ without causing some contradiction [22]. That is:

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

$EQ_2$: $\mathcal{T} \cup \mathcal{A} \nvdash \perp$

While abduction can be used to generate explanation engines (see section 4.3), we believe that $EQ_1$ and $EQ_2$ are more than just a description of "inference to the best explanation". $EQ_1$ and $EQ_2$ can be summarised as follows: make what inferences you can that are relevant to some goal, without causing any contradictions. Note that the proof trees used to solve $EQ_1$ and $EQ_2$ are the case models/SSMs/worlds we seek to compute.

To execute HT4, the user must supply a theory $\mathcal{T}$ (e.g. $\mathcal{T}_1$ = Figure 2) comprising a set of uniquely labeled statements $\mathcal{S}_x$. For example, from Figure 2, we could say that:

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

An $\mathcal{S}_i$ statement is like a macro that expands into the super-set of explanations acceptable to the author of $\mathcal{S}_i$. This super-set is the search space for the proof generation. We represent this search space as a dependency graph $\mathcal{D}$. $\mathcal{D}$ is directed and possibly cyclic. Figure 3 shows $\mathcal{D}_1$, the dependency graph generated from $\mathcal{T}_1$. $\mathcal{D}$ 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. Each edge $\mathcal{E}_x$ and vertex $\mathcal{V}_y$ is labeled with the $\mathcal{S}_z$ that generated it. Figure 3 contains sample and-vertices and or-vertices. For example:

- `xUp` is an or-vertex which we can believe if we also believe `dUp` *or* `aUp`.
- `&003` is an and-vertex which we can believe if we also believe `gUp` *and* `bUp` (but see section 4.2 for alternative ways of handling and-vertices).

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

```
%i(X,Y): X and Y cannot be believed together
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.
```

## 3.3 The Model Compiler

When converting $\mathcal{T}_i$ to $\mathcal{D}_i$, a *model compiler* is required to capture any special domain semantics. For example, in a qualitative reasoning domain, we can reach a *STEADY* via a conjunction of two competing upstream influences (e.g. `&003`). In practice, these model compilers are very small. Our qualitative domain compiler is less than 100 lines of Smalltalk.

HT4-style inference is feasible for representations that support such a translator between $\mathcal{T}$ and $\mathcal{D}$. Recall that $\mathcal{D}$ is an explicit and-or graph of literals (positive or negative propositions) that represents the superset of explanations acceptable to

the author of $\mathcal{T}$. Such and-or graphs can be extracted from many representations including propositional expert systems and certain types of equational systems [36, 34]. HT4 could also be used for first-order theories, but only where that theory can be partially evaluated to an equivalent ground (i.e. no variables) theory.

Once such a model-compiler is available, then the practical limit to HT4 is the size of $\mathcal{D}$. These limits are explored further in Section 5.

### 3.4 Proofs of $\mathcal{OUT}$puts

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 $\mathcal{IN} \subseteq \mathcal{V}$). $\mathcal{IN}$ 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 and-vertices in a proof must also appear in that proof. One parent of all or-vertices in a proof must also appear in that proof *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:

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

### 3.5 Assumption Sets

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

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

Recall from the above that the proofs in our example made the assumptions:

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

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

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

The controversial assumptions of our example were:

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

The *base controversial assumptions* ($\mathcal{A}_B$) are the controversial assumptions which have no controversial assumptions in their ancestors (i.e. are not downstream of any other controversial assumptions). The base controversial assumptions of our example are:

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

### 3.6 World Generation

Maximal consistent subsets of $\mathcal{P}$ (i.e. maximal with respect to size, consistent with respect to $\mathcal{I}$) are grouped together into what we call worlds $\mathcal{W}$ ($\mathcal{W}_i \subseteq \mathcal{E}$) (recall that world $\equiv$ case model $\equiv$ SSM). 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* $\mathcal{ENV}$ ($\mathcal{ENV}_i \subseteq \mathcal{A}_B \subseteq \mathcal{A}_C \subseteq \mathcal{A} \subseteq \mathcal{V}$). 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 $\mathcal{W}_i$. One world is defined for each environment; i.e. $|\mathcal{W}| = |\mathcal{ENV}|$. In order to check for non-contradiction, we use $\mathcal{I}$ to find the vertices that are forbidden by each proof:

$$\mathcal{P}_j^{forbids} = \bigcup_{\mathcal{V}_l} \left\{ \mathcal{V}_k \in \mathcal{P}_j^{used} \wedge \mathcal{I}(\mathcal{V}_k, \mathcal{V}_l) \right\}$$

For example, $\mathcal{P}_5^{forbids}$ = {`bDown, bSteady, fUp, fSteady`}.

A proof $\mathcal{P}_j$ belongs in world $\mathcal{W}_i$ if its forbids set does not intersect with $\mathcal{ENV}_i$; i.e.:

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

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.

### 3.7 Assessing Worlds

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 $\mathcal{OUT}$ 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}

covered(w(1))  = {dUp, eUp, fDown}
covered(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. Clancey has a clear opinion on what is the $\mathcal{BEST}$ world:

> When there are multiple causal links for classifying data - multiple explanations-inference must be controlled to avoid redundancy, namely multiple explanations when one would have been sufficient. The aim is to produce a *coherent* model that is *complete* (accounting for the most data) and *simple* (involving one fault process) [11, p331]

Expressed in terms of HT4, Clancey's preferred $\mathcal{BEST}$ is to favour worlds that maximises the *covered* while minimising the *causes* (ideally, to a single cause). Numerous other $\mathcal{BEST}$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) [28];
2. the fewest *causes* [69];
3. the largest *covered* [47, 45];
4. the largest number of specific concepts [59];
5. the largest subset of $\mathcal{E}$ [56];
6. the largest number of edges that model processes which are familiar to the user [58];
7. the largest number of edges that have been used in prior acceptable solutions [39];

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

### 3.8 HT4 is Abduction

Given certain renamings, HT4 satisfies the definition of abduction given in Section 3.2 (see $EQ_1, EQ_2$). HT4-style abduction is the search for (i) a subset of $\mathcal{E}$ called $\mathcal{W}_i$, (ii) a subset of $\mathcal{IN}$ called $\mathcal{W}_i^{causes}$, (iii) a subset of $\mathcal{OUT}$ called $\mathcal{W}_i^{covered}$, and (iv) a subset of $\mathcal{V}$ called $\mathcal{ENV}_i$ such that:

$EQ_{1.1}$: $\mathcal{W}_i \wedge \mathcal{W}_i^{causes} \wedge \mathcal{ENV}_i \vdash \mathcal{W}_i^{covered}$

$EQ_{2.1}$: $\mathcal{W}_i \wedge \mathcal{W}_i^{causes} \wedge \mathcal{ENV}_i \wedge \mathcal{W}_i^{covered} \wedge \neg\mathcal{I}$

The assumptions $\mathcal{A}$ found by HT4 are the useful inputs ($\mathcal{W}_i^{causes}$), some assumptions about intermediaries between the useful inputs and the covered outputs ($\mathcal{ENV}_i$), and the edges *relevant* to a particular $\mathcal{TASK}$ ($\mathcal{W}_i^{used}$). In the case where multiple worlds can be generated, the $\mathcal{BEST}$ operator decides which world(s) to show to the user.

## 4 Applications of Abduction

This section argues that a wide variety of $\mathcal{KL}_B$ tasks can be mapped into the above abductive framework.

### 4.1 Prediction

*Prediction* is the process of seeing what will follow from some events $\mathcal{IN}$. This can be implemented in HT4 by making $\mathcal{OUT} \subseteq \mathcal{V} - \mathcal{IN}$; i.e. find all the non-input vertices we can reach from the inputs. This is a non-naive implementation of prediction since mutually exclusive predictions (the *covered* elements of $\mathcal{OUT}$) will be found in different worlds. Note that in the special case where:

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

then our abductive system will compute ATMS-style *total envisionments*; i.e. all possible consistent worlds that are extractable from the theory (for more on the ATMS, see Section 6.4). 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. $\mathcal{OUT} \subset \mathcal{V} - \mathcal{IN}$).

```
if   day = tuesday and weather = fine and
     wind = high
then wash

if   weather = raining and football = on
then watchTV
```

Figure 6: $\mathcal{T}_2$: Tuesday can be washing day.

## 4.2 Classification

*Classification* is just a special case of prediction with the *interesting subset* set to the vertices representing the possible classifications. Consider a theory $\mathcal{T}$ containing conjunctions of attributes that list the properties of some class. When converted to $\mathcal{D}$, the classes and attributes become different vertices of $\mathcal{D}$. Inference edges are added from the attributes to the proposition that some class is true (the *modus ponens* link). Further, we link the negation of the class with the negation of the conditions (the *modus tollens* link). For example, the rules in Figure 6 are the theory $\mathcal{T}_2$. When executing this theory, $\mathcal{OUT}$ are the classes {`wash, watchTV`}. Note that we can use the modus tollens links to prove `not(watchTV)` if we can prove `not(weather=raining)` *or* `not(football=on)`.

$\mathcal{D}_2$ is generated by a model-compiler for propositional systems (see Figure 7, note the modus tollens links). The invariants for $\mathcal{D}_2$ are shown in Figure 8. $\mathcal{D}_2$ contains *partial-match and-vertices* (`&009=partial` and `&010=partial`). HT4 can interpret partial-match vertices in one of two ways:

- *Total-match*: Partial-match vertices can be used as a true and-vertex; i.e. we can only reach `wash` if we can also reach all of {`day=tuesday, weather=fine, wind=high`}.
- *True partial-match*: In the partial-match case, HT4 would treat (e.g. `&009=partial`) as an or-vertex during world generation. However, when applying $\mathcal{BEST}_{PARTIAL-MATCH}$, we could elect to favour the worlds that contain post-conditions with the most number of pre-conditions. For example, if $\mathcal{FACTS}$ was {`day=tuesday, football=on`} and we had no information about the `weather` or the `wind`, then a $\mathcal{BEST}$ operator could still make a case that `watchTV` was more likely than `wash` since 50% of the ancestors of `watchTV` are known compared with 33% of the ancestors for `wash`.

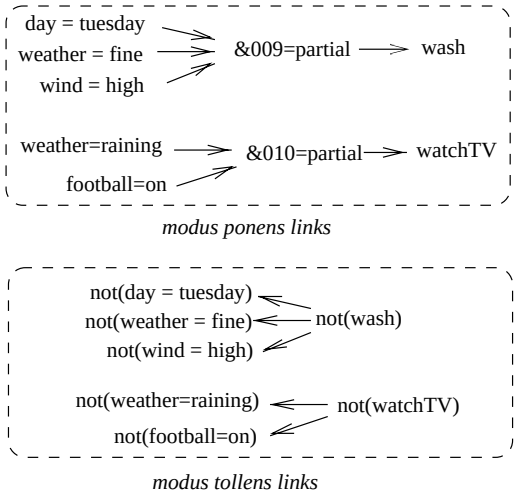


Figure 7: $\mathcal{D}_2$ generated from $\mathcal{T}_2$.

```
% A proposition and its negation are
% inconsistent
i(not(X),X).

% X cannot be in two different states
i(X=State1,X=State2) :- not(State1=State2).

% Can't wash and watch TV at the same time.
i(wash,watchTV).
```

Figure 8: Invariants $\mathcal{I}$ for $\mathcal{D}_2$.

The model compiler for theories containing generalisation links (e.g. $\mathcal{T}_3$ shown in Figure 9) must add extra links. Given a super-class, we can infer *down* to some sub-class if we can demonstrate that the extra-properties required for the sub-class are also believable. The vertex `&013=partial` in Figure 10 is such a specialisation link (for the sake of simplicity, we do not show the modus tollens links in Figure 10).

$\mathcal{T}_3$ contains an interesting semantic issue. `Emus` override the `motion` slot inherited from `birds`. Ignoring, temporarily, this issue, we can see from $\mathcal{D}_3$, we can infer from `bird` to `emu` if that animal lives in Australia. The classifications returned by HT4 could be further customised by using $\mathcal{BEST}_{SPECIFIC}$ that favours the worlds that include the most-specific classes [59] (e.g. `emu` is better than `bird`).

Returning now to the `motion` issue, we note that cancelations in inheritance networks is a difficult

```
frame(bird, [diet      = worms,
             big-limbs = 2,
             motion    = flies,
             home      = nest]).

% An emu is a bird that does not fly and
% lives in australia
frame(emu,  [isa       = bird,
             habitat   = australia,
             motion    = walks]).
```

Figure 9: $\mathcal{T}_3$: Things that fly and walk.

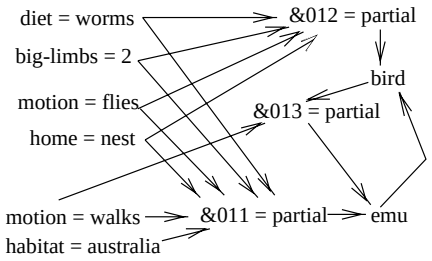


Figure 10: $\mathcal{D}_3$ (modus ponens links only).

problem. Brachman argues that we should not use these since such overrides complicate the semantics of the network [3] (e.g. Figure 11). In the general case, the processing of overrides in inheritance networks requires some form of multiple-worlds reasoning [23] such as default logic [71]). Default logic and abduction share a computational core [74]. Using HT4, we can process $\mathcal{T}_3$. In the case where `motion` is an assumption and both `motion = walks` and `motion = flies` appear in proofs, then `emu` and `bird` will appear in separate worlds. $\mathcal{BEST}$ can then be customised to select the world(s) that are most acceptable to the user (e.g. $\mathcal{BEST}_{SPECIFIC}$).

## 4.3 Explanation

Wick and Thompson report that the current view of *explanation* is more elaborate than merely "print the rules that fired" or the "how" and "why" queries of MYCIN [83]. Explanation is now viewed as an inference procedure in its own right rather than a pretty-print of some filtered trace of the proof tree. In the current view, explanations should be customised to the user and the task at hand. For example, Paris describes an explanation algorithm that switches from process-based explanations to

```
frame(generic_bird, [diet      = worms,
                     big-limbs = 2,
                     home      = nest]).

frame(bird,         [isa    = generic_bird,
                     motion = flies]).

frame(emu,          [isa     = generic_bird,
                     habitat = australia,
                     motion  = walks]).
```

Figure 11: $\mathcal{T}_{3.1}$: A Brachman-style version of $\mathcal{T}_3$.

parts-based explanations whenever the explanation procedure enters a region which the user is familiar with [58].

This current view of explanation can be modeled as abduction (an insight we first gained from Leake [39]). Given a user profile listing the vertices familiar to the user and the edges representing processes that the user is aware of, then $\mathcal{BEST}_{EXPLANATION}$ favours the worlds with the largest intersection to this user profile. For example, suppose the link `g` $\stackrel{++}{\rightarrow}$ `e` in Figure 2 represented a process that some user did not understand. Their user profile would therefore not contain the edge `gUp` $\rightarrow$ `eUp`. Applying $\mathcal{BEST}_{EXPLANATION}$, we would reject $\mathcal{W}_1$ (see Figure 4) and report $\mathcal{W}_2$ (see Figure 5).

## 4.4 Tutoring

Suppose we can assess that the $\mathcal{BEST}$ explainable world was somehow sub-optimum; e.g. there exists worlds which explain far more $\mathcal{OUT}$puts that those explained by the worlds found by $\mathcal{BEST}_{EXPLANATION}$. We could then set a tutoring goal; i.e. educate our user about the edges which they currently can't accept as explanations.

Continuing the example in the previous section, an abductive tutoring system would note that the user's lack of knowledge about `g` $\stackrel{++}{\rightarrow}$ `e` was compromising their ability to reason effectively. Hence, it would could present to the user $\mathcal{IN}$put-$\mathcal{OUT}$put pairs which exercised that edge. The tutoring session would be termed a success when the user starting accepting explanations based on `g` $\stackrel{++}{\rightarrow}$ `e`.

## 4.5 Qualitative Reasoning

HT4 was originally developed as a *qualitative reasoning* algorithm for neuroendocrinology [25, 26,

52]. Qualitative reasoning is the study of systems whose numeric values are replaced by one of three qualitative states: *UP*, *DOWN* or *STEADY* [35]. A fundamental property of such systems is their indeterminacy. In the case of competing qualitative influences, three possible results are *UP*, *DOWN* or *STEADY*. These alternatives and their consequences must be considered separately.

Abduction can maintain these alternatives in separate worlds. The processing of $\mathcal{T}_1$ (described above) shows how we process qualitative theories abductively. For more details, see [48].

## 4.6 Planning

*Planning* is the search for a set of operators that convert some current state into a goal state. We can represent planning in our abductive approach as follows:

- Represent operators as rules that convert some state to some other state;
- Augment each operator rule with:
  - a unique label $\mathcal{S}_1$, $\mathcal{S}_2$, etc. When $\mathcal{D}$ is generated, each edge will now include the name(s) of the operator(s) that generated it.
  - A cost figure representing the effort required to apply this operator rule.
- Set $\mathcal{IN}$ to the current state, $\mathcal{OUT}$ to the goal state, and $\mathcal{FACTS} = \mathcal{IN} \cup \mathcal{OUT}$.
- Set $\mathcal{BEST}_{PLANNING}$ to favour the world(s) with the least cost (the cost of a world is the maximum cost of the proofs in that world).
- Run HT4. Collect and cache the generated worlds.
- For each $\mathcal{BEST}$ world, collect all the names of the operators used in the edges of that world. These operators will be in a tree structure that reflects the structure of the $\mathcal{BEST}$ worlds. Report these trees as the output plans.

## 4.7 Monitoring

*Monitoring* is the process of checking that the current plan(s) are still possible. The worlds generated by the above planner will contain some assumptions. As new information comes to light, some of these assumptions will prove to be invalid. Delete those worlds from the set of possible plans. The remaining plans represent the space of possible ways to achieve the desired goals in the current situation. If all plans are rejected, then run HT4 again with all the available data.

## 4.8 Diagnosis & Probing

Parsimonious *set-covering diagnosis* [69] 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 [38].

The opposite of set-covering diagnosis is *consistency-based diagnosis* [19, 29, 60, 72] where all worlds consistent with the current observations are generated[1]. Computationally, this is equivalent to the *prediction* process described above, with small variants. For example, in Reiter's variant on consistency-based diagnosis [72], all predicates relating to the behaviour of a theory component $\mathcal{V}_x$ assume a test that $\mathcal{V}_x$ in not acting $AB$normally; i.e. $\neg AB(\mathcal{V}_x)$. $\mathcal{BEST}_{REITER}$ is to favour the worlds that contain the least number of $AB$ assumptions.

A related task to diagnosis is *probing*. When exploring different diagnosis, an intelligent selection of tests (probes) can maximise the information gain while reducing the testing cost [19]. In HT4, we would know to favour probes of $\mathcal{A}_B$ over probes of $\mathcal{A}_C$ over probes of non-controversial assumptions.

## 4.9 Validation

KBS *validation* tests a theory's validity against external semantic criteria. 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 $\mathcal{IN} \cap \mathcal{W}_x$) [47].

Note that this definition of *validation* corresponds to answering the following question: "can a theory of $X$ explain known behaviour of $X$?". We have argued elsewhere that this is the definitive test for a theory [45]. Note that this is a non-naive implementation of KBS validation since it handles certain interesting cases. In the situation where no current theory explains all known behaviour, competing theories can still be assessed by the extent to which they cover known behaviour. Theory $X$ is

[1] For an clear comparison of set-covering diagnosis as abduction and consistency-based-diagnosis as abduction see [14]

definitely better than theory $Y$ if theory $X$ explains far more behaviour than theory $Y$.

As an example of validation-as-abduction, recall that $\mathcal{W}_1$ (see Figure 4) was generated from $\mathcal{T}_1$ when $\mathcal{IN} = \{\texttt{aUp, bUp}\}$ and $\mathcal{OUT} = \{\texttt{dUp, eUp,fDown}\}$. Note that $\mathcal{W}_1^{covered}$ is all of $\mathcal{OUT}$. $\mathcal{T}_1$ is hence not invalidated since there exists a set of assumptions under which the known behaviour can be explained.

HT4 was originally built for validation purposes (HT is short for "hypothesis tester"). The historical precursor to HT4 was Feldman & Compton's QMOD/JUSTIN system (which we call HT1). Feldman & Compton applied their QMOD/JUSTIN algorithm [25, 26] to a qualitative model of a summary paper by Smythe [76] on glucose regulation. They found that 109 of the 343 (32%) data points published to support the Smythe theory could not be explained with respect to that theory. Further, when they showed these findings to the researchers who contributed to the Smythe theory, they found that the errors detected by QMOD/JUSTIN had not been seen before. That is, the faults detected by QMOD/JUSTIN were invisible to existing model review techniques in neuroendocrinology (all the analysed models and data were taken from international refereed journals). Our own subsequent study using HT4 corrected some features of the Feldman & Compton study to increase the inexplicable percentage from 32% to 45%. $\mathcal{D}_{SMYTHE}$ contained 294 and-vertices, 260 or-vertices and had an average fanout of 2.25 (the fanout $\mathcal{F}$ of a graph equals $\frac{|\mathcal{E}|}{|\mathcal{V}|}$ as is the average number of edges leaving a vertex).

Another smaller study [43] found faults in another published scientific theory [75]. Apart from the insights into neuroendocrinological models, the above results are interesting for two reasons:

- 32-45% inexplicable data seems surprisingly high for models that have run the gauntlet of international peer review. We will later find that the computational complexity of the validation-as-abduction inference process is high. It is therefore no surprise that human beings, with their limited short-term memory, do not completely test their models.

- Significantly, this study faulted a model using the data published to support that model. Clearly, human researchers do not rigorously explore all the consequences of their observations (perhaps since the process is so computational complex). Automatic tools such as HT4 can be a useful intelligent assistant for checking hypothetical ideas.

An interesting variant on our external semantic testing approach are the automatic test suite generation procedures offered by the dependency-network approaches of Ginsberg [30, 31] and Zlatereva [87, 88]. The dependencies between rules/conclusions are computed and divided into mutually consistent subsets. The root dependencies of these subsets represent the space of all reasonable tests. If these root dependencies are not represented as inputs within a test suite, then the test suite is incomplete. Test cases can then be automatically proposed to fill any gaps.

The advantage of this technique is that it can be guaranteed that test cases can be generated to exercise all branches of a knowledge base. The disadvantage of this technique is that, for each proposed new input, an expert must still decide what constitutes a valid output. This decision requires knowledge external to the model, least we introduce a circularity in the test procedure (i.e. we test the structure of $\mathcal{T}_i$ using test cases derived from the structure of $\mathcal{T}_i$). Further, auto-test-generation focuses on incorrect features in the current model. We prefer to use test cases from a totally external source since such test cases can highlight what is absent from the current model. For these reasons, we caution against automatic test suite generation. Nevertheless, if it is required, HT4 can compute these test suites. Once a *total envisionment* is executed (recall section 4.1), the required test suites are the roots and base controversial assumptions of the generated worlds.

## 4.10 Verification

KBS *verification* tests a theory's validity against internal syntactic criteria [67]. HT4 could be used for numerous KBS verification tests. For example:

- *Circularities* could be detected by computing the transitive closure of the and-or graph. If a vertex can be found in its own transitive closure, then it is in a loop.

- *Ambivalence* (a.k.a. inconsistency) could be reported if more than one world can be generated.

- *Un-usable rules* could be detected if the edges from the same $\mathcal{S}_x$ statement in the knowledge base touch vertices that are incompatible (defined by $\mathcal{I}$).

We prefer external semantic criteria (e.g. the above validation technique) to internal syntactic criteria since we know of fielded expert systems that contain syntactic anomalies, yet still perform adequately [68].

# 5 Practicality

## 5.1 Complexity

The core computational problem of HT4 is the search for the assumptions $\mathcal{ENV}_i$ which define each world $\mathcal{W}_i$. Earlier versions of HT4 [25, 26, 43] computed the $\mathcal{BEST}$ worlds $\mathcal{W}$ via a basic depth-first search chronological backtracking algorithm (DFS) with no memoing. Mackworth [41] and DeKleer [16] 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: the *facts sweep*, the *forwards sweep*, the *backwards sweep*, and the *worlds sweep*.

### 5.1.1 Facts Sweep

In the case where $< \mathcal{V}, \mathcal{E} >$ is pre-enumerated and cached and $\mathcal{I}$ has an arity of 2, a hash table $\mathcal{NOGOOD}$ can be built in $O(|V|^2)$ time that maps every vertex to the set of vertices that it is incompatible with. Once $\mathcal{NOGOOD}$ is known, the facts sweep can cull all $\mathcal{V}_x$ that are inconsistent with the $\mathcal{FACTS}$ in time $O(|V|)$. Note a simplifying assumption made by HT4 is that $\mathcal{NOGOOD}$s are only defined for $\mathcal{V}^{or}$ vertices (i.e. the $\mathcal{NOGOOD}$ sets for $\mathcal{V}^{and}$ are empty).

### 5.1.2 Forwards Sweep

The controversial assumptions $\mathcal{A}_C$ are computed as a side-effect of forward chaining from $\mathcal{IN}$ (ignoring $\mathcal{I}$) to find $\mathcal{IN}^*$, the vertices reachable from $\mathcal{IN}$. In the worst case, finding $\mathcal{IN}^*$ is transitive closure (i.e. $O(|V|^3)$). Once $\mathcal{IN}^*$ is known, $\mathcal{A}_C$ can be found in time $O(|V|)$.

### 5.1.3 Backwards Sweep

The base controversial assumptions $\mathcal{A}_B$ are computed as a side-effect of growing proofs back from $\mathcal{OUT}$ across $\mathcal{IN}^*$. Each proof $\mathcal{P}_y$ contains it's *forbids* set (the vertices that, with $\mathcal{P}_y^{used}$, would violate $\mathcal{I}$), and the upper-most $\mathcal{A}_C$ (called the proof *guess*) found during proof generation. The backwards sweep handles $\mathcal{V}^{or}$ vertices differently to $\mathcal{V}_{and}$ vertices:

- A candidate $\mathcal{V}_x^{or}$ for inclusion in $\mathcal{P}_y^{used}$ must satisfy $\mathcal{V}_x^{or} \notin \mathcal{P}_y^{used}$ (loop detection) and $\mathcal{V}_x^{or} \notin \mathcal{P}_y^{forbids}$ (consistency check). If the candidate vertex is added to the proof, the vertices that are $\mathcal{NOGOOD}$ with $\mathcal{V}_x^{or}$ are added to $\mathcal{P}_y^{forbids}$.

- After checking for looping, a candidate $\mathcal{V}_x^{and}$ (which is not a partial-and vertex) that seeks inclusion in $\mathcal{P}_y^{used}$ must check all combinations of all proofs which can be generated from its parents. The cross-product $\theta$ of the proofs from the $\mathcal{V}_x^{and}$ parent vertices is calculated (which implies a recursive call to the backwards sweep for each parent, then collecting the results in a temporary). The proofs in $\theta_i$ plus $\mathcal{P}_y^{used}$ are *combined* to form the single proof $\theta_i'$. Proof combination generates a new proof whose *used*, *forbids* and *guesses* sets are the union of these sets from the combining proofs. A combined proof is said to be *valid* if the *used* set does not intersect with the *forbids* set. Each valid $\theta_i'$ represents one use of $\mathcal{V}_x^{and}$ to connect an $\mathcal{OUT}$ vertex to the $\mathcal{IN}$ set.

- Partial-and vertices are treated as or-vertices by the backwards sweep.

After all the proofs are generated, the union of all the proof *guess* sets is $\mathcal{A}_B$. If the average size of a proof is $N$ and the average fanout of the graph is $\mathcal{F}$, then worse case backwards sweep is $O(N^{\mathcal{F}})$.

### 5.1.4 Worlds Sweep

HT4 assumes that its $\mathcal{V}^{or}$ are generated from attributes with a finite number of mutually exclusive discrete states (e.g. {*day=mon, day=tues,...*}). With this assumption, the generation of $\mathcal{ENV}_i$ is just the cross product of all the used states of all the attributes found in $\mathcal{A}_B$. The worlds sweep is simply two nested loops over each $\mathcal{ENV}_i$ and each $\mathcal{P}_j$ (i.e. $O(|\mathcal{ENV}| * |\mathcal{P}|)$).

Somewhere within the above process, the $\mathcal{BEST}$ criteria must be applied to cull unwanted worlds. HT4 applies $\mathcal{BEST}$ after world generation. 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.
- *Worlds-level* assessment operators can execute when the worlds are known; e.g. the validation algorithm described in section 4.9.

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 $O(N^{\mathcal{F}})$ complexity of the backwards sweep (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.

## 5.2 Experiments

Abduction has a reputation of being impractically slow [22]. Selman & Levesque show that even when only one abductive explanation is required and $\mathcal{D}$ is restricted to be acyclic, then abduction is NP-hard [74]. Bylander *et. al.* make a similar pessimistic conclusion [6].

In practice these theoretical restrictions may not limit application development. Ng & Mooney report reasonable runtimes for their abductive system using a beam-search proof-level assessment operator [56]. Figure 12 shows the average runtime for executing HT4 using a worlds-level assessment operator over 94 and-or graphs and 1991 $< \mathcal{IN}, \mathcal{OUT} >$ pairs [45]. For that study, a "give up" time of 840 seconds was built into the test engine. HT4 did not terminate for $|\mathcal{V}| \geq 850$ in under that "give up" time (shown in Figure 12 as a vertical line).

In practice, how restrictive is a limit of 850 vertices? Details of the nature of real-world expert

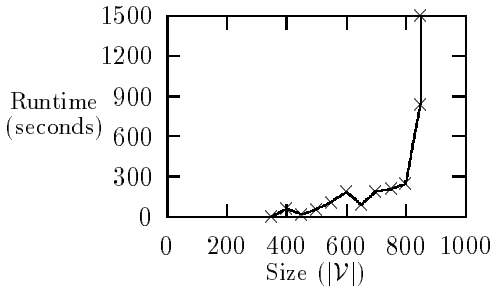


Figure 12: Average runtimes.

| Application | $\lvert\mathcal{V}\rvert$ | $\frac{\lvert\mathcal{E}\rvert}{\lvert\mathcal{V}\rvert}$ |
|---|---|---|
| displan | 55 | 2 |
| mmu | 65 | 7 |
| tape | 80 | 4 |
| neuron | 155 | 4 |
| DMS-1 | 510 | 6 |

Figure 13: Figures from fielded expert systems.

systems are hard to find in the literature. The only reliable data we could find is shown in Figure 13 which shows the size of the dependency graph between literals in fielded propositional expert systems [68]. Figure 13 suggests that a practical inference engine must work at least for the range $55 \geq |\mathcal{V}| \geq 510$ and $2 \geq \frac{|\mathcal{E}|}{|\mathcal{V}|} \geq 7$.

Note that the Figure 12 results were obtained from a less-than-optimum platform: Smalltalk/V on a PowerBook170 (a port to "C" on a Sparc station is currently in progress). However, the current results on a relatively slow platform show that even when we run HT4 sub-optimally (i.e. using worlds-level assessment), it is practical for the theory sizes we see in practice.

Figure 14 studies the practicality of HT4 for models of varying fanout [48]. In this study, model size was kept constant while the fanout was increased. Six models were used of sizes $|\mathcal{V}| = \{449, 480, 487, 494, 511, 535\}$. At low fanouts, many behaviours were inexplicable. However, after a fanout of 4.4, most behaviours were explicable. Further, after a fanout of 6.8, nearly all the behaviours were explicable.

We make two conclusions from Figure 14:

1. HT4 is practical for nearly the range of fanouts seen in fielded expert systems.

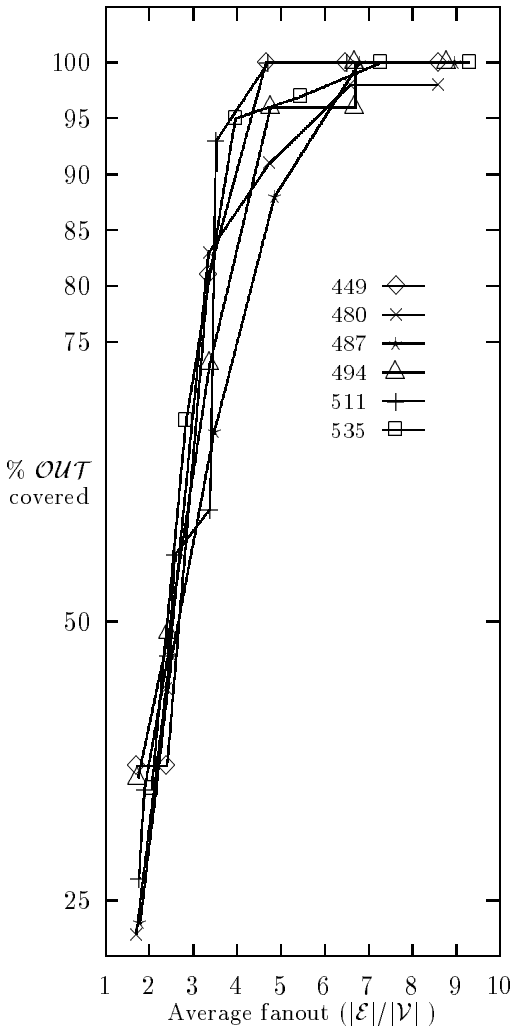


Figure 14: Explicable outputs.

2. However, after a certain level of interconnectivity, a theory is able to reproduce any input/output pairs. An inference procedure that condones any behaviour at all from a theory is not a useful inference procedure. After the *Pendrith limit* (the point where % $\mathcal{OUT}$ *covered* approaches 100%) then worlds-level abductive assessment becomes useless.

We do not view the Pendrith limit as a drawback of our particular abductive approach. Rather, we interpret this result as a general statement of limits to expert systems inference. The advantage of our abductive framework is that it provides for a simple computational view of KBS inference. The computational limits to abduction are really the computational limits to expert systems inference [45]. Figure 14 is telling us that we lose the ability to reason meaningfully about *any* knowledge base for tasks requiring worlds-level assessment (e.g. validation) if it is highly connected (i.e. $\frac{|\mathcal{E}|}{|\mathcal{V}|} > 7$). This point is explored further in [45].

# 6 Discussion

## 6.1 Abduction and $\mathcal{KL}_B$

HT4 is more general than Clancey's approach since it makes explicit certain assumptions which are only tacit in Clancey's approach. For example, Clancey assumes that the best world uses the fewest number of $\mathcal{IN}$s [11, p331]. We have shown above that this is not universally true (recall the different $\mathcal{BEST}$s listed above in sections 3 and 4). Further, HT4 is a single approach for implementing *both* heuristic classification and construction. HT4 supports all the inference primitives required for heuristic classification; i.e. partial match, and the ascent and descent of classification hierarchies. To execute heuristic classification-as-abduction, just execute HT4 with no invariants. Any proofs found between $\mathcal{IN}$ and $\mathcal{OUT}$ can be reported to the user. HT4 also supports the inference primitive required for heuristic construction: assessment of competing inferences. The construction of an SSM from a QM that satisfies some task (specified by $< \mathcal{IN}, \mathcal{OUT} >$) in the presence of invariants is exactly the HT4 algorithm described above. Both proposals can generate multiple worlds/SSMs. Note that HT4 worlds are guaranteed to satisfy Clancey's *coherence* requirement.

As to Breuker's proposal, his components of solutions sounds to us like three recursive calls to a single inference procedure. Recall his argument: all expert system tasks contain the same four components of solutions: an *argument structure* which is extracted from a *conclusion* which is in turn extracted from a *case model* which is in turn extracted from a *generic domain model*. Note that, in all cases, each sub-component is generated by extracting a relevant subset of some background theory to generate a new theory (i.e. abduction). Returning now to HT4, we note that this algorithm also extracts sub-models from super-models. The extracted models are relevant to a particular task; i.e. $< \mathcal{IN}, \mathcal{OUT} >$.

We note that other researchers have recognised

some similarities between different $\mathcal{KL}_B$ tasks. Breuker is a researcher in the $\mathcal{KL}_B$ community. Also, Wielinga *et. al.* note that their description of *systematic diagnosis* and *monitoring* share common features [84]. Finally, Tansley & Hayball [80] note that:

- *Localisation* and *causal tracing* are basically the same process (which they call *systematic diagnosis*), except the former uses *part-of* knowledge while the latter uses *causal* knowledge. In terms of our framework, both would execute over the same and-or graph, but the user's interpretation of the edges would differ.
- Heuristic classification could be used for both classification *and* diagnosis since (says Tansley & Hayball) diagnosis is the backwards process to classification.
- *Scheduling*, *planning*, and *configuration* are actually the same problem, divided on two dimensions (" goal states known or not" and "temporal factors considered or not").

.

However, with the exception of Breuker and Clancey, the $\mathcal{KL}_B$ community does not actively explore these similarities.

## 6.2 Abduction and $\mathcal{KL}_A$ (SOAR)

Our approach has some similarities to the $\mathcal{KL}_A$ SOAR project [53, 55, 54, 73, 85]. We view the state space traversal of SOAR as a directed and-or graph which can be extended at runtime. While an HT4 $\mathcal{D}$ vertex contains a single literal, the vertices of the SOAR state space contain conjunctions of literals.

We prefer our HT4 approach over SOAR for three reasons:

1. HT4 knowledge bases can be validated without additional architecture. In other expert systems approaches (e.g. SOAR), validation requires additional architecture.
2. HT4 is a less complicated architecture than SOAR. SOAR is built on top of an intricate forward-chaining rule-based system. HT4 uses a simpler graph-theoretic approach.
3. HT4 models abduction better than SOAR. Experiments with adding abductive inference to SOAR relied on an interface to an external abductive theorem prover. In Steier's CYPRESS-SOAR/ RAINBOW system, SOAR production rules modeled control decisions, while the RAINBOW abductive inference engine generated possible designs [79]. Given a vertex with N out edges (or, in SOAR-speak, a state space with N associated operators), HT4 assesses the utility of each edge using (potentially) a deferred global analysis. SOAR must make its operator assessment at the local level. SOAR's run-time selective generation of the and-or graph has efficiency advantages since it culls unacceptable alternatives as they are first encountered. Our approach has the potential to be slower, but the explicit representation of all alternatives permits allows for global assessment criteria (e.g. our validation procedure described above).

## 6.3 Other Abductive Research

Descriptions of abduction date back to the "fourth-figure" of Aristotle [81]. In the modern era, abduction was described by Charles Sanders Pierce in the last century as follows:

> The surprising fact C is observed. But if A were true, C would be a matter of course. Hence, there is reason to suspect that A is true [57, introduction].

Pople noted the connection between diagnosis and abduction in 1973. Pople's diagnosis inference process explores a first-order theory looking for hypotheses which, if assumed, could explain known symptoms [65]. The connection between diagnosis and abduction was confirmed later by Reggia in 1985 [70] and other researchers since, particularly in the field of model-based diagnosis (MBD). For example, our distinction between consistency-based diagnosis and set-covering diagnosis in Section 4.8 came from the MBD literature [14].

By the late 1980s, many researchers had recognised the applicability of abduction to a wide-range of domains. The 1990 AAAI Spring Symposium on Automated Abduction [57] lists the following domains as applications of abduction: natural language processing, learning, financial reasoning, analogical reasoning, causal reasoning, probabilistic and qualitative reasoning, just to name a few. Several basic AI algorithms proved to be fundamentally abductive in nature [6, 14, 40]. For example:

- The ATMS (discussed in Section 6.4) is an incremental abductive inference engine. When

a problem solver makes a new conclusion, this conclusion and the reasons for believing that conclusion are passed to the ATMS. The ATMS updates its network of dependencies and sorts out the current conclusions into maximally consistent subsets (which HT4 would call worlds). HT4 borrows the term *minimal environments* from the ATMS research (but shortens it to $\mathcal{ENV}$).

- Bayesian reasoning can be viewed as abduction, but in a numeric paradigm [6]. For an example of Bayesian abduction, see Poole [64]. This numeric Bayesian abductive paradigm may not explictedly represent the multiple-world assumption space of non-numeric abductive techniques such as the ATMS and HT4. We have argued here that the direct manipulation of that assumption space is a useful technique for a wide variety of KBS tasks.

Logic-based frameworks for abduction such as Pople's are common in the literature (e.g. [13, 15, 22, 37, 62, 63, 65]). Our preferred framework uses a graph-theoretic approach; i.e. inference is the selection of some subset of edges from the network of possible proof trees. We find that underneath efficient theorem provers is some sort of graph representation. Hence, we have elected to work directly at the graph-level. The logical school is more concerned with understanding the complex semantics of abduction rather than in the construction of practical systems (counter examples: [22, 63]). Pople himself moved away from a pure logic-based approach in his later work [66] as has other "logical-school" researchers (e.g. Poole [64]).

This "logical-school" typically adopts minimality as the sole criteria for assessing worlds (counter-example: [59]). For example, Console and Torasso [13] explicitly argue for minimality for alternative assessment. Our view is that minimality is pragmatically useful for reducing the time for the inference. Hence, HT4 calculates a minimal critical assumption set $\mathcal{A}_B$. However, we have argued in this article that not all $\mathcal{BEST}$ explanations are minimal. Rather, a comprehensive knowledge-level modeling framework can be developed assuming customisable world assessment.

Abduction is an interesting framework in which to explore non-standard logics [49, 63, 81]. Many of the criticisms against AI (e.g. [2]) are really criticisms of standard deductive logics where the conclusions reached are context independent. We view abduction as a more plausible model of human reasoning since the conclusions made are context-dependent on the $\mathcal{TASK}$ at hand.

Various researchers note that abductive definitions of "explanation" are philosophically problematic. Charniak & Shimony comment that, pragmatically, a logical framework for "explanation" is a useful definition since:

> ...it ties something we know little about (explanation) to something we as a community know quite a bit about (theorem proving) [10].

However, abductive explanation blurs causal implication and logical implication. Charniak & McDermott [9] and Poole [60] caution against mixing up these operators in a single knowledge base. Many researchers acknowledge this as a research area, but then quickly change the topic (e.g. [9, p454], [13, p663],[6, p27], [40, p1061], [50]).

## 6.4 Default Logic and the ATMS

Our base controversial assumptions and worlds are akin to ATMS labels [16, 17, 18, 27] and default logic extensions respectively [71]. However, we differ from ATMS/ default logic two ways:

1. HT4 worlds only contain *relevant* literals; i.e. only those literals that exist on pathways between inputs and outputs. This means that, unlike default logic extensions, not all consequences of a literal that are consistent with that world are in that world. For example, if the $\mathcal{OUT}$ set of our example did not include `eUp`, then `eUp` would not have appeared in the $\mathcal{W}_1$ or $\mathcal{W}_2$ of Figures 4 and 5.

2. A default logic extension must contain the initial set of facts. An HT4 world contains only some subset of the initial $\mathcal{FACTS}$ and $\mathcal{IN}$. HT4 is the search for some subset of the given theory, which can use some subset of the $\mathcal{IN}$puts to explain some subset of desired $\mathcal{OUT}$outs.

Note that HT4 is different to the ATMS in another way. HT4 does not separate a problem solver into an inference engine and an assumption-based truth maintenance system. Such a split may be pragmatically useful for procedural inference engines. However, if we try to specify the innerworkings of a procedural reasoning system, we find that we can model it declaratively by abduction plus $\mathcal{BEST}$ (recall the discussion in section 5.1.4 on how to use $\mathcal{BEST}$ as a procedure to control search space traversal).

### 6.5 Vague Domains

One interesting property of abduction is that it can perform all the above reasoning tasks in *vague domains* which we have previously characterised [45] as domains that are:

- *Poorly measured*: i.e. known data from that domain is insufficient to confirm or deny that some inferred state is valid. Inference in poorly measured domains means making guesses or assumptions. Mutually exclusive assumptions must be managed in separate worlds. HT4 implements this multiple-world reasoning directly.

- *Hypothetical*: i.e. the domain lacks an authoritative oracle that can declare knowledge to be "right" or "wrong". Note that in a well-measured domain, the authoritative oracle could be a database of measurements. Since vague domains lack an authoritative oracle, theories about them may be widely inaccurate. Modeling in vague domains therefore requires a validation engine. HT4 supplies such a validation engine.

- *Indeterminate*: i.e. inferencing over a knowledge base could generate numerous, mutually exclusive, outcomes. For example, recall Figure 2. In the case of both $A$ and $B$ going $UP$, then we have two competing influences on $C$ and it is indeterminate whether $C$ goes $UP$, $DOWN$, or remains $STEADY$. Since the results that can be inferred from the theory are uncertain, it is indeterminate. The indeterminacy of the possible inferences requires some non-monotonic reasoning module. HT4 models non-monotonic reasoning using multiple worlds.

In a review of the KBS domains we have studied in detail since 1985, we found that all were vague domains [45]. These domains were process control, farm management, economic modeling, biochemical interpretation, consumer lending, and model-based diagnosis. Based on this review, we believe that tools that can execute in vague domains are widely applicable.

Easterbrook makes a similar, but more general, point [21]. He finds that most software problems have *conflicts* since they usually contain:

- A thin spread of application domain knowledge (i.e. no definitive oracle).

- Fluctuating and conflicting requirements; e.g. user groups with conflicting needs; conflicts between stated constraints; conflicts between perceived needs; conflicts between evaluations of priorities.

- Breakdowns in communication and co-ordination.

- Areas in which there are different ways of looking at things.

Easterbrook believes that it is artificial to remove these conflicts in software models.

> This insistence that that expertise must be consistent and rational imposes restrictions of the knowledge acquired. The knowledge acquisition process becomes not so much the modeling of the expert's behaviour, but the synthesis of a domain model which need not resemble any mental model used by the expert [21, p264].

Easterbrook argues that software should explicitly model these conflicts since it is exactly this conflicts that will be required to understand opposing positions. We agree. Tools such as HT4 which can explicitly represent conflicts are widely applicable.

## 7 Conclusion

The core shared insight of Clancey and Breuker's work is that theory subset extraction is the central task of KBS inference. Our goal was the description of a minimal architecture necessary to perform this process. Our proposed architecture is the HT4 abductive inference algorithm. In this approach, expert knowledge is represented in the topology of $\mathcal{D}$ and the $\mathcal{BEST}$ operators. We have shown above how HT4 can be used for KBS verification and validation. If we implement expert systems as abduction, then we can execute *and* evaluate our knowledge bases. Further, HT4 can execute in vague and conflicting domains (which we believe occur very frequently). We have found that numerous, seemingly different, $\mathcal{KL}_B$ tasks can be mapped into this single inference procedure. Therefore we believe that abduction in general (and HT4 in particular) is a useful framework for knowledge-level modeling.

# References


[1] J. Bachant and J. McDermott. R1 Revisited: Four Years in the Trenches. *AI Magazine*, pages 21–32, Fall 1984.

[2] L. Birnbaum. Rigor Mortis: A Response to Nilsson's 'Logic and Artificial Intelligence'. *Artificial Intelligence*, 47:57–77, 1991.

[3] R. Brachman. I Lied About the Trees, or Defaults and Definitions in Knowledge Representation. *The AI Magazine*, pages 80–93, Fall 1985.

[4] J. Breuker. Components of Problem Solving and Types of Problems. In *8th European Knowledge Acquisition Workshop, EKAW '94*, pages 118–136, 1994.

[5] B.G. Buchanan and R.G. Smith. Fundamentals of Expert Systems. In P.R. Cohen A. Barr and E.A. Feigenbaum, editors, *The Handbook of Artificial Intelligence, Volume 4*, volume 4, pages 149–192. Addison-Wesley, 1989.

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

[7] A.N. Campbell, V.F. Hollister, R.O. Duda, and P.E. Hart. Recognition of a Hidden Material Deposit by and Artificially Intelligent Program. *Science*, 217:927–929, 3 September 1982.

[8] B. Chandrasekaran. Design Problem Solving: A Task Analysis. *AI Magazine*, pages 59–71, Winter 1990.

[9] E. Charniak and D. McDermott. *Introduction to Artificial Intelligence*. Addison-Wesley, 1987.

[10] E. Charniak and S.E. Shimony. Cost-based abduction and MAP explanation. *Artificial Intelligence*, pages 345–374, 1994.

[11] W. Clancey. Heuristic Classification. *Artificial Intelligence*, 27:289–350, 1985.

[12] W.J. Clancey. Model Construction Operators. *Artificial Intelligence*, 53:1–115, 1992.

[13] L. Console, D.T. Dupre, and P. Torasso. On the Relationship Between Abduction and Deduction. *Journal of Logic Programming*, 1:661–690, 5 1991.

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

[15] P.T. Cox and T. Pietrzykowski. Causes for Events: Their Computation and Applications. In *8th International Conference on Automated Deduction*, pages 608–621. Springer-Verlag, 1986.

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

[17] J. DeKleer. Extending the ATMS. *Artificial Intelligence*, 28:163–196, 1986.

[18] J. DeKleer. Problem Solving with the ATMS. *Artificial Intelligence*, 28:197–224, 1986.

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

[20] R.O. Duda, P.E. Hart, and R. Reboh. Letter to the Editor. *Artificial Intelligence*, 26:359–360, 1985.

[21] S. Easterbrook. Handling conflicts between domain descriptions with computer-supported negotiation. *Knowledge Acquisition*, 3:255–289, 1991.

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

[23] D.W. Etherington and R. Reiter. On Inheritance Hierarchies with Exceptions. In *AAAI-83*, pages 104–108, 1983.

[24] 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.

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

[26] B. Feldman, P. Compton, and G. 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.

[27] K.D. Forbus and J. DeKleer. *Building Problem Solvers*. The MIT Press, 1993.

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

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

[30] A. Ginsberg. A new Approach to Checking Knowledge Bases for Inconsistentcy and Redundancy. In *Proc. 3rd Annual Expert Systems in Government Conference*, pages 102–111, 1987.

[31] A. Ginsberg. Theory Reduction, Theory Revision, and Retranslation. In *AAAI '90*, pages 777–782, 1990.

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

[33] K. Hirata. A Classification of Abduction: Abduction for Logic Programming. In *Proceedings of the Fourteenth International Machine Learning Workshop, ML-14*, page 16, 1994. Also in *Machine Intelligence* 14 (to appear).

[34] Y. Iwasaki. Causal Ordering in a Mixed Structure. In *Proceedings of AAAI '88*, pages 313–318, 1988.

[35] 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.

[36] Y. Iwasaki and H.A. Simon. Causality in Device Behaviour. *Artificial Intelligence*, 29:3–31, 1986.

[37] A.C. Kabas and P. Mancrella. Generalized Stable Models: A Semantics for Abduction. In *ECAI-90*, 1990.

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

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

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

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

[42] D. Marques, G. Dallemagne, G. Kliner, J. McDermott, and D. Tung. Easy Programming: Empowering People to Build Their Own Applications. *IEEE Expert*, pages 16–29, June 1992.

[43] 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.

[44] T. J. Menzies. Applications of Abduction #1: Intelligent Decision Support Systems. In *Proceedings of the Melbourne Workshop on Intelligent Decision Support*. Department of Information Systems, Monash University, Melbourne, 1996. Also, TR95-16, Department of Software Development, Monash University.

[45] 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.

[46] T.J. Menzies. Limits to Knowledge Level-B Modeling (and KADS). In *Proceedings of AI '95, Australia*. World-Scientific, 1995.

[47] T.J. Menzies. On the Practicality of Abductive Validation. Technical Report TR95-38, Department of Software Development, Monash University, 1995.

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

[49] T.J. Menzies. Situated Semantics is a Side-Effect of the Computational Complexity of Abduction. In *Australian Cognitive Science Society, 3rd Conference*, 1995.

[50] T.J. Menzies. Applications of Abduction: Knowledge Level Modeling. *International Journal of Human Computer Studies*, September, 1996.

[51] T.J. Menzies and P. Compton. *A Precise Semantics for Vague Diagrams*, pages 149–156. World Scientific, 1994.

[52] T.J. Menzies, P. Compton, B. Feldman, and T. Toft. Qualitative Compartmental Modeling. In *Proceedings of the AAAI Symposium on Diagrammatic Reasoning Stanford University, March 25-27*, 1992.

[53] A. Newell. The Knowledge Level. *Artificial Intelligence*, 18:87–127, 1982.

[54] A. Newell. Reflections on the Knowledge Level. *Artificial Intelligence*, 59:31–38, Feburary 1993.

[55] A. Newell, G.R. Yost, J.E Laird, P.S. Rosenbloom, and E. Altmann. Formulating the Problem Space Computational Model. In P.S. Rosenbloom, J.E. Laird, and A. Newell, editors, *The Soar Papers*, volume 2, pages 1321–1359. MIT Press, 1991.

[56] 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.

[57] 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.

[58] 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.

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

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

[61] 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.

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

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

[64] D. Poole. Probabilistic Horn Abduction and Bayesian Networks. *Artificial Intelligence*, 64:81–129, 1993.

[65] H.E. Pople. On the mechanization of abductive logic. In *IJCAI '73*, pages 147–152, 1973.

[66] H.E. Pople. The Formation of Composite Hypothess in Diagnositic Problem Soving: An Exercise in Synthetic Reasoning. In *IJCAI '77*, pages 1030–1037, 1977.

[67] A.D. Preece. Principles and Practice in Verifying Rule-based Systems. *The Knowledge Engineering Review*, 7:115–141, 2 1992.

[68] A.D. Preece and R. Shinghal. Verifying Knowledge Bases by Anomaly Detection: An Experience Report. In *ECAI '92*, 1992.

[69] 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.

[70] J.A. Reggia. Abductive Inference. In *Proceedigns of the Expert Systems in Government Symposium*, pages 484–489, 1985.

[71] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.

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

[73] P.S. Rosenbloom, J.E. Laird, and A. Newell. *The SOAR Papers*. The MIT Press, 1993.

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

[75] G.A. Smythe. Hypothalamic noradrenergic activation of stress-induced adrenocorticotropin (ACTH) release: Effects of acute and chronic dexamethasone pre-treatment in the rat. *Exp. Clin. Endocrinol. (Life Sci. Adv.)*, pages 141–144, 6 1987.

[76] G.A. Smythe. Brain-hypothalmus, Pituitary and the Endocrine Pancreas. *The Endocrine Pancreas*, 1989.

[77] L. Steels. Components of Expertise. *AI Magazine*, 11:29–49, 2 1990.

[78] M. Stefik, J. Aikins, R. Balzer, J. Benoit, L. Birnhaum, F. Hayes-Roth, and E. Sacerdoti. The Organisation of Expert Systems, A Tutorial. *Artificial Intelligence*, 18:135–127, 1982.

[79] D.M. Steier. CYPRESS-SOAR: A Case Study in Search and Learning in Algorithm Design. In P.S. Rosenbloom, J.E. Laird, and A. Newell, editors, *The SOAR Papers*, volume 1, pages 533–536. MIT Press, 1993.

[80] D.S.W. Tansley and C.C. Hayball. *Knowledge-Based Systems Analysis and Design*. Prentice-Hall, 1993.

[81] P. Wang. From Inheritance Relation to Non-Axiomatic Logic. Technical report, Center for Research on Concepts and Cognition, Indiana University, Bloomington, Indiana,, 1993.

[82] S.M. Weiss, C.A. Kulikowski, and S. Amarel. A Model-Based Method for Computer-Aided Medical Decision-Making. *Artificial Intelligence*, 11, 145-172 1978.

[83] M.R. Wick and W.B. Thompson. Reconstructive Expert System Explanation. *Artificial Intelligence*, 54:33–70, 1992.

[84] B.J. Wielinga, A.T. Schreiber, and J.A. Breuker. KADS: a Modeling Approach to Knowledge Engineering. *Knowledge Acquisition*, 4:1–162, 1 1992.

[85] G.R. Yost and A. Newell. A Problem Space Approach to Expert System Specification. In *IJCAI '89*, pages 621–627, 1989.

[86] V.L. Yu, L.M. Fagan, S.M. Wraith, W.J. Clancey, A.C. Scott, J.F. Hanigan, R.L. Blum, B.G. Buchanan, and S.N. Cohen. Antimicrobial Selection by a Computer: a Blinded Evaluation by Infectious Disease Experts. *Journal of American Medical Association*, 242:1279–1282, 1979.

[87] N. Zlatareva. CTMS: A General Framework for Plausible Reasoning. *International Journal of Expert Systems*, 5:229–247, 3 1992.

[88] N. Zlatareva. Distributed Verification and Automated Generation of Test Cases. In *IJCAI '93 workshop on Validation, Verification and Test of KBs Chambery, France*, pages 67–77, 1993.

Some of the Menzies papers can be found at *http:// www.sd.monash.edu.au/ ~timm/pub/ docs/papersonly.html* .