Vague Models and Their Implications for the KBS Design Cycle

Tim Menzies
Dept. of Software Development, Monash University,
Melbourne, Australia
timm@insect.sd.monash.edu.au

Simon Goss
Air Operations Division, Aeronautical & Maritime
Research Laboratory, Melbourne, Australia.
simon.goss@dsto.defence.gov.au*

September 22, 1996

Abstract

Standard software engineering methodologies are typically prescriptions on how to develop some *initial* system. Here we formalise the process of using an *existing*, possibly poorly understood, system. Informal vague causal diagrams are a common technique for illustrating and sharing intuitions about such poorly understood systems. Normally, such diagrams are viewed as precursors to other modeling techniques. Here, we take another approach and explore what we can do with these vague diagrams *without* requiring precise analysis. Vague models can contain inaccuracies and must be tested. In vague domains, *if we can't test it then we shouldn't model it.* That is, the computational properties of the test engine constrains the modeling process

1 Introduction

Informal vague causal diagrams such as Figure 1 are a common technique for illustrating and sharing intuitions about a domain. Normally, such diagrams are viewed as precursors to other modeling techniques. That is, the standard approach is to continue the analysis process till the vagueness goes away.

Here, we take another approach and explore what we can do with these vague diagrams *without* requiring precise analysis. Our approach is useful for both the acquisition and maintenance of knowledge. Standard software engineering methodologies are typically prescriptions on how to develop some initial system. Here we formalise the process of handling an existing, possibly poorly understood, system. Our goal is the structuring and the optimisation of the resources available

*Proceedings of the Pacific Knowledge Acquisition Workshop, Sydney, Australia, 1996. Available from *http:// www.cse.unsw.edu.au/ ~timm/pub/ docs/papersonly.html*

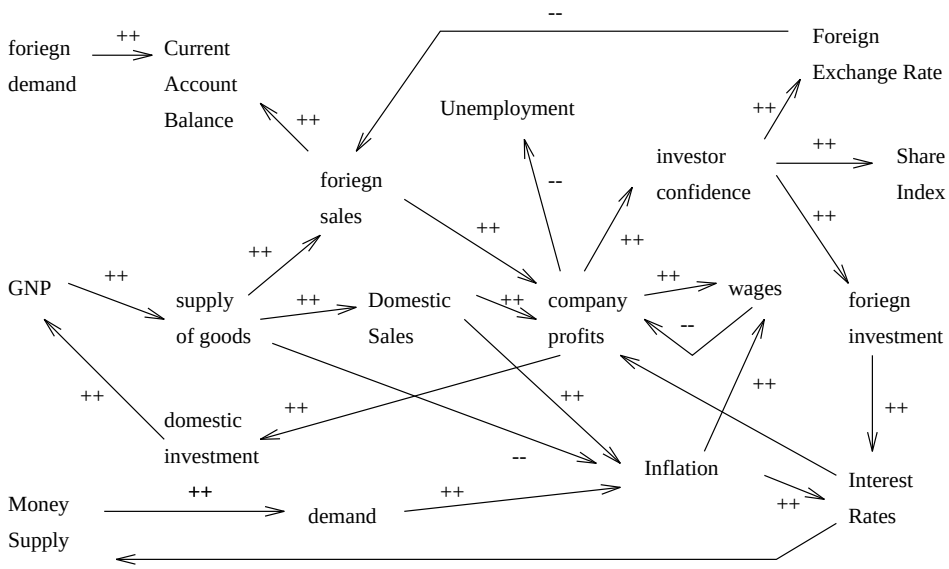


Figure 1: An economic vague causal diagram; from [5].

during the maintenance phase. In the usual case, software maintenance is the process of understanding someone else's idiosyncratic code which may or may not be documented. An extreme software maintenance problem occurs when handling *legacy systems* where the maintainer may not even have access to the source code.

Vague models must be processed differently to standard software systems. The standard software *design activity cycle* shown in Figure 2 [11]. Logical design comprises a brainstorming *elicitation* stage followed by a *representation* stage that systematises the concepts found during elicitation. To make the representations exhibit behaviour, they must be *transformed* into a lower-level representation which can be operationalised. During this transformation stage, implementation-dependent details will contort the representation. Once operationalised, these concepts can be executed and tested. Testing can be divided into *verification* (i.e. was the system built right?) and *validation* (i.e. was the right system built?). Feedback from the testing process can improve the physical and logical designs.

In modeling of *vague domains* (see §2.1), systems development spends more time in testing and correction than initial logical design. Hence, development should move through the logical design stage as quickly as possible since most of the specification will be found in the testing stage. Vague modeling cannot produce *correct* models. Rather, it can only produce models that contain portions that are useful for known purposes.

Our preferred test engine is the HT4 abductive validation engine (summarised in §2.2). Previously, we have discussed HT4 and its applications to KBS inference and verification [17, 18]. This article maps these low-level discussions into the high-level processes referred to in Figure 2. At its core, HT4 maps a theory down into a directed graph expressing the connections and invariants

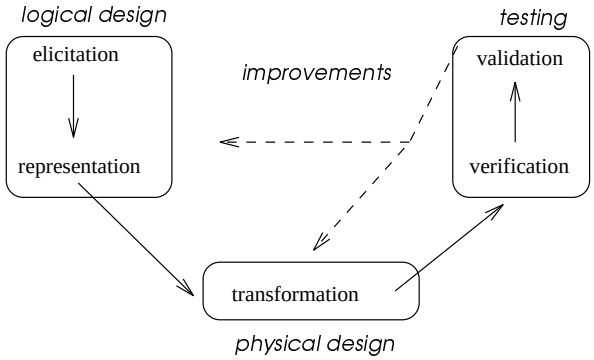


Figure 2: The design activity cycle; adapted from [11]

between entities in the theory. This low-level data structure, we will argue, is a unifying framework for operationalising most of the vague modeling process.

Figure 3 shows our proposed design activity cycle for vague domains. Elicitation (§3.1) lets us articulate and represent (§3.2) our *theory*; a library of desired behaviour (called the *observations*); and a success criteria for judging between competing inferences. In the transformation stage (§3.3), we:

- Build a *model compiler* to convert the theory into a directed graph showing dependencies between entities in the theory (§3.3.1). This dependency graph can be passed to a graph-theoretic verification engine (§3.4).
- Build a *data compiler* to map our library of desired behaviours into a set of pairs: $< \mathcal{IN}, \mathcal{OUT} >$ (§3.3.2).
- Implement the success criteria as the $\mathcal{BEST}$ assessment operator (§3.2.3).

At execution time, each $< \mathcal{IN}, \mathcal{OUT} >$ pair is processed. Portions of the dependency graph are extracted that connect members of $\mathcal{OUT}$ back to members of $\mathcal{IN}$. Recall that we are executing vague models. Vague models are necessarily incomplete; i.e. they are under-specified. When executing such under-specified models, guesses have to be made and mutually exclusive guesses must be managed separately. The extracted portions of the dependency graph must be divided into internally consistent *worlds*. The generation and assessment of the these worlds is a framework for many expert system inference tasks, including validation. Good theories can generate worlds that contain most of $\mathcal{OUT}$; i.e. they can explain a majority of the known behaviour.

Vague models contain inaccuracies. Therefore, vague modeling is only possible when the theory can be tested. That is, the computational properties of the validation engine constrain the modeling process; i.e. *if we can't test it then we shouldn't model it.* We will note below where the limits of the test device constrain the design process.

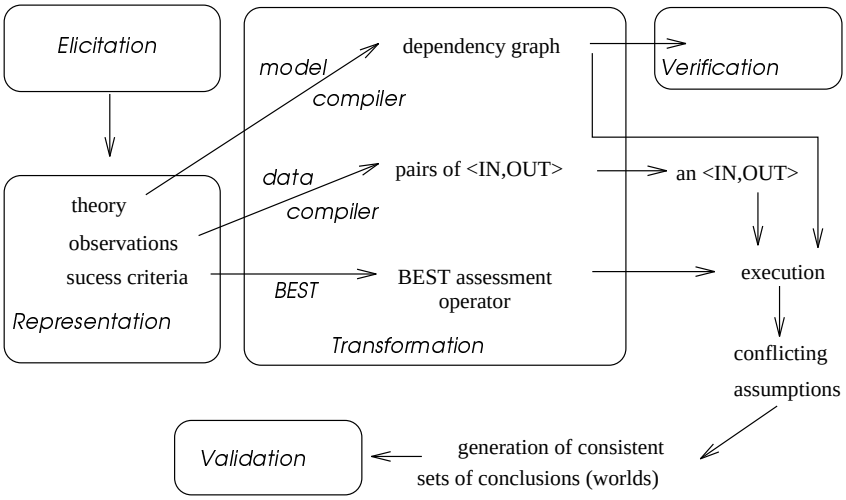


Figure 3: Vague Modeling: the Details

2 Background

In order to explicate our argument, we begin with some background material about example vague domains (§2.1) and abduction (§2.2).

2.1 Example Vague Domains

Elsewhere [15], we have characterised vague domains as: (i) *poorly measured*, i.e. known data from that domain is insufficient to confirm or deny that some inferred state is valid; (ii) *hypothetical*, i.e. the domain lacks an authoritative oracle (e.g. a person or a large library of known behaviour) that can declare knowledge to be "right" or "wrong"; (iii) *Indeterminate*, i.e. inferencing over a knowledge base could generate numerous, mutually exclusive, outcomes.

The approached proposed here was developed by the first author for a neuroendocrinological domain (§2.1.1) which we would describe as the analysis of perturbations of an equilibrium system[1]. Our current goal is to test the generality of that previous work approach in the domain of black-box comprehension (§2.1.2). We would describe the air traffic control simulators found in this second, black-box comprehension domain as spatio-temporal reasoning over non-equilibrium systems. Note that our work in this second black-box comprehension domain is still in its preliminary stages.

2.1.1 Neuroendocrinology

Neuroendocrinology is the study of the interaction of glands and nerves. Obtaining values for certain chemicals within the body is not as simple as, say,

[1] HT4 is an extension and a generalisation of the Feldman & Compton HT1-QMOD system for hypothesis testing in neuroendocrinology [8, 9]. HT1-QMOD was a prototype system that was specific for neuroendocrinology which could not explain `steady` values or multiple causes.

attaching a volt meter to an electric circuit:

- In one extreme case, 300,000 sheeps brains had to be filtered to extract 1.0 milligrams of purified thyroptin-releasing hormone [12].
- In the usual case, delicate measurements have to be made by skilled staff using expensive equipment. Some of the values measured are in the pico-MOLE range ($10^{-12}M$).

Measurement in this domain can therefore be an expensive process and not all entities are fully measured. For example, in studies [8, 17] of a sketch of a model of glucose regulation [23], data was collected from six journal articles. This sketch was represented as a compartmental model [2]. In all, none of the flow constants were known and only 39% of the compartments were measured and not all at the same time interval.

2.1.2 Black-Box Comprehension

Operations Research makes extensive use of large complex simulation models. These can be the result of many person-years of development and incorporate modules obtained from external sources. It is common that such a system may be obtained from a third party. Typically, some group called $TEAM_1$ derives some model $\mathcal{M}_1$ representing their *initial understanding* of a problem (e.g. modeling the performance of a missile system). This model is *operationalised* in some third generation language to become $\mathcal{M}_2$. Perhaps an attempt is made to document $\mathcal{M}_1$ in a *manual* $\mathcal{M}_3$ which may be incomplete. $\mathcal{M}_2$ and $\mathcal{M}_3$ are then shipped to another site where a second team ($TEAM_2$) tries to understand them. Conceptually, $TEAM_2$ builds $\mathcal{M}_4$, a model representing the *local understanding* of $\mathcal{M}_2$ and the incomplete $\mathcal{M}_3$.

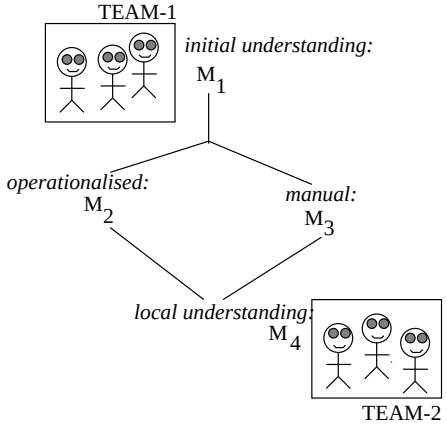


Figure 4: Commissioning Remote Models

[2] Compartmental models utilise the principal of conservation of mass and assume that the sum of flows of substance in and out of a compartment must equal zero. Flows are typically modeled using a time-dependent exponential function since the rate of flow is often proportional to the amount of stuff in the compartment [14].

The effort required to use $\mathcal{M}_2$ with confidence can be non-trivial, especially when object code is supplied without source code or access to the author. Current practice is for $\mathcal{M}_4$ to be documented in an incomplete manner (e.g. some procedural manual advising parametric sensitivity and constants relating to the local physical and operational environment). $\mathcal{M}_2$ can be regarded as a *black-box model* that $TEAM_2$ must convert (with some support from $\mathcal{M}_3$) into a *gray-box model* $\mathcal{M}_4$. Once validated, $\mathcal{M}_4$ would be used for planning, prediction, and optimisation studies. However, first it is essential that we can validate the vague model $\mathcal{M}_4$ since local conditions may invalidate $\mathcal{M}_2$. For example, the Australian Defence Forces use aircraft in configurations that are different to how they are used overseas. Certain parameters representing important domain knowledge are stored in compiled numerical matrices and are inaccessible to $TEAM_2$. For example, these parameters may (i) be based on experimental data from tests in other climates or (ii) contain certain tacit assumptions about aircraft operation. Prior to relying on $\mathcal{M}_2$, $TEAM_2$ would like to validate this model under local conditions.

2.2 HT4: An Abductive Inference Engine

Elsewhere [18], we have given a precise overview of abduction. Here, we offer an approximate characterisation of abduction as the search for consistent subsets of some background theory that are relevant for achieving some goal. If multiple subsets can be extracted, then a preference operator (which we call $\mathcal{BEST}$ and Bylander *et. al.* [2] call the plausibility operator *pl*) must be applied to select the preferred subset(s). Abduction is different to deduction. In deductive inference, all consequences of known literals must be found by the inference procedure. In abduction, the inference engine can ignore parts of the background theory if they are not relevant for the current goal.

Abduction is a very general process. Abductive frameworks for various KBS tasks have been defined by numerous researchers; e.g. frame-based reasoning [20], Bayesian reasoning [2], diagnosis [6], explanation [13], and default reasoning [21]. We have also argued that abduction is also useful for planning, monitoring, verification, prediction, classification, qualitative reasoning [18], and validation [17]. The reason for this generality is that the processing of *subset extraction* is a synonym of known general frameworks for expert systems; i.e. Breuker's [1] components of solutions and Clancey's [4] situation-specific models [18]. For example, the formal difference between heuristic classification and heuristic construction [3] is the extraction of one subset or multiple conflicting subsets, respectively.

2.2.1 An Example

In order to present a detailed example of abduction, we will run an example through our HT4 abductive inference engine [18]. A portion of Figure 1 is shown in Figure 5. The edge symbols of Figure 5 are explained in Figure 6.

In the case where $\mathcal{IN}$=`{foriegnSalesUp, domesticSalesDown}` and $\mathcal{OUT}$=`{investorConfidenceUp, inflationDown, wagesDown}`, we can generate the following proofs $\mathcal{P}$ for $\mathcal{OUT}$:

- $\mathcal{P}_1$= `foriegnSalesUp` $\rightarrow$ `companyProfitsUp` $\rightarrow$ `investorConfidenceUp`;

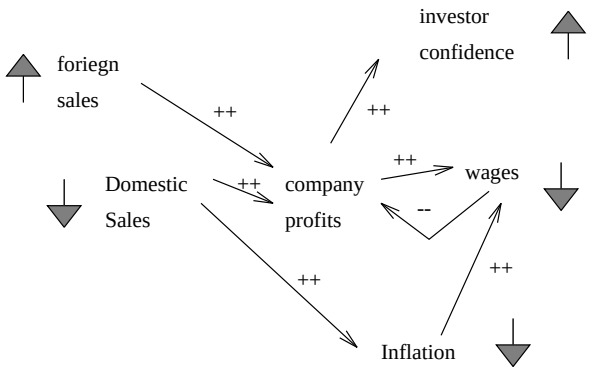


Figure 5: A portion of Figure 1.

Type	Statement	Notation	Can be used to represent...
direct	The more `X` the more `Y` and the less `X` the less `Y`	`X` $\stackrel{++}{\rightarrow}$ `Y`	mathematical proportionality
inverse	The more `X` the less `Y` and the less `X` the more `Y`	`X` $\stackrel{--}{\rightarrow}$ `Y`	mathematical inverse proportionality

Figure 6: Edge symbols of Figure 1 and Figure 5.

- $\mathcal{P}_2$= `domesticSalesDown` $\rightarrow$ `inflationDown`;
- $\mathcal{P}_3$= `domesticSalesDown` $\rightarrow$ `companyProfitDown` $\rightarrow$ `wagesDown`;
- $\mathcal{P}_4$= `domesticSalesDown` $\rightarrow$ `inflationDown` $\rightarrow$ `wagesDown`.

These proofs may contain assumptions, i.e. literals that are not known $\mathcal{FACTS}$. Continuing the example of Figure 5, if $\mathcal{FACTS}$=$\mathcal{IN} \cup \mathcal{OUT}$, then {`companyProfitsUp`, `companyProfitsDown`} are assumptions. If we can't believe that a variable can go `up` and `down` simultaneously, then we can declare these assumptions to be conflicting (denoted $\mathcal{A}_c$). Hence the key conflicting assumptions are the controversial assumptions which are not dependent on other controversial assumptions. In our example, these *base controversial assumptions* (denoted $\mathcal{A}_b$) are all of our controversial assumptions. We can used $\mathcal{A}_b$ to find consistent belief sets called worlds $\mathcal{W}$. A proof $\mathcal{P}_i$ is in $\mathcal{W}_j$ if that proof does not conflict with the environment $\mathcal{ENV}_j$ (an environment is a maximal consistent subset of $\mathcal{A}_b$). In our example, $\mathcal{ENV}_1$={`companyProfitsUp`}, $\mathcal{ENV}_2$={`companyProfitsDown`}. Hence, $\mathcal{W}_1$={$\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_4$} and $\mathcal{W}_2$={$\mathcal{P}_2$, $\mathcal{P}_3$ $\mathcal{P}_4$} (see Figure 7)[3].

Which is the $\mathcal{BEST}$ world? We have argued elsewhere [18] that, depending on the choice of $\mathcal{BEST}$ operators, a variety of KBS tasks can be emulated. $\mathcal{BEST}$ operators can be defined for one of three levels: vertex, proof, and world. When extending a proof from a particular node, a *vertex-level* $\mathcal{BEST}$ could elect to use edges with the highest numerical rating. This could be used to emulate a

[3]The connection of HT4 to DeKleer's ATMS system [7] is explored elsewhere [18]

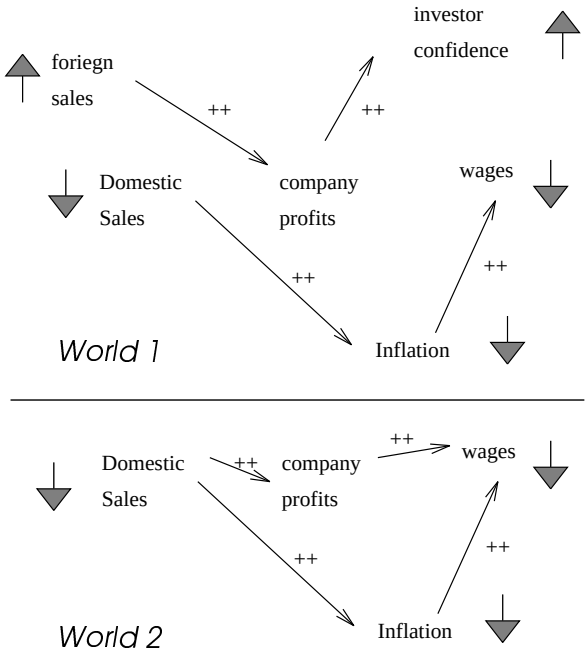


Figure 7: Worlds from Figure 5.

MYCIN-style certainty factor traversal. A *proof-level* $\mathcal{BEST}$ could execute when some proofs or partial proofs are known; e.g. beam search during planning. A *worlds-level* $\mathcal{BEST}$ could execute when the worlds are known. Minimal-fault diagnosis is a world-level $\mathcal{BEST}$ assessment operator that returns world(s) which maximise the number of $\mathcal{OUT}$puts obtained while minimising the number of $\mathcal{IN}$puts to those world. $\mathcal{BEST}_{EXPLANATION}$ returns the worlds(s) with the greatest overlap with known user profiles; i.e. return the worlds a particular user is liable to understand. Our preferred validation approach is a worlds-level $\mathcal{BEST}$ that returns the worlds that offer explanations for the most number of $\mathcal{OUT}$puts. This $\mathcal{BEST}_{VALIDATION}$ would therefore favour $\mathcal{W}_1$ (which explains 3 members of $\mathcal{OUT}$) over $\mathcal{W}_2$ (which only explains 2 members of $\mathcal{OUT}$).

2.2.2 Computational Limits of Testing

The computational limits of the test engine constrains what can be represented in vague modeling. Elsewhere, we have explored the practicality of HT4 for medium to large theories [17]. In our *mutation study*, 257 theories were artificially generated by adding random vertices and edges to the and-or graph from a theory called Smythe '89[4]. These were run using 5970 experimental comparisons. In the *changing* N mutation study, the fanout was kept constant and the size of the model was increased. Model size N was measured by the number of vertices $\mathcal{V}$ in the *ground graph*. For example, `foriegnSales` contributes three vertices to the ground graph: {`foriegnSalesUp, foriegnSalesSteady,`

[4]The Smythe '89 theory is discussed further in §3.5

foriegnSalesDown}. For that study, a "give up" time of 840 seconds was built into HT4. HT4 did not terminate for $|\mathcal{V}| \geq 840$ in under that "give up" time. We conclude that the "knee" in the exponential runtime curve kicks-in at around 800 literals. These result came from a Smalltalk V implementation on a Macintosh Powerbook 170. A port to "C" on a Sparc Station is underway.

The *changing fanout* mutation study examined the practicality of HT4 for models of varying fanout (fanout is the number of edges $\mathcal{E}$ that touch each vertex $\mathcal{V}$; i.e. $\frac{|\mathcal{E}|}{|\mathcal{V}|}$). In that study, the Smythe '89 theory size was kept constant, but edges were added at random to produce new graphs of larger fanouts. 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 [17].

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

Figure 8: Parameters from fielded expert systems. From [19].

In practice, how restrictive is a limit of 850 vertices and $\frac{|\mathcal{E}|}{|\mathcal{V}|} < 6.8$? The only reliable data we could find on fielded expert systems is shown in Figure 8 [19]. Figure 8 suggests that a practical inference engine must work at least for the range $55 \geq |\mathcal{V}| \geq 510$ and $2 \geq \frac{|\mathcal{E}|}{|\mathcal{V}|} \geq 7$. We therefore conclude that vague modeling is practical for the size of expert systems seen in current practice and for nearly the range of fanouts seen in fielded expert systems. However, after a certain level of inter-connectivity, 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 point where % $\mathcal{OUT}$ *covered* approaches 100%, vague modeling becomes impractical.

3 The Design Activity Cycle of Vague Modeling

Above, we have defined vague domains and the abductive framework that is the backbone of the process. We have explored the computational limits of validation in that framework. We can show where the testing process constrains the various stages of the design process; i.e. elicitation (§3.1), representation (§3.2), transformation (§3.3), verification (§3.4), and validation (§3.5).

3.1 Elicitation

The elicitation stage is an exploration of the domain. The goal of this stage is to make a first pass at discovering (i) list of terms for "things" in this domain; (ii)

a rough idea of how these "things" influence each other; (iii) what "things" are incompatible (the invariants); (iv) a set of scenarios recording what behaviour your models should be able to reproduce.

In the elicitation, domain experts are encouraged to sketch out vague causal diagrams like Figure 1. Note that these models will be under-specified and can generate conflicts. At this stage, experts should not try and resolve these conflicts. Rather, their goal is to generate the set of all possible explanations that are acceptable to them. The process of extracting relevant and consistent subsets will be handled later.

Once drawn, the next stage in vague modeling is to walk through the diagrams with *scenarios*; i.e. examples of how things impact the system. These scenarios will later become our $\mathcal{IN}$puts and our $\mathcal{OUT}$puts.

Tacit in the vague diagrams is *invariant knowledge*. For example, having said that `inflation` can go `up` or `down` is the statement that we can't believe that `inflation` can go `up` and `down` simultaneously. More generally, these diagrams can include invariants between any pairs of states. Invariants can be *absolute* or *gradual*:

- *Absolute invariants* are statements like `rain` and `suntan` are mutually exclusive (denoted `rain=up` $\overset{\otimes}{\leftrightarrow}$ `suntan=up`.
- *Gradual invariants* are statements like the more `rain` the less `suntan`. These are modeled using our standard edge types; e.g. `rain` $\overset{--}{\rightarrow}$ `suntan`.

When working with sensor data, it may be useful to introduce *summary* nodes that generalise certain data. For example, `prolongedExposure` to sunlight and `sunLamps` might both connect to `tanned`; e.g. `prolongedExposure` $\overset{++}{\rightarrow}$ `tanned` and `sunLamps` $\overset{++}{\rightarrow}$ `tanned`.

Vague causal diagrams can be generated from many pre-existing sources such as:

- Existing business rules; i.e. preconditions lead to postconditions.
- Equations; i.e. using `A` $\overset{++}{\rightarrow}$ `B` to model proportionality and `A` $\overset{--}{\rightarrow}$ `D` to model inverse proportionality.
- Object-oriented models; i.e. instance attributes imply that instance's class, sub-classes imply super-classes and super-classes plus certain added features imply a sub-class. Note that in standard OO, sibling classes are mutually exclusive (see Figure 9).

Domains typically have pre-existing special constructs which may be useful in elicitation of the vague causal diagrams. For example, neuroendocrinology using compartmental models. In compartmental models, compartments are connected by "pipes" and each pipe has a flow control box. These flow rate through these flow controllers can be changed by other compartments; e.g. `glucose` (a compartment) can increase the `glucagonProduction` (a flow controller on a pipe going into `glucagon`); i.e. `glucose` $\overset{++}{\rightarrow}$ `glucagonProduction` and `glucagonProduction` $\overset{++}{\longrightarrow}$ `glucagon`.

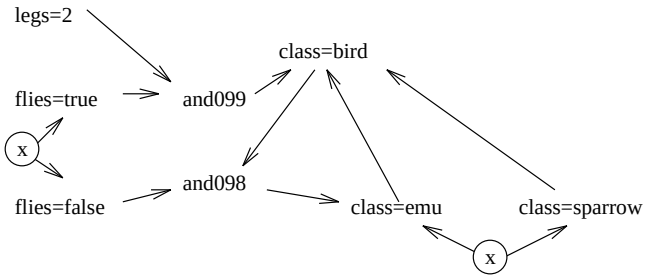


Figure 9: About birds.

3.1.1 Restrictions on Elicitation

Details of the test engine constrain the elicitation stage. In particular, we must restrict the size of vague models. In practice, this means avoiding *low-level detail* and *removing time*.

Avoiding low-level detail: Vague modeling is not a technique for recording of low-level detail since the HT4 inference engine has problems handling very big models (recall §2.2.2). Hence, if models get too big, we need to move up a level of abstraction. For example, collapse `sales to china` and `sales to canada` to one "thing": `foriegnSales`. In the neuroendocrinological domain, we adopted the usual qualitative modeling trick of replacing continuous variables with `up, down, Steady`; i.e. one of three states representing the sign of the change in those variables. Note that this introduces some indeterminacy into the model. Recall that in Figure 1, if `foriegnSales` and `domesticSales` go `up` and `down` respectively, then we have contradictory possible values for `companyProfits` (`up` *OR* `down`). In vague modeling, we defer the resolution of these conflicts till the testing phase. In the meantime, recall that if HT4 generates multiple worlds for `companyProfitsUp or companyProfitsDown`, then these contradictory assumptions will be stored in multiple worlds.

Removing time:

- We could naively model looping in simulations by repeating each node in a theory for every time tick in the simulation. That is, the graph would contain a node $\mathtt{a}_{t=1}$, $\mathtt{a}_{t=2}$, $\mathtt{a}_{t=3}$, ... Given T time ticks, the practical limit to model size is the limits noted in §2.2.2, divided by T. That is, the longer the simulation, the smaller the model we can test.

- A non-naive alternative is to repeat each loop only for *measured* time ticks. We are exploring this option, but this is only preliminary work-in-progress.

In practice, for the current work, we plan to build our models without time. In the neuroendocrinological domain, time is removed by only using scenarios without time series data. This restriction still provides us with many data points. Much of the experimentation in neuroendocrinology is performed on separate groups of rats. Conclusions are made by comparing measurements in some control group to measurements in some treated group. In such scenarios, there is no sense of time flowing from the control group to the treated group.

In the black-box comprehension domain, time is harder to remove. Many of the systems in the target domain are large FORTRAN or C++ simulations which explicitly model time-series events. Our current approach in this domain is to develop an elicitation framework aimed at declarative representation. Time and position are often inter-changeable indexes in the navigation strategies of mariners and pilots. Traditional AI planning (e.g. STRIPS) avoids explicit representation of time by creating ordered lists representing priorities in schedules. We conjecture that pilot knowledge expressed in the this prioritised list framework, need not mention time. Rather, our rules will be reactions to some immediate situation rather than a reflection of some between-situations plan.

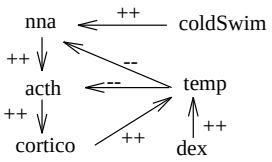
3.2 Representation

The representation stage fills in the details sketched in the elicitation stage. The goal of this stage is creation of (i) a *theory* (§3.2.1); (ii) some *observations* containing the scenarios that the theory should be able to reproduce (§3.2.2); and the *success criteria* (§3.2.3).

3.2.1 Representing a Theory

Smythe '87 [22] is a theory proposing connections between serum adrenocorticotropin (`acth`), serum corticosterone (`cortico`), and neuro-noradrenergic activity (`nna`). `Nna` was measured as the ratio of noradrenaline to its post-cursor, 3,4-dihydroxphenyl-ethethyleneglycol. This theory is shown as the vague causal diagram of Figure 10. The `temp` vertex was introduced to model the user's statement that `dex` has the same effect as `cortico`.

A. Causal diagram



B. Theory

```
name       = ´16. Smythe 87´.
objects(e) = [coldSwim,dex].
objects(m) = [nna, acth, cortico,temp].

coldSwim ++ nna.      nna     ++ acth.
acth     ++ cortico.  cortico ++ temp.
temp     -- acth.     temp    -- nna.
dex      ++ temp.
```

Figure 10: The Smythe '87 theory

The Smythe '87 theory is shown in Figure 10 in HT4 syntax. This theory starts with a `name = ´a string´` statement and is followed by two `object` statements. The `object` statements describe the `events` and the `measures` of the diagram. Events (denoted `objects(e)`) are like booleans while measures (denoted `objects(m)`) are for continuous variables. The body, which follows the head, contains the influence statements. The influence statements are pairs of measures connected by one of the topoi symbols (see Figure 6).

3.2.2 Observations

The Smythe '87 paper contains values comparing these values in various experiments: (i) *control* i.e. no treatments; (ii) `dex` i.e. an injection of dexamethasone

at 100 $\frac{mg}{kg}$; (iii) `coldSwim` i.e. a two minute swim in a bath of ice cold water; and (iv) `coldSwim, dex` i.e. both a `coldSwim` and an injection of `dex`. A sample of experimental results from Smythe '87 is shown as the observations of Figure 11 (in HT4 syntax). The experiments are shown in the columns and the measures taken in the different experiments are shown in the rows.

```
observations  =  [    [],    [dex], [coldSwim], [dex,coldSwim]]
              =
     [nna     - [  0.122,    0.105,      0.210,     0.246    ],
      cortico - [ 129    ,   11.3  ,   1232.0  ,    32.8      ],
      acth    - [  89    ,    0.0  ,    240.0  ,     0.0      ]].
```

Figure 11: The observations for the Smythe '87 model of Figure 10.

Our *scenarios* are comparisons between two experiments. When we compare (e.g.) `[coldSwim]` to `[dex, coldSwim]`, we say that the input ($\mathcal{IN}$) to scenario was `change(coldSwim)=arrived`, `change(dex)=arrived` and the output ($\mathcal{OUT}$) of the scenario was that `acth=down, cortico=down, nna=up`. The scenario that our theory has to try and reproduce is `change(coldSwim)=arrived` and `change(dex)=arrived` leads to `nna=up`; that is cold swims and injections of steroids increases stress levels in a rat.

3.2.3 The Success Criteria

The success criteria will be mapped into the HT4 $\mathcal{BEST}$ operator. We use the success criteria to select preferred inferences. Success criteria extract summaries of the inferences, and then cull some inferences based on those summaries. The exact $\mathcal{BEST}$ operator is domain dependent. For neuroendocrinology, we have used a $\mathcal{BEST}$ operator that returns the world(s) that cover the most $\mathcal{OUT}$puts. Note that this answers the following question: "can a theory of X reproduce known behaviour of X?". Elsewhere (§3.5) we describe experiments were this was applied as a validation criteria which successful found faults in published models of neuroendocrinology. For other domains, (e.g. machine learning) we may chose to (i) return the smaller worlds; (ii) the world(s) that make the fewest assumptions; that (iii) use the least controversial assumptions; (iv) the "cheapest" worlds (assuming that each edge is associated with some notion of the cost of its inclusion); (v) etc.

3.2.4 Restrictions on Representation

The details of what can be represented are restricted by what can be handled at the transformation stage.

3.3 Physical Design: Transformation

The physical design is the bridge between the logical design and an executable version which we can test. Specifically, the goals of the *transformation* stage is to be able to:

- Execute the theory. This implies the use of a *model compiler* (see §3.3.1) which can generate the ground dependency graph from the theory.
- Convert the observations into sets of $\mathcal{IN}$put-$\mathcal{OUT}$put pairs. Each pair of $<\mathcal{IN},\mathcal{OUT}>$ is then passed to the output of the *data compiler* (see §3.3.2) for execution.
- Implement the success criteria to resolve any conflicts found during the execution. When implemented, the success criteria is called the $\mathcal{BEST}$ assessment operator. $\mathcal{BEST}$ operators have to be coded in the HT4 implementation language (Prolog).

3.3.1 The Model Compiler

Two styles of model-compilers are planned for the black-box comprehension domain. The first style is a generic partial-evaluator of rules with variables into the network of connections between literals in the rule. This partial-evaluator will need knowledge of the range of each field of each fact that can be asserted. The second style will be an attempt to "shoe-horn" the rule representation into the representation used in the neuroendocrinological domain. A hand inspection of the rules will be used to draw vague causal diagrams which will be mapped into the neuroendocrinological theory.

With the neuroendocrinological model compiler, if a vertex can be in one of N states, then one vertex is added for each such state. If an edge condones connections between two states of adjacent vertices, then an edge is added between those states. For example, `power ++ lights` is expanded into Figure 12.

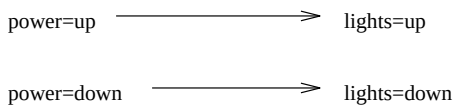


Figure 12: Light is directly proportional to power.

All the vertices of Figure 12 are `or` vertices; i.e. belief in these vertices requires a belief in only one of its parents. The model compiler can add the disabling effects of `rats` via `and` vertices; i.e. vertices which we can only believe if we believe all their parents. For example, if we know that the rats in the basement disable the connection between the power and the lights (when they bump the cable), then we say `if not rats then power ++ lights` which expands into Figure 13.

At this point, the neuroendocrinological model compiler gets intricate. A conjunction of competing upstream influences can combine to explain a steady vertex. Also, changes to an object's value downstream of an abler link can be explained in terms of changes to the abler. Returning to our rats, in the case of power not rising (but on) and the rats being present, the lights are dark. Now consider the same situation, but the rats suddenly disappearing. The lights going up can now be explained in terms of a change in the rat population. If we take these into account, we get Figure 14. In Figure 14 we include for the first time the incompatible vertices (see the double-headed arrows marked with a cross).

For more details on the neuroendocrinology model compiler, see [16].

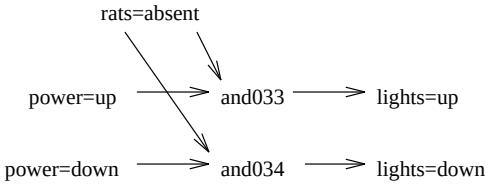


Figure 13: The lights/power relationship works when rats are absent.

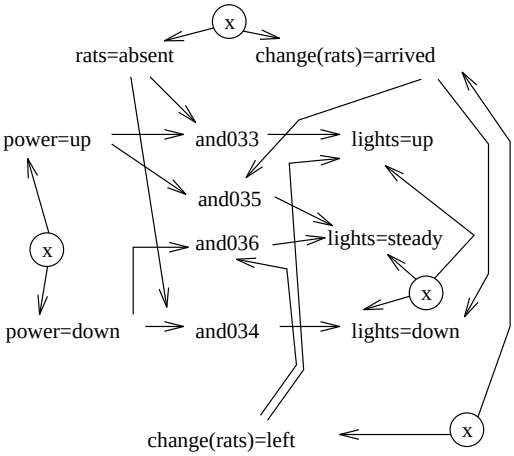


Figure 14: The final rats model.

3.3.2 The Data Compiler

The data compiler converts tables like Figure 11 into $\mathcal{IN}$ and $\mathcal{OUT}$ sets of vertices in the dependency graph. For example, in the scenario `[]` to `[dex]` in Figure 11, $\mathcal{IN}_1$= `dex=arrived, coldSwim=absent` and $\mathcal{OUT}_2$= `nna=up, cortico=down, acth=down`.

There are five other scenarios: $<\mathcal{IN}_i,\mathcal{OUT}_i>$ pairs: `[]` to `[coldSwim]`, `[]` to `[dex,coldSwim]`, `[dex]` to `[coldSwim]`, `[dex]` to `[dex,coldSwim]`, and `[coldSwim]` to `[dex,coldSwim]`. Therefore, we can run six scenarios through the Smythe '87 model of Figure 10.

Smythe '87 is a *symmetrical model*. In symmetrical models, a comparison from $experiment_i$ to $experiment_j$ is the same as $experiment_j$ to $experiment_i$; i.e. after studying `[dex]` to `[coldSwim]`, we do not need to study `[coldSwim]` to `[dex]`. Non-symmetrical domains are those that include asymmetric influences. `Direct` and `inverse` are symmetric influences while ablers are asymmetric. In asymmetric domains, therefore, twice as many scenarios must be studied.

3.4 Verification

Verification is an exploration of the internal syntactic structure of a program. Preece, Shinghal and Batarekh [19] (hereafter, PSB) define a taxonomy of struc-

tural "anomalies" in rule-based expert systems (see Figure 15) and argue that a variety of verification tools target different subsets of these anomalies (perhaps using different terminology).

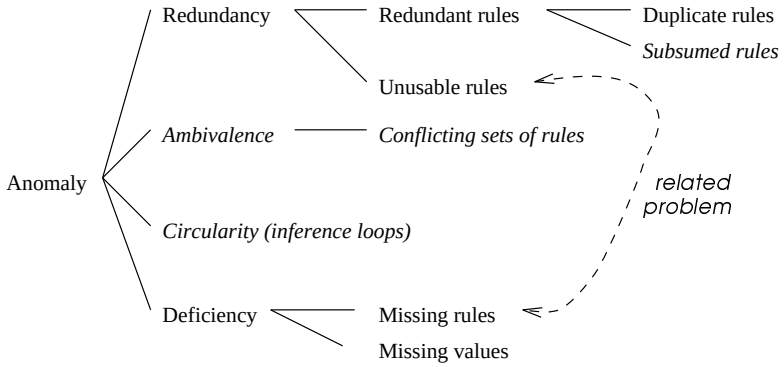


Figure 15: PSB anomalies.

Some of these anomalies require meta-knowledge about knowledge base literals:

- A literal is *PSB-askable* if it represents a datum that the knowledge base can request from the outside world. All *askable*s must be events.
- A literal is a *PSB-final* hypothesis if it is declared to be so by the KB author and only appears in a rule conclusion. *Final* vertices may be events or measures.

A rule is *PSB-redundant* if the same final hypotheses are reachable if that rule was removed. An unusable redundant rule has some impossible premise. A knowledge base is *PSB-deficient* if a consistent subset of askables leads to zero final hypotheses. A *PSB-duplicate* redundant rule has a premise that is a subset of another rule premise. PSB define duplicate rules for the propositional case and subsumed redundant rules for first- order case (where instantiations have to be made to rule premise variables prior to testing for subsets). PSB define *ambivalence* as the case where, given a consistent subset of askables, a rule-base cannot infer some final hypotheses.

PSB stress that the entries in their taxonomy of KBS anomalies may not be true errors. For example, the dependency network from a rule-base may show a circularity anomaly between literals. However, this may not be a true error. Such circularities occur in (e.g.) user input routines that only terminate after the user has supplied valid input. For this reason, the "errors" detected by internal testing are called anomalies, not faults. Internal test anomalies are used as pointers into the system which direct the developer to areas that require a second glance.

In vague domains, the anomalies shown in *italics* of Figure 15 are to be expected. For example:

- There is a case that *subsumed rules* are actually desirable in the case where different audiences use different logic to reach the same conclusion. Such

different approaches to the same problem are very useful for explaining an inference to different people.

- PSB ambivalence can be reported when more than one world is generated at runtime. Note that in vague domains, ambivalence is to be expected.

However, a graph-theoretic analysis of the dependency graph $\mathcal{D}$ can detect many of the other PSB anomalies:

1. Compute the components of the dependency graph; i.e. the disconnected sub-graphs of $\mathcal{D}$. Find the roots and leaves of each component. Compute the set intersection and set difference of the askables with the roots and the final hypotheses with the leaves. An inspection of these sets will detect specification anomalies
2. Compute the transitive closure of each vertex using Warshall's algorithm. Report a circularity anomaly if a vertex is in its own transitive closure.

Other useful verification studies are:

- Generate an alphabetical list of all vertices in the system. A quick scan of this list can find certain common spelling mistakes.
- Instrument the model compilation process:
  - If a statement in the theory proposes a set of new edges for $\mathcal{D}$, and those edges are already in $\mathcal{D}$, then that statement is a duplicate of a previous statement.
  - If a statement in a theory proposes edges between incompatible vertices, then that statement is unusable.

3.5 Validation

Validation is an assessment of the program based on external semantic criteria. For each model generated from the logical design, we find:

- The average percent *coverage* of that model for all known behaviour using a worlds-level $\mathcal{BEST}$ assessment operator that favours world(s) that covers the most outputs (recall §3.2.3).
- The frequency with each edge in $\mathcal{D}$ was used when processing the behaviour. Edges that are never/rarely use can be reviewed with the business users and (possibly) culled from the model.

In the case of competing models, we reject the models that explain significantly less behaviour than their competitors. Returning to the Smythe '87 theory (Figure 10), in the scenario `[coldSwim]` to `[dex, coldSwim]`, we can infer from Figure 11 that $\mathcal{OUT}$=`{acthDown,corticoDown, nnaUp}` and $\mathcal{IN}$= `{change(dex)=arrived, coldSwim=present}`. In this comparison `nnaUp` can't be explained since their exists no proof from `nnaUp` to $\mathcal{IN}$ which does not violate the proof invariants. Another error can be found in the comparison `[dex]` to `[dex,coldSwim]`. In this comparison $\mathcal{IN}$= `{change(coldSwim)=arrived, dex=present}` and $\mathcal{OUT}$= `{acthSteady, corticoUp, nnaUp}` and only `nnaUp`

can be explained. What is interesting here is that the Smythe '87 theory had been published in an international refereed journal [22] and its validation errors were found by a detailed examination of the data published to support it! Further, when these errors were shown to the author of the theory, he found them novel and exciting; i.e. they were significant errors.

Smythe '87 is a small theory. The Smythe '89 study explores how well abductive validation scales up to medium-sized models. Smythe '89 [23] is a theory of human glucose regulation. It contains 27 possible inputs and 53 measurable entities which partially evaluated into an and-or graph with 554 vertices and 1257 edges. Smythe '89 is a review paper that summaries a range of papers from the field of neuroendocrinology. Those papers offer 870 experimental comparisons with between 1 to 4 inputs and 1 to 10 outputs.

Smythe '89 was originally studied by QMOD, the technical precursor to HT4. That study found that 32% of the observations were inexplicable. QMOD could not explain studies or handle multiple causes. These restrictions implied that it could only handle of 24 possible comparisons. Even with these restrictions, QMOD found several errors in Smythe '89 that were novel and exciting to Smythe himself [8]. Like the Smythe '87 study, these errors had not been detected previously by international peer review.

When HT4 ran over the full 870 comparisons, it found more errors than QMOD. Only 150 of the comparisons could explain 100% of their $\mathcal{OUT}$puts. On average, 45% of the $\mathcal{OUT}$s in those comparisons were inexplicable. The level of critique offered by QMOD and HT4 is surprisingly high. This is both a disturbing and exciting finding. It is disturbing in the sense that if the very first large-scale medical theory analysed by abductive validation contains significant numbers of errors, then it raises doubts as to the accuracy of theories in general. This result is exciting in the sense that the level of critique is so high. Abductive validation promises to be a powerful technique for theory review.

4 Related Work

Vague modeling makes little use of two of the dominant themes in KA: ontological engineering [10] and reusable problem solving strategies (e.g. KADS [24]). While an appropriate ontology may be useful for "kick-starting" the elicitation stage, they may not be productivity tools for the rest of the design activity life cycle in vague domains. Further, it is not clear what (e.g.) KADS offers for the testing phase of a KBS. Our framework easily supports validation and verification. While KADS has a "verification" inference model[5], in its current form it cannot be applied reflectively on the KADS models themselves. Also, this KADS "verification model" makes no mention to assumption space management; a technical issue that is central to our architecture and validation process.

5 Conclusion

Vague modeling is the process of making what sense we can of half-formed ideas in a poorly measured domain. In this paper we have explored the design activity

[5]In our view, KADS "verification" is really "validation" since it relies on external semantic criteria.

cycle of vague modeling based around vague causal diagrams and abductive inference. We have contrasted vague modeling with conventional modeling. In conventional modeling, most of the design is generated in the analysis stage. In vague modeling, most of the model is found during testing. Further, the limits of the computational device used in the testing phase restricts the modeling process back at the logical design phase. Hence, the test engine should be built *first* since, in vague domains, what can't be tested shouldn't be modeled. HT4 imposes two restrictions: the need to avoid low-level detail and the need for declarative representations that avoid time. Models that satisfy these restrictions can be verified and validated.

Originally, the approach described here was developed as a general approach, using the neuroendocrinology domain as an example. Our current task is to test the generality of this approach in the black-box comprehension domain. While the work done to date is preliminary, it has served to highlighted certain restrictions with the software developed for the neuroendocrinology domain. For example, we need better tools for building of model and data compilers (see §3.3.1 and §3.3.2). While ultimately these must be customised to a particular domain, model compilers for rules and frames should be added to the current system.

References

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

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

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

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

[5] P. Clark and S. Matwin. Using Qualitative Models to Guide Inductive Learning. In P. Utgoff, editor, *Proceedings of the Tenth International Machine Learning Conference, ML-93*, pages 49–56, 1993.

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

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

[8] 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.

[9] 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.

[10] T.R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[11] B. Hodgson. Personal communication. Hodgson derived his meta-model from a recent analysis of how CASE tools are used in business environments, 1996.

[12] D.T. Krieger. The Hypothalmus and Neuroendocrinology. In D.T. Krieger and J.C. Hughes, editors, *Neuroendocrinology*, pages 3–122. Sinauer Associates, Inc., 1980.

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

[14] J.E.A. McIntosh and R.P. McIntosh. *Mathematical Modeling and Computers in Endocrinology*. Springer-Verlag, 1980.

[15] 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.

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

[17] T.J. Menzies. On the Practicality of Abductive Validation. In *ECAI '96*, 1996.

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

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

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

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

[22] 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.

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

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

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