Generalised Test = Generalised Inference

Tim Menzies
Dept. of Software Development, Monash University, Australia
`timm@insect.sd.monash.edu.au`;
`http://www.sd.monash.edu.au/σtimm`

Abstract— **We are used to viewing verification and validation as tasks to be performed after a knowledge-base is built. In this standard view, the inference engine is built first, the knowledge base is then filled out, then finally a test engine is added to verify or validate the knowledge. Here, we reverse this order. First, we define a general test engine. Then, we argue that this "test" engine is also a general inference engine. That is, an architecture supporting a generalised test engine subsumes architectures that merely support inference.**

I. Introduction

The knowledge acquisition community can be divided into a conventional majority and a minority school of thought:

- In the majority school (e.g. KADS), KBS development is a *analysis intensive process* (AIP); i.e. before *doing*, much time is spent in *thinking about doing*. Further, previous *thinkings about doing* can be re-used for new designs.
- The minority situated cognition school [1–3, 9, 19] argues that concepts elicited prior to direct experience are *less* important than functional units developed via direct experience with the current problem. This situated cognition school prefers a *maintenance intensive process* (MIP); i.e. after a little *thinking about doing*, most of the system is developed via *doing* then *fixing*.

Note the importance of verification and validation in the maintenance intensive process:

- In the situated cognition view, using V&V tools drive the modeling process.
- Concepts that can't be verified or validated cannot be used. That is, limits to the V&V tools become limits to the modeling process.

Here, we explore computational architectures for general V&V. Fortunately, we find that a generalised abductive validation engine (HT4) is also a framework for handling common knowledge-level tasks; that is a generalised *test* engine is also a *inference* engine. 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 [6]. That is:

- $\mathcal{T} \cup \mathcal{A} \vdash \mathcal{OUT}$
- $\mathcal{T} \cup \mathcal{A} \not\vdash \perp$

The proof trees $\mathcal{P}$ used to satisfy these two equations can be cached and sorted into *worlds* $\mathcal{W}$: maximal consistent subsets (maximal with respect to size). Each world condones a set of inferences. A domain-specific $\mathcal{BEST}$ operator can then be used to return the world(s) that satisfy some criteria (e.g. shortest inference paths). Our general claim is that this abductive process describes both execution and testing.

Given that abduction can handle knowledge modeling and V&V, then we can use this framework to implement maintenance intensive processing.

This paper is structured as follows. Section II describes our HT4 [11] abductive inference engine. Section III argues that HT4 is a framework for validation and verification. Section IV argues that this framework can also operationalise many of the knowledge-level tasks seen in expert systems; i.e. prediction, classification, explanation, tutoring, qualitative reasoning, planning, monitoring, set-covering diagnosis, and consistency-based diagnosis. Section V discusses related work. Portions of this paper have appeared elsewhere [11, 12].

II. Abduction

In this section we expand on the description of abduction given in the introduction using our HT4 abductive framework.

Given a set of goal $\mathcal{OUT}$puts and known $\mathcal{IN}$puts, then HT4 can use (e.g.) the qualitative theory of Figure 1 to build a set of proof trees $\mathcal{P}$ connecting $\mathcal{IN}$puts to $\mathcal{OUT}$puts. In Figure 1:

- `x` $\overset{++}{\rightarrow}$ `y` denotes that `y` being `up` or `down` could be explained by `x` being `up` or `down` respectively
- `x` $\overset{--}{\rightarrow}$ `y` denotes that `y` being `up` or `down` could be explained by `x` being `down` or `up` respectively.

If we assume that:

- 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),

then we can partially evaluate Figure 1 into the and-or graph of literals of Figure 2. This and-or graph contains one vertex for each possible state of the nodes of Figure 1 as well as *and* vertices which models combinations of influences (for example, `gDown` and `bDown` can lead to `fSteady`).

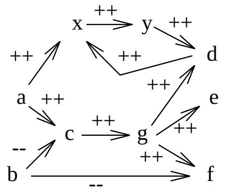

Fig. 1. A qualitative theory.

For example, in the case where $\mathcal{OUT}$ = {`dUp`,`eUp`,`fDown`} and $\mathcal{IN}$={`aUp`,`bUp`}, then all the possible proofs are:

- $\mathcal{P}_1$= `aUp` → `xUp` → `yUp` → `dUp`
- $\mathcal{P}_2$= `aUp` → `cUp` → `gUp` → `dUp`
- $\mathcal{P}_3$= `aUp` → `cUp` → `gUp` → `eUp`
- $\mathcal{P}_4$= `bUp` → `cDown` → `gDown` → `fDown`
- $\mathcal{P}_5$= `bUp` → `fDown`

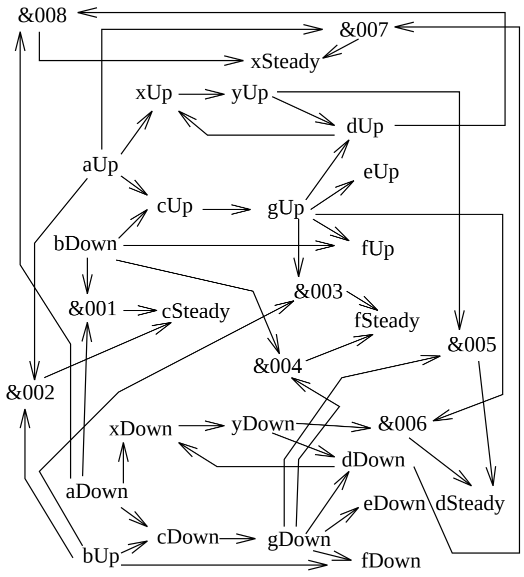

Fig. 2. The search space tacit in Figure 1

Some of these proofs make assumptions; i.e. use a literal that is not one of the known $\mathcal{FACTS}$ (typically, $\mathcal{FACTS} = \mathcal{IN} \cup \mathcal{OUT}$). Note that some of the assumptions will contradict other assumptions and will be *controversial*(denoted $\mathcal{A}_C$). For example, assuming cDown and cUp at the same time is contradictory.

In terms of uniquely defining an assumption space, the key controversial assumptions are those controversial assumptions that are not dependent on other controversial assumptions. We denote these *base* controversial assumptions $\mathcal{A}_B$. In our example, $\mathcal{A}_C$={cUp,cDown,gUp,gDown} and $\mathcal{A}_B$ = {cUp, cDown} (since Figure 1 tells us that g is fully determined by c).

If we assume cUp, then we can believe in the *world* $\mathcal{W}_1$ containing the proofs $\mathcal{P}_1$ $\mathcal{P}_2$ $\mathcal{P}_3$ $\mathcal{P}_5$ since those proofs do not assume cUp. If we assume cDown, then we can believe in the world $\mathcal{W}_2$ containing the proofs $\mathcal{P}_1$ $\mathcal{P}_4$ $\mathcal{P}_5$ since these proofs do not assume cDown. These worlds are shown in Figure 3. Note that each world is merely a subset of the edges shown in Figure 2.

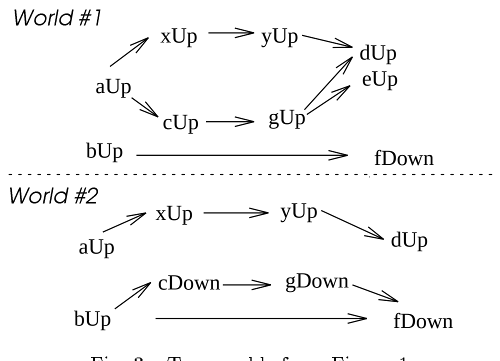

Fig. 3. Two worlds from Figure 1

The overlap of $\mathcal{W}_1$ and $\mathcal{OUT}$ is {dUp,eUp,fDown} and the overlap $\mathcal{W}_2$ and $\mathcal{OUT}$ is {dUp,fDown}; i.e. $\mathcal{W}_1^{cover} = 3 = 100\%$ and $\mathcal{W}_2^{cover} = 2 = 67\%$. The maximum cover is 100%; i.e.:

- There exist a set of assumptions ({cUp}) which let us explain all of $\mathcal{OUT}$.
- This theory has passed abductive validation.

The complexity of the above process is a function of when we apply the $\mathcal{BEST}$ inference assessment operator to cull generated inferences. A *vertex-level* $\mathcal{BEST}$ could execute at the local-propagation level; e.g. use the edges with the highest probability. A *proof-level* $\mathcal{BEST}$ could execute when some proofs or partial proofs are known; e.g. beam search. A *worlds-level* could execute when the worlds are known. For example, our abductive validation algorithm [12] returns the world(s) with the maximum cover.

For more details on the internals of HT4, see [11].

III. Abduction as a Test Engine

A. 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 favors the worlds with largest number of covered outputs (i.e. maximise $\mathcal{IN} \cap \mathcal{W}_x$) [12].

Note that this process can be summarised as: "can a theory of X explain known behaviour of X?". We have argued elsewhere [10] that this is the non-naive implementation of KBS validation since it handles certain interesting cases:

- If a theory is globally inconsistent, but contains local portions that are consistent and useful for explaining some behaviour, HT4 will find those portions.
- In the situation where no current theory explains all known behaviour, competing theories can be assessed by the extent to which they cover known behaviour. Theory X is definitely better than theory Y if theory X explains far more behaviour than theory Y.

Elsewhere, we have shown examples where this framework has faulted theories published in the international peer-reviewed literature. Interesting, we have found these faults using the data published to support those theories [12].

B. Verification

KBS *verification* tests a theory's validity against internal syntactic criteria [15]. 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. That is, given a set of $\mathcal{IN}$puts, mutually exclusive conclusions can be made.
- *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 $\mathcal{I}$).

IV. Abduction as a Inference Engine

A. Prediction

Prediction is the process of seeing what will follow from some events $\mathcal{IN}$. We can find all the non-input vertices reachable from $\mathcal{IN}$ by making $\mathcal{OUT} \subseteq \mathcal{V} - \mathcal{IN}$. A efficient case for prediction is when $\mathcal{IN}$ 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. $\mathcal{OUT} \subset \mathcal{V} - \mathcal{IN}$).

B. Classification

Classification is just a special case of prediction with the *interesting subset* set to the vertices representing the possible classifications. The and-or graph of a classification theory would include edges (i) from class attributes to the proposition that some class is true; and (ii) from sub-classes to super-classes (e.g. if emu then bird). $\mathcal{BEST}_{CLASSIFICATION}$ could favors the worlds that include the most-specific classes [14] (e.g. emu is better than bird).

C. Explanation and Tutoring

If we have a profile of our users comprising vertices familiar to the user and the edges representing processes that the user is aware of, then we can build an *explanation* and *tutoring system*. $\mathcal{BEST}_{EXPLANATION}$ could favors the worlds with the largest intersection to this user profile. That is, we return the world(s) that the user is most likely to understand. We base this explanation proposal on analogous work by Paris [13].

Further, suppose we can assess that the $\mathcal{BEST}$ explainable world was somehow sub-optimum. We could then make a entry is some log of teaching goals that we need to educate our user about the edges which are not in their $\mathcal{BEST}$ explainable world but are in other, more optimum, worlds.

D. Qualitative Reasoning

HT4 was developed from a *qualitative reasoning* algorithm for neuroendocrinology [7]. A fundamental property of such systems is their indeterminacy which generate alternative values for variables. These alternatives and their consequences must be considered separately. Abduction can maintain these alternatives in separate worlds.

E. Planning

Planning is the search for a set of operators that convert some current state into a goal state. Given a set of operators, we could partially evaluate them into the dependency graph they propose between literals. $\mathcal{BEST}_{PLANNING}$ could favor the world(s) with the least cost (the cost of a world is the maximum cost of the proofs in that world). If we augment each edge with the identifier of the operator(s) that generated it, then we could report HT4's $\mathcal{BEST}$ worlds as the union of the operators that generated the $\mathcal{BEST}$ worlds.

F. Monitoring

Once generated, the $\mathcal{BEST}$ planning worlds could be passed to a *monitoring* system. As new information comes to light, some of these assumptions made by our planner will prove to be invalid. Hence, some of our worlds (a.k.a. plans) will also be invalid. The remaining plans represent the space of possible ways to achieve the desired goals in the current situation.

G. Diagnosis

Parsimonious *set-covering diagnosis* [16] 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 $\mathcal{W}_x \cap \mathcal{IN}$).

The opposite of set-covering diagnosis is *consistency-based diagnosis* [4,17] where all worlds consistent with the current observations are generated. Computationally, this is equivalent to the *prediction* process described above, with $\mathcal{OUT} = \mathcal{V} - \mathcal{IN}$.

In Reiter's variant on consistency-based diagnosis [17], 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.

H. Probing

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 [5]. In HT4, we would know to favor probes of $\mathcal{A}_B$ over probes of $\mathcal{A}_C$ over probes of non-controversial assumptions.

V. Related Work

We are not the first to argue that non-monotonic techniques are useful for KBS validation. For example, Ginsberg [8] and Zlatereva [22] automatically build test suites that exercise the entire KBS from a TMS-style analysis of the dependency graph between KB literals. Our contribution here is to demonstrate that non-monotonic architectures are useful for much more than just validation. DeKleer's ATMS system made a similar claim in the 1980s.

Generic design principles for expert system design are proposed by the KADS community (e.g. [20, 21]. In the KADS approach, a library of abstract *interpretation models* for common KBS tasks such as classification and diagnosis are used to guide KBS design. Workers in that field have noted non-trivial similarities between the different interpretation models. For example, Tansley & Hayball [20] note that that scheduling, planning, and configuration are actually the same problem, divided on two dimensions ("goal states known or not" and ""temporal factors considered or not"). Also, Wielinga *et. al.* [21] acknowledge certain similarities between the inference procedures such as diagnosis and monitoring. However, having noted some low-level similarities between the interpretation models that they have proposed, KADS researchers do not take the next step and simplify their distinctions according to these observed similarities.

We view the state space traversal of SOAR [18] as a directed and-or graph which can be extended at runtime. While an HT4 vertex contains a single literal, the vertices of the SOAR state space contain conjunctions of literals. We prefer HT4 approach over SOAR for two reasons. HT4 knowledge bases can be validated without additional architecture. In other expert systems approaches (e.g. SOAR), validation requires additional architecture. Also, HT4 is a less complicated architecture than SOAR. SOAR is built on top of

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

if   weather = raining and football = on
then watchTV

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

Fig. 4. Tuesday can be washing day or football day, but not both.

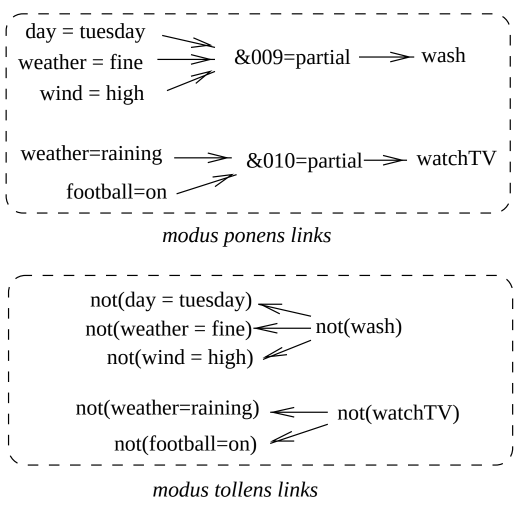

Fig. 5. And-or graph for Tuesday, washing, and football. Note the added modus tollens links.

an intricate forward-chaining rule-based system. HT4 uses a simpler graph-theoretic approach.

VI. Discussion

Our example here was based on qualitative reasoning. However, note that the internal data structure of HT4 (recall Figure 1) is an and-or graph of literals. HT4-style abduction executes over any theory that can be reduced to such a and-or graph of literals. Many representations can be mapped into this form. Our example above was from qualitative theories but the technique could be applied to other representations. For example, the rules of Figure 4 can be converted into the and-or graph of Figure 5.

Similar and-or graphs can be generated from frame-based systems. For example, Figure 7 shows the and-or graph tacit in the frame-based theory of Figure 6; That is 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.

More generally, any first-order theory that can be unfolded in a finite number of steps to a ground theory can be processed in this framework.

VII. Conclusion

The distinction between the test engine and the inference engine is unnecessary. We find that numerous, seemingly different, knowledge-level tasks can be mapped into a single abductive inference procedure. Also, we find that abduction can be used for KBS validation and verification. If we implement expert sys-

```
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]).
```

Fig. 6. Things that fly and walk.

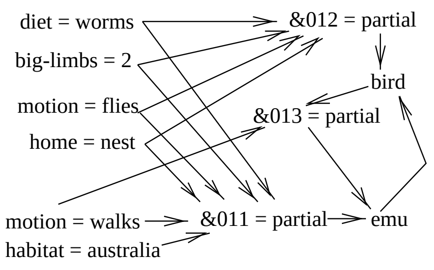

Fig. 7. Things that fly and walk: and-or graph. For the sake of simplicity, modus tollens links not shown.

tems as abduction, then we can execute *and* evaluate our knowledge bases within the same framework. This is a very important conclusion for advocates of situated cognition (like ourselves) since we can use design concepts evolved for analysis intensive processing (e.g. knowledge level modeling) for mainteance intensive processing.

References

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

[2] W. Clancey. Situated Action: A Neuropsychological Interpretation Response to Vera and Simon. *Cognitive Science*, 17:87–116, 1993.

[3] P.J. Compton and R. Jansen. A Philosophical Basis for Knowledge Acquisition. *Knowledge Acquisition*, 2:241–257, 1990.

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

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

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

[7] 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.

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

[9] D. McDermott. A Critique of Pure Reason. *Computational Intelligence*, 3:151–160, 1987.

[10] 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.

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

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

[13] 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.

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

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

[16] 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.

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

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

[19] J.R. Searle. Minds, Brain, and Programs. *The Behavioral and Brain Sciences*, 3:417–457, 1980.

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

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

[22] N. Zlatereva. Truth Mainteance Systems and Their Application for Verifying Expert System Knowledge Bases. *Artificial Intelligence Review*, 6, 1992.