Expert Systems Inference = Modeling Conflicts

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

Abstract— **Many expert systems tasks imply making assumptions. Where assumptions conflict, these assumptions must be managed in separate worlds. We describe an abductive device for managing such conflicts. This device lets us manage the conflicting assumptions, validate the (potentially) conflicting theory, as well as perform expert systems inference across theories that generate conflicts. We find that *conflict management* is a general framework for building test engines and inference engines in knowledge-based systems.**

I. Introductions

A computational device that can manage conflicts is both a *necessary* and *sufficient* device for processing expert systems.

Necessary: Easterbrook [7] argues that software usually contains unresolved conflicting contents since any software design contains:

- 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 software should explicitly model these conflicts since it is exactly this conflicts that will be required to understand opposing positions. Further, he argues 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 [7, p264].

Sufficient: The process of conflict resolution is, at a computational level, a general description of knowledge-level modeling. Clancey [1] argues that there are two basic problem-solving methods used by expert systems: *heuristic classification* and *heuristic construction* [2]. By heuristic classification, Clancey means that the inference engine merely *selects* a pre-existing inference path. In heuristic construction, the inference engine *constructs* its conclusions from partial inferences supplied in the knowledge base. Construction in conflicting situations is much harder than mere selection. Literals in different partial proofs may be conflict; i.e. while we can believe $A \vee B$, it may not be true that we can believe $A \wedge B$. The constructed pathways must be built with care in order to take into account these *cancelation interactions*. Multiple, conflicting assumptions may be possible and these must be managed separately. Note that heuristic classification is the "non-conflict" case and heuristic construction is the "conflict" case. The argument of this paper is that a single *abductive* framework which can handle the "conflict" situation can handle the "non-conflict" situation as a special case.

Easterbrook has made the case that conflict management is a necessary component for expert systems. Here, we focus on the *sufficient* argument. Section II gives examples of conflicts in knowledge bases. Section III gives an overview of abduction.

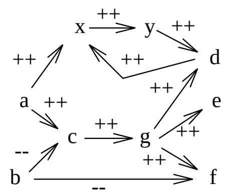

Fig. 1. $\mathcal{T}_1$ is a qualitative theory.

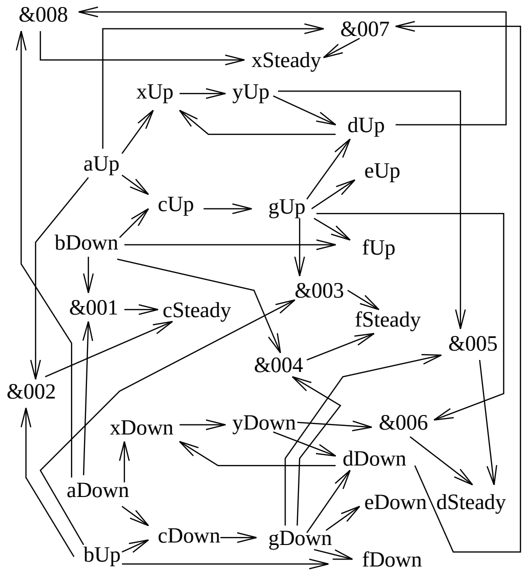

Fig. 2. $\mathcal{D}_1$ is the search space tacit in Figure 1

Section IV discusses the application of abductive devices to the general task of expert system execution and evaluation.

II. Examples of Conflicts in KBs

Consider the qualitative theory $\mathcal{T}_1$ of Figure 1. In that figure $\mathtt{x} \stackrel{++}{\rightarrow} \mathtt{y}$ denotes that `y` being `up` or `down` could be explained by `x` being `up` or `down` respectively while $\mathtt{x} \stackrel{--}{\rightarrow} \mathtt{y}$ denotes that `y` being `up` or `down` could be explained by `x` being `down` or `up` respectively. If we assume that (i) the conjunction of an `up` and a `down` can explain a `steady`; (ii) 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`). In the case of both A and B going UP, then we have two conflicting influences of 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.

```
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. 3. $\mathcal{T}_2$: Tuesday can be washing day or football day, but not both.

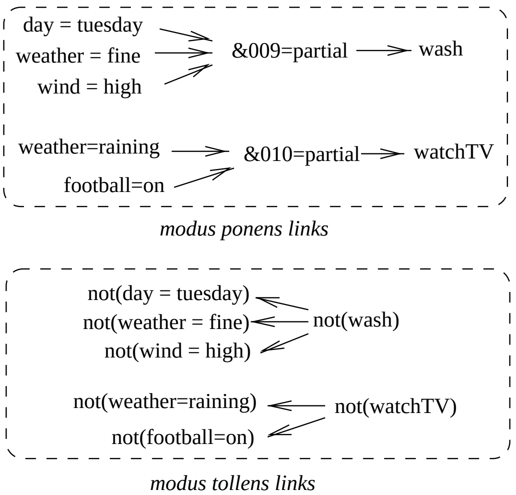

Fig. 4. $\mathcal{D}_2$ generated from $\mathcal{T}_2$.

Such indeterminacy from conflicting assumptions can be found in other types of qualitative theories. For example, the rules in Figure 3 are the theory $\mathcal{T}_2$. When executing this theory, $\mathcal{OUT}$ are the classes {`wash, watchTV`} which we define to be conflicting using the invariant violation report predicate $\mathcal{I}$. Figure 4 shows $\mathcal{D}_2$, the and-or graph associated with $\mathcal{T}_2$ (note the modus tollens links).

Similar and-or graphs can be generated from frame-based systems. For example, Figure 6 shows $\mathcal{D}_3$, the and-or graph tacit in the frame-based theory $\mathcal{T}_3$ in Figure 5. Note that given a superclass, 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 6 is such a specialisation link (for the sake of simplicity, we do not show the modus tollens links in Figure 6).

Note that both Figures 3 & 5 can generate conflicting conclusions:

- Figure 3: when `weather` is unknown
- Figure 5: when `motion` is unknown.

```
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. 5. $\mathcal{T}_3$: Things that fly and walk.

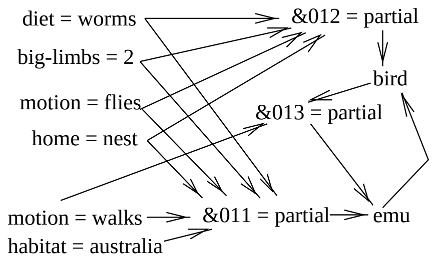

Fig. 6. $\mathcal{D}_3$ (modus ponens links only).

III. Abduction

In a single abductive framework, we can process the conflicts in all the above examples. 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 [8]. That is: $\mathcal{T} \cup \mathcal{A} \vdash \mathcal{OUT}$ and $\mathcal{T} \cup \mathcal{A} \not\vdash \perp$.

The proof trees $\mathcal{P}$ used to satisfy these two rules 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).

Returning to Figure 1, in the case where $\mathcal{OUT}$ = {`dUp,eUp,fDown`} and $\mathcal{IN}$={`aUp,bUp`}, then all the possible proofs are: $\mathcal{P}_1$= aUp $\rightarrow$ xUp $\rightarrow$ yUp $\rightarrow$ dUp; $\mathcal{P}_2$= aUp $\rightarrow$ cUp $\rightarrow$ gUp $\rightarrow$ dUp; $\mathcal{P}_3$= aUp $\rightarrow$ cUp $\rightarrow$ gUp $\rightarrow$ eUp; $\mathcal{P}_4$= bUp $\rightarrow$ cDown $\rightarrow$ gDown $\rightarrow$ fDown; $\mathcal{P}_5$= bUp $\rightarrow$ fDown.

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 since, for this model, the invariant violation report predicate $\mathcal{I}$ states that we can't believe in two different states for the same node:

```
i(X = State1, X = State2) :-
     \+ State1 = State2.
```

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 7. Note that each world is merely a subset of the edges shown in Figure 2.

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 $\mathcal{W}_2^{cover} = 2 = 67\%$. Note that if our task is expert system validation, then we would favour the world(s) that explain the most number of outputs. In this case, an abductive validation engine would favour $\mathcal{W}_1$ since it has a cover of 100%.

The examples of figures 3 & 5 would be handled in a similar manner. However, the way we assess rules may be different:

- Figure 3: if the known $\mathcal{FACTS}$ were {`day=tuesday, football=on`} and we had no information about the `weather` or the `wind`, then we could make a case that

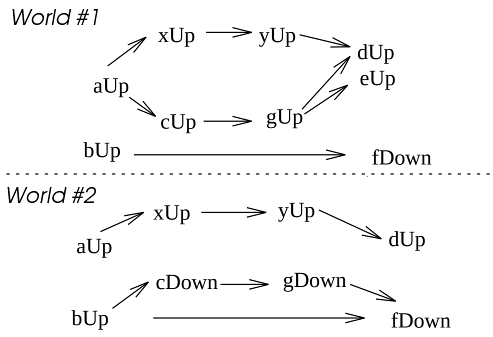

Fig. 7. Two worlds from Figure 1

watchTV was more likely than wash since 50% of the ancestors of watchTV are known compared with 33% of the ancestors for wash.

- Figure 5: We could favour the worlds that include the most-specific classes [13] (e.g. emu is better than bird).

IV. Conflict Modeling as a General Architecture for Expert Systems

A. Test Engines

Our HT4 abductive engine [11, 12] was originally built for validating qualitative models in neuroendocrinology. This system favored the worlds that explained the most outputs. 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.

B. Inference Engines

Many "constructive" expert system processes are abductive in nature [11]. For example:

- The connection between *diagnosis* and abduction is well-documented [3, 14, 15].
- *Explanation* can be characterised as the process of favoring the worlds which contain the most number of literals that the user has seen before.
- *Tutoring* is an extension to explanation. If the best explainable worlds were somehow sub-optimum, then 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 best explainable world but are in other, more optimum, worlds.
- *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. For planning, we could favor the world(s) with the least cost (the cost of a world is the maximum cost of the proofs in that world). Once generated, the best planning worlds could be passed to a *monitoring* system. As new information comes to light, we could reject the plans (worlds) which contradict the new information.

For a discussion of other abductive expert system tasks, see [11].

V. Connection to the ATMS

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* [4–6, 9]; i.e. all possible consistent worlds that are extractable from the theory. A more efficient case is that $\mathcal{IN}$ is smaller than all the roots of the graph and some *interesting subset* of the vertices have been identified as possible reportable outputs (i.e. $\mathcal{OUT} \subset \mathcal{V} - \mathcal{IN}$).

The ATMS 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).

Our base controversial assumptions and worlds are akin to ATMS labels and default logic extensions respectively [16]. However, we differ from ATMS/default logic in 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 $\mathcal{W}_2$.
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 inner-workings of a procedural reasoning system, we find that we can model it declaratively by abduction plus $\mathcal{BEST}$.

VI. Conclusion

We believe that an abductive architecture that generates multiple worlds which isolate conflicting assumptions and permits the customisation of the worlds assessment operator is a general architecture for Clancey-style expert systems. More specifically:

- HT4 handles heuristic classification as the case where $\mathcal{I}$ is empty; i.e. no invariant violations are possible. In this case, only one world is generated.
- When $\mathcal{I}$ is non-empty, and $\mathcal{A}_C$ is also non-empty, then each "construction" is a separate world.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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