Evaluating a Temporal Causal Ontology

Tim Menzies[1], Sam Waugh[2], Simon Goss[2], Robert F. Cohen[3]

[1]Artificial Intelligence Department,
School of Computer Science and Engineering,
University of NSW, Australia, 2052
[2]Defence Science and Technology Organisation,
Air Operations Division, Melbourne, Australia, 3001
[3]Department of Computer Science and Software Engineering,
University of Newcastle, NSW, Australia, 2308

timm@cse.unsw.edu.au\\sam.waugh@dsto.defence.gov.au\\simon.goss@dsto.defence.gov.au\\rfc@cs.newcastle.e

http://www.cse.unsw.edu.au/~timm

December 18, 1997

Abstract

We describe an experimental method to evaluate options within an ontology. (I) We commit to some precise description of the *task* of a theory written in an ontology. (II) A *mutator* automatically generates variants to known problems according to the different ontological options. (III) We try to perform the task over the mutations. (IV) We search for cases where the task fails. In this framework, ontological options can be assessed as follows. The better ontological options fail less on the task. This framework is instantiated in the context of a causal ontology. Four variants are proposed for handling time in the given ontology. Two of those four are found to be clearly superior and two were found to be clearly inferior.

Submitted to the International Conference on Formal Ontology in Information Systems (FOIS'98) (in conjunction with the 6th International Conference on Principles of Knowledge Representation and Reasoning: KR'98) Trento, Italy, June 6-8, 1998.

1 Introduction

Are all ontologies good ontologies? Given a particular domain and a particular community of analysts, a range of ontologies can be proposed. How are we to assess the utility of these different ontologies? In this paper, we take the view that ontologies are useful for theory construction and theories are written for some task. We will show that seemingly trivial variants in a temporal causal ontology block the performance of a task we call KB-maintainability. Roughly speaking, KB-maintainability means checking if we can tell if some change is an improvement to a theory.

If ontologies are assessed with respect to a specific task (e.g. KB-maintainability), then that task will skew the choice of ontologies. Consequently, we find that we cannot propose a single *best* temporal causal ontology. Rather, we think that knowledge engineers should develop libraries of ontological variants. When exploring a new domain, representative problems from that domain should be explored to find which variants are useful. However, all the ontological options discussed here are really small variants on a small part of an ontology. If this is generally the case, then we can look forward to the day in which ontological engineering is a partially automatic process in which assessment agents (such as the one proposed here) explore variants in sub-classes of a meta-ontology. When these agents terminate, they report back which ontological variants are relevant to the current problem. An analogous process of recursively exploring options in problem solving methods has been extensively explored (e.g. [Benjamins & Jansweijer, 1994, O'Hara & Shadbolt, 1997]).

Experimental ontological assessment is an infant science. To our knowledge, our work is the first proposal to assess ontologies using empirical methods; i.e. hundreds of thousands of trials, controlled variants between each trial, numeric data collected during each trial. A drawback with the current work is that, given the state-of-the-art, we can only report results on a very limited ontology. For example, the ontology studied here is only a flat causal structure without terms in a subsumption hierarchy. However, our belief (as yet untested) is that the methods used here (mutators, analysis of KB-maintainability) are quite general and could be used to find restrictions to other ontologies. Menzies has argued [Menzies, 1997] that the KE field needs such general evaluation methodologies.

This paper is structured as follows. Our ontology was developed in response to certain drawbacks with the 1980s research into qualitative reasoning. That work is reviewed first, followed by our ontology. Next, we explore how we would add time into that ontology. Several options are identified for adding time. An experiment is then defined to assess these options. When the experiment was performed, it was found that not all these options were useful. Our discussion section generalises this work into a framework for assessing ontologies.

2 Causality, Qualitative Reasoning, and QCM

Qualitative compartmental modeling (QCM) [Menzies & Compton, 1997, Waugh *et al.*, 1997] offers a physical causal ontology for testing hypotheses in neuroendocrinology (the study of nerves and glands). QCM is a generalisation of QMOD: Feldman and Compton's work [Feldman *et al.*, 1989a, Feldman *et al.*, 1989b] on qualitative reasoning (QR). This section reviews the research into QR and causality. In the rest of this article, we use the term QCM to refer to both QMOD and its successor QCM.

Systems developed in the late 1970s let the domain expert specify explicit networks of causal connections. Early work demonstrated the utility of this approach; e.g. CASNET [Weiss *et al.*, 1978] and MECHANISMS LAB [Rieger & Grinberg, 1977]. Subsequent work argued for adding abstraction hierarchies to causal models; e.g. ABEL [Patil *et al.*, 1981]. Causal models at various levels of abstraction permit inferencing down/up/across abstraction level(s) if more/less/same abstraction is useful in the reasoning. These early systems were

significant pieces of original and innovative research and successes in their own right. However, they failed to provide general principles for later work. In a quest for more general principles, the qualitative reasoning (QR) community focused on the processing of systems called qualitative differential equations (QDE) which are:

- Piece-wise well-approximated by low-order linear equations or by first-order non-linear differential equations;
- Whose numeric values are replaced by one of three qualitative states: up, down, or steady [Iwasaki, 1989].

(Note that not all QR is based directly on QDEs. For example, Yip discusses the dynamics of hamiltonians [Yip, 1991]. Bratko *et. al.* discuss the KARDIO system. KARDIO generated a rule-base for heart disease via a machine learning program that condensed the output from an indeterminate qualitative model of heart disease [Bratko *et al.*, 1989]. Yip and Bratko do not discuss causality directly so we will leave them out of the subsequent discussion. In the bond graph approach, models are built out of components representing abstract energy sources, sinks, storage, and dissipater devices [Top & Akkermans, 1991]. We still group bond graphs with the rest of equation- based QR since bond graph models serve as a front-end to equation specification.)

A QDE is still a mathematical equation and mathematics is a poor model for causality. Ohms's Law ($R=V/I$) relates resistance R to current I and voltage V. Note that changes in voltage and current do not cause changes in resistance, even though the mathematical formulae suggests this is possible. Resistors cannot be manufactured to a certain specification merely by attaching wire to some rig and altering the voltage and current over the rig. Ignoring the effects of temperature and high-voltage breakdown, resistance is an invariant built into the physics of a wire. Hidden within Ohm's Law are rules regarding the direction of causality between voltage, current, and resistance. Such rules are invisible to a mathematical formulation.

Qualitative reasoning, explanation, and causality are intimately connected. Causality was a central concern in QR till the mid-1980s [Coiera, 1992]:

> *... It is clear that causality plays an essential role in our understanding of the world ... to understand a situation means to have a causal explanation of the situation.* [Iwasaki, 1988].

Initially two qualitative ontologies were proposed: DeKleer and Brown's 1984 CONFLUENCES system [DeKleer & Brown, 1984] and Forbus's 1984 qualitative process theory (QPT) [Forbus, 1984]. Later work in 1986 recognised that both these systems processed QDEs and a special theorem prover, QSIM, was written by Kuipers especially for QDEs [Kuipers, 1986]. Compilers were written to covert QPT models into QSIM [Crawford *et al.*, 1992].

After an inclusive public debate in 1986 between public debate between the CONFLUENCES approach and a rival theory [Iwasaki & Simon, 1986], the term *causality* was avoided by many QR researchers. Forbus's 1992 retrospective on causality and the 1980s QR research is primarily negative:

> *... In terms of violating human intuitions, each system of qualitative physics fails in some way to handle causality properly. Like (QPT)*

> *theory, deKleer and Brown's CONFLUENCES theory... fails to distinguish between equations representing causal versus non-causal laws. Kuipers QSIM contains no account of causality at all.* [Forbus, 1992].

In summary, the 1980s experiment with using QDEs to model causal explanations failed. Hence, Feldman and Compton [Feldman *et al.*, 1989b], followed by Menzies and Compton [Menzies, 1995, Menzies & Compton, 1997], explored causal modelling via direct causal mappings. Unlike the QR research of the 1980s, QCM does not infer causality from a non-causal source (e.g. equations). Rather, like the earlier work of CASNET, MECHANISMS LAB, and ABEL, experts could enter their causal links directly. However, due to the vagueness of causality, QCM does not treat its causal models as gospel truths. Rather, the causal models are treated as very approximate, possible very incorrect, models. All models are assessed via an *abductive validation engine* discussed below.

QCM adopted a physical ontology taken from compartmental modelling [McIntosh & McIntosh, 1980]. Informally, the QCM metaphor is that of liquids sloshing around between containers. More precisely, a QCM knowledge base contains pipes connecting *tubs* (we will specialise *tub* below to be a compartment or a flow). The state of a tub is the sign of the first derivative of the value in that tub (i.e. *up, down* or *steady*):

```
change be something
       has value    are in (up,down,steady).

tub  be  something
     has inflow     are of pipe
                    are many
     has outflow    are of pipe
                    are many
     has state      are of change.

pipe be  something
     has from       are of tub
     has to         are of tub
     has influence  are of effect
     has controller are of abler
                    are maybe.
```

(This article uses Menzies' *Pirate* Prolog-based notation for UML-type specifications [Booch *et al.*, 1997]. *X be Y* denotes that *Y* is a subclass of *X*. *Something* is the default initial superclass. *X has Y* denotes a slot *Y* inside class *X*. The slot *Y* inside *X* is denoted *X.Y*. *Are* denotes a slot constraint. Several kinds of constraints are defined: multiplicity, range, and types. *X are of Y* is a type constraint saying that slot *X* is filled with instances of type *Y*. The default type constraint is *X are of something*. *X are 1 to many* is a multiplicity constraint. There is a shorthand defined for common multiplicity constraints: *many* means *0 to many* and *maybe* means *0 to 1*. If no multiplicity constraint is offered, the default is *1 to 1*. *X are in List* is a range constraint. Default values are set using =. Subclasses can install new defaults into slots defined in superclasses using *with*. Comments in the above figure offer explanations of *Pirate* constructs.)

The flow through pipes can be enabled or disabled by controller events. If that event is known to be *on* (which QCM interprets as meaning *state.change=up*), then the enabler will permit flow through the pipe. On the other hand, disablers block the flow if the event is *on*.

```
abler be something
      has owner are of event.
disabler be abler.
enabler  be abler.
```

There are two special kinds of tubs: compartments and flows. Flows can increase or decrease the rate of transfer of stuff through a pipe. Two special kinds of compartments are experimental interventions (called events) and measured outcomes (called measures). Compartments can change the flow rates; i.e. flows rates are controlled by compartments.

```
compartment be tub.
measure     be compartment.
event       be compartment.
flow        be tub
            has controller are maybe
                           are of compartment.
```

There are several types of influences according to how the state at one end of the pipe influences the state at the other end of the pipe: *++*, *–*, *+–*, *+-+*:

```
xplain has fromType  are of tub
       has cause     are of change
       has toType    are of tub
       has effect    are of change.

effect be  something
       has symbol
       has synonym      are maybe
       has transitions are of xplain
                        are of 1 to 2.
direct    be   effect
          with symbol       = '++'    with synonym = 'encourages'
          with transitions = (xplain and cause=up   and effect=up,
                              xplain and cause=down and effect=down).
inverse   be   effect
          with symbol       = '--'    with synonym = 'discourages'
          with transitions = (xplain and cause=up   and effect=down,
                              xplain and cause=down and effect=up).
creator   be   effect
          with symbol        = '+-+'
          with transitions = (xplain with cause=up   and effect=up).

destroyer be   effect
          with symbol        = '+--'
          with transitions = (xplain with cause=up   and effect=down).
```

X++Y is called a direct or encourages influence and means that *Y.state* being *up* or *down* could be explained by *X.state* being *up* or *down* respectively. Direct influences could be used to partially replicate mathematical proportionality. *X–Y* is called an inverse or discourages influence and means that *Y.state* being *up* or *down* could be explained by *X.state* being *down* or *up* respectively. Inverse influences could be used to partially replicate mathematical inverse proportionality. Note that *++* and – are not the same as QSIM's *M+* and *M-* constraint since QCM only applies these influences if they satisfy an abductive inference plausibility operator (discussed below). *X+-+Y* is called a creator influence and means that *Y.state* being *up* could be explained by *X.state* being *up*. Creators can be used to model in-flows into the top of a tank. Such

in-flows can only ever increase the amount of stuff in that tank. If the level of that in-flow decreases, then the amount of stuff in the tank does not decrease. *X+−Y* is called a destroyer influence and means that *Y.state* being *down* could be explained by *X.state* being *up*. Destroyers can be used to model out-flows from the bottom of a tank. Such out-flows can only ever decrease the amount of stuff in that tank. If the level of that out-flow decreases, then the amount of stuff in the tank does not increase.

To use QCM, the user first enters statements which are compiled down into instances of the above classes. For example, consider the statements *throwing the power switch turns on the lights, but only if the rats are not shorting out the wires in the basement.* This is modeled as the disabler statement *if rats then not power ++ lights* where *power* and *lights* are measures connected by a direct influence. The connection *power ++ lights* is disabled if the *rats* event is on. Note the indeterminacy in the definitions of the effects. Consider the case of someone losing weight (*weight=down*) and taking less exercise (*exercise=down*) in the system *weight ++ heartDisease* and *exercise – heartDisease*. It is unclear if *heartDisease* will do up or down. Such *chatter* is a fundamental part of qualitative reasoning [Kuipers, 1986]. We handle chatter using an abductive inference engine. Abduction is the search for consistent explanations [Eshghi, 1993]. Explanations can make assumptions and mutually exclusive assumptions must be kept in separate *worlds* [Menzies, 1996b]. Unlike deduction, abduction is a plausible, not certain inference. The utility of making a particular abductive inference must be assessed via some plausibility operator *pl* [Bylander *et al.*, 1991]. A range of different knowledge-level tasks can be modeled via an appropriate choice of plausibility operators [Menzies, 1996a, Menzies & Mahidadia, 1997]. For example: least-cost planning is a synonym for abduction with a plausibility operator that favours explanations that maximises goal coverage whilst minimising inference cost. Minimal-fault diagnosis is a synonym for abduction [Console & Torasso, 1991] with a plausibility operator that maximises goal coverage whilst minimising the number of input causes. Validation is a synonym for abduction with a plausibility operator that maximises goal coverage [Menzies & Compton, 1997, Menzies, 1996b]. Other synonyms are discussed in [Menzies, 1996a].

Next, a problem is specified. A problem is a pair of input and output tub states (e.g. *output = (light.state=up), input=(rats=down)*). Inputs are always events and outputs are always measures. Abduction is then performed looking for pathways from outputs back to inputs. Pathways must satisfy the following criteria: (1) only terminates on members of the input set. (2) starts on members of the output set; (3) must not contain loops; and must (4) not contain two states from the same tub (i.e. it is illegal to believe both *heartDiesase=up* and *heartDisease=down* in the same proof). If an output vertex can be so connected to an input vertex, it is called an *explicable* output. Once the pathways are generated, they are divided into maximal consistent subsets which become our abductive worlds. For more details, see [Menzies & Compton, 1997].

Pathway criteria number 4 makes this an NP-hard problem. Gabow et.al. [Gabow *et al.*, 1976] showed that finding a directed path across a directed graph that has at most one of a set of forbidden pairs is NP-hard. Such forbidden pairs are present in the above definition of abduction; i.e. two mutually exclusive states of some tub. When implemented, QCM has been observed to exhibit exponential runtimes [Menzies, 1995], as one would expect for an NP-hard

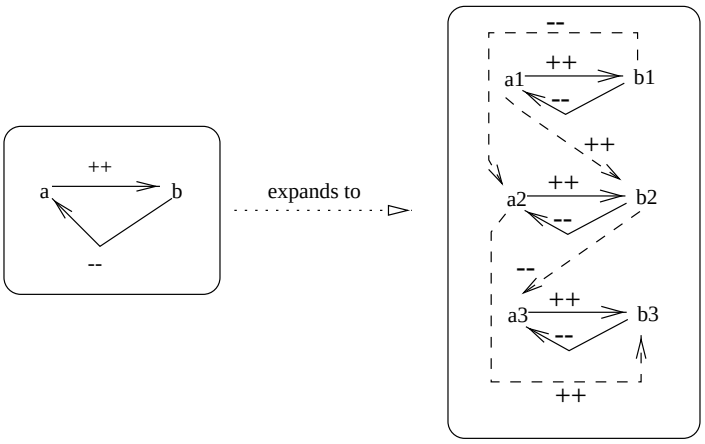

Figure 1: A theory (left) renamed over 3 time intervals and connected using IEDGE (time edges shown as dashed edges).

problem. Despite being NP-hard, it has been shown that QCM is practical tool for validating many real-world theories such as certain fielded expert systems and theories from neuroendocrinology [Menzies, 1996a]. Recall from the above discussion that abductive validation is the search for worlds that explain the maximum number of outputs. If after making all assumptions possible a theory can still not generate worlds that cover the known output, then that theory is clearly faulty. Feldman and Compton [Feldman *et al.*, 1989a], followed by Menzies [Menzies & Compton, 1997], have shown that abductive validation engines could detect large numbers of previously unseen errors in neuroendocrinology theories published in international refereed journals. Surprisingly, these faults were found using the data published to support those theories

3 Adding Time to QCM

This section discusses temporal extensions to the QCM ontology.

Recall the above pathway criteria number 4: we cannot believe in two states of the same tub. This criteria should not hold in the case of feedback loops. Suppose we executed the system *sunshine ++ hot* and *hot ++ evaporation* and *evaporation ++ rain* and *rain – sunshine* for several time ticks. We would see the state of *sunshine* rise and fall. That is, we would be believing in several values of *sunshine*, but at different times.

We can extend QCM to handle such time-based feedback loops as follows. We create one copy of the search space for each time interval in the simulation. For example, consider the theory A *++* B but B – A. If we executed A*++*B and B–A over three time ticks, we could search the space shown in Figure 1.

Here we are connecting objects at different time intervals by an *implicit time edge linking* or IEDGE policy: when i.e. X*++*Y implies that X at time T=i also connects to Y at T=i+1 (see the dashed lines). Pathway criteria number 4 is now extended to: pathways must not contain loops and must not contain two states from the same tub *at the same time.*

Alternatives to IEDGE are discussed later. Here, we pause to comment that IEDGE has some interesting properties for long simulation runs. We have shown elsewhere [Menzies *et al.*, 1997] that for ...

- Tubs which take S states without steadies;
- Pipes which are not creators or destroyers;
- Pipes which have no ablers;
- Simulations that are connected via IEDGE

... then if we cannot generate a proof in S copies, then we will never be able to generate a proof at all. Roughly speaking, if every edge offers a comment on all the states of its downstream vertices (e.g. direct and inverse), then the state space rapidly *saturates* and we can reduce the granularity of the time axis to just under twice the granularity of the measurements of that theory. We can use this proof as an optimisation technique as follows. Suppose, for example, that we ran a simulation over the tubs for a million time steps and we only collected measurements at three time points (say, 1 and 500,000 and 1,000,000). If we had not proved saturation for our device, then the search for these proofs would have to explore 1,000,000 renamings. Given the exponential runtimes of abduction, this is clearly undesirable. However, since we have proved saturation, then we know that if a proof cannot be found in two renamings, then no such proof exists. Consider a proof from (e.g.) time 1 to time 500,000. If that proof cannot terminate in the space 1 and 500,000, then it will never terminate. That is, when building this proof, we could ignore the search space that used the renamings from 2 to 499,999. A similar argument could be made for proofs from 500,000 to 1,000,000. In practice, we would only need to search the space created for the three measured time points. That is, in this example, this optimisation lets us reduce the search space to three-millionths of the unoptimised version.

4 Temporal Extensions to the QCM Ontology

How can time be represented in QCM? We saw above the use of IEDGE: when X connects to Y, then we say that X at time $T{=}i$ also connects to Y at $T{=}i{+}1$. IEDGE is only one of a family of temporal QCM ontological variants. Firstly, we can cross time over edges or over nodes. Secondly, we can cross time on all structures in the theory, or only on those explicitly mentioned by the user. Each combination of these alternatives defines a temporal causal ontology illustrated in Figure 2. In this figure, in the *explicit node linking* language (or XNODE), we only cross time on the nodes explicitly denoted as time nodes by the user (in this example, A). In the *implicit node linking* language (or INODE), we cross time on all nodes. In the *explicit edge linking* language (or XEDGE), we only cross time on the edges explicitly denoted as time edges by the user (in this example, A to B). Lastly, in IEDGE, we cross time on all edges.

Note that there are many more possible interpretations of time. However, in this article, we restrict ourselves to four.

All these variants imply a small variation to the above ontology. All pipes and tubs have to know their *timeLinkStatus*. In XNODE, only some tubs have

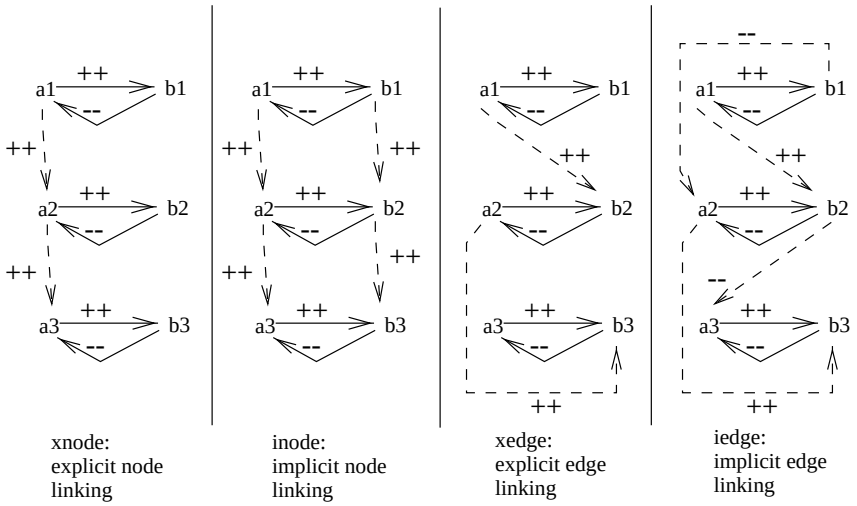

Figure 2: Figure 1 (left) renamed over 3 time intervals using different time linking methods. Dashed lines indicate time edges.

timeLinkStatus=true while in INODE, all tubs have *timeLinkStatus=true*. Similarly, in XEDGE, only some pipes have *timeLinkStatus=true* while in INODE, all pipes have *timeLinkStatus=true*. Further, the nature of the time link must be specified. A link can be within the same time tick (e.g. *A1++B1*) which takes zero time to traverse (the *mythical* time of the CONFLUENCES system [DeKleer & Brown, 1984]) or across time (e.g. *A1+=A2*) which will take 1 time tick to traverse. As a default, we assume zero time for the traversal.

```
qcmElement be something
     has timeLinkStatus=false
         are of boolean
     has timeLink are of delay.
tub  be qcmElement.
pipe be qcmElement.

delay be something
      has duration=0 are in (0,1).
```

4.1 KB-maintainability

Are all the variants (XNODE, XEDGE, INODE, IEDGE) valid? This section defines a necessary, but not sufficient pre-condition for KB-maintainability which we will use to assess these variants.

How do we know if we can maintain a theory? One requirement would be the ability to check if some variation of a theory is better than the original theory. We divide this requirement into two parts. Firstly, we need a quick first-pass test that can clearly identify obviously poor theories. Secondly, we need a more detailed view that can assess minor variants on a theory. Given a range of theories which degrade from *good* to *poor*, we like to see the *maintenance success* curve of Figure 3.

A good theory is *permissive*; i.e. it can explain known behaviour (the success curve rises high on the left-had-side). However, we also need to test that the

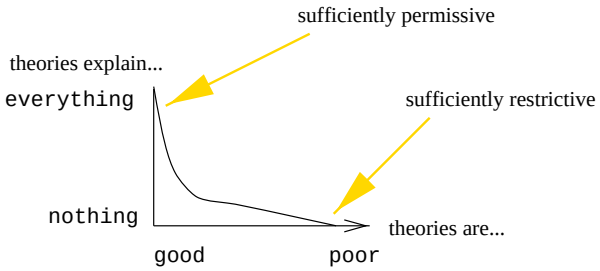

Figure 3: Visualising a maintainable representation language. The KB-testability curve of a good representation should be both *permissive* to all good theories (explains correct behaviour) and *restrictive* to all bad theories (prevent poor theories explaining any behaviour).

permissiveness is not due to a poorly-constrained theory. A theory that condones any conclusion is a poor theory. Hence, a theory must also be *restrictive* (the success curve descents to a low value on the right-hand-side). Further, we want the second derivative of this graph to be non-negative; e.g. the ski-slope shape of success curve. With such a curve, we can quickly get feedback if some change improves or degrades a theory.

We say that a representation is KB-maintainable if its this test curve for good to bad theories looks like the ski-slope of the success curve. We show below that only some ontological options for handling time in QCM satisfy this definition of KB-maintainability. Hence, in QCM, the ontologies should be restricted to these representations of time.

Note that our current definition of KB-maintainability is a necessary, but not sufficient pre-condition to true maintainability. We only claim that if a system does not satisfy our KB-maintainability test, then it will not support true maintainability (and leave the converse statement to future work).

4.2 Testing KB-maintainability

To test KB-maintainability for these four temporal languages, we performed the following experiment. First, we implemented a quantitative *fisheries simulation model* using the equations from [Bossel, 1994] (pages 135-141). Secondly, we built a qualitative form of the fisheries model as shown in Figure 4.

Note that this fisheries model is silent on the issue of how to handle time. We must add in a temporal causal interpretation is order to handle the feedback loops. In the following experiment, we added in XNODE or XEDGE or INODE or IEDGE and checked if any of them supported the maintenance success curve. The quantitative model was used to generate numeric test data which was stored in the *measure* array. From each comparison of *measure(i)* with *measure(j)*, entries were written to an array of qualitative observations called *changes*. *Changes* was used to generate the input/output needed for for abductive validation. For example, if in comparison *change(37)*, the fish density *fdens* was increased and the fish catch *fcatch* was always seen to decrease at all time steps, then *change(37).in* is *(fdensUp)* and *change(37).out* is *(fcatch(t=1)Down, fcatch(t=2)Down, fcatch(t=3)Down, fcatch(t=4)Down, fcatch(t=5)Down)*

Next, we need to generate a range of theories. To do this, we built a *mutator*

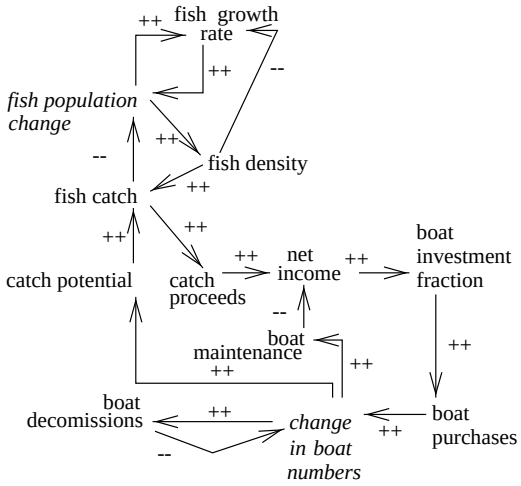

Figure 4: The fisheries model. Adapted from [Bossel, 1994] (pp135-141). Variables in italics were used in the XNODE study. Edges downstream of an XNODE were used as the explicit time edges in the XEDGE language.

that produced successively worse versions of the fisheries model. The mutator worked as follows: the sign on a random sample of the X statements in the qualitative theory were *corrupted*: i.e. flipped (*++* to *–* or visa versa). Given a model with E edges, then as we vary X from *0 to E*, we are moving from a *good* model to a *poor* model; i.e. the x-axis of our maintenance success curve. This corrupted model was then copied to create a set of time *copies*. These time copies were connected using one of our temporal languages: IEDGE, INODE, XEDGE, XNODE. *Change* inputs were mapped into *copy(0)*. *Change* outputs were mapped into some *copy(i)* (i greater than zero). The success of each run was assessed by recording the percent of the *explicable outputs* i.e. those outputs that the model could connect back to inputs. The details of this experiment are shown in Figure 5.

4.3 Results

Figure 6 graphs the average values of *Xplicable* variable from the experimental design. INODE was not sufficiently restrictive. Even with all edges corrupted, INODE allowed theories to explain correct behaviour. Also, XEDGE was not sufficient permissive. Even on correct theories, XEDGE could only explain half the known behaviour. XNODE was the closest to the maintenance success curve, followed by IEDGE. However, IEDGE was not as permissive as XNODE. Hence, except in the case where we need to optimise long simulation runs, we recommend XNODE. Otherwise, we recommend IEDGE.

Note that for XNODE and IEDGE, after only a third of the theory being mutated, only half the outputs were inexplicable. This is a nice result: with these modelling languages, we can get a clear and early indication if we are stray from a good theory.

We conclude that, in terms of KB-maintainability, the four variants on the

```
T  := MaxTime;
M0 := fisheries;                    measure := run_quantitative_model(T,M0)
M1 := qualitativeVersionOf(M0);     change  := comparisons(measure)
for policy ∈ (IEDGE, INODE, XNODE, XEDGE)  begin
   for corrupted:=0 to |E| begin
      for r:=1 to 20 begin
         M2 :=corruptSomeEdgesChoosenAtRandom(corrupted,M1)
         for t:=0 to T   copy(t):= M2
         for t:=0 to T-1 time_connect(copy(t),copy(t+1),policy)
         for i:=1 to |change| begin
            (In,Out(1..T)):= change(i)
            Xplained(1..T):= run_qualitative_model(copy,In,Out)
            Xplicable(policy,r,corrupted,i):=|Xplained(T)|*100/|Out(T)|
end end end end
```

Figure 5: Experimental design for assessing linking options

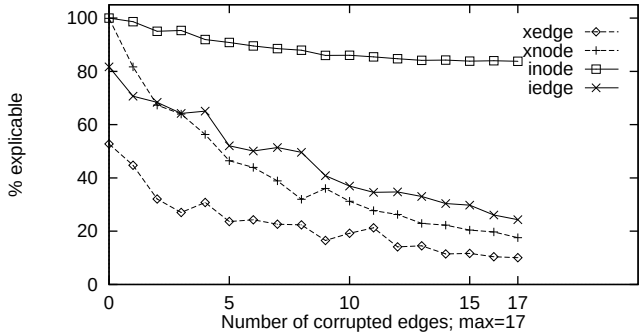

Figure 6: Graph of `Xplicable` from Figure 5.

QCM conceptual modelling language are ranked as follows: XNODE (best for short to medium length simulation runs), then IEDGE (best for long simulation runs where XNODE fails due to the size of the search space), then XEDGE (poor: no permissive enough) and INODE (poor: not restrictive enough).

5 Related Work

We have argued above that seemingly minor variants in an ontology can have major effects on the utility of that ontology for generating theories which can satisfy some task. This is an analogous results to other researchers:

- The satisfiability community reports that there exist very narrow regions in which the behaviour of a knowledge base can change markedly, and undesirably (e.g. [Smith, 1996, Cheeseman *et al.*, 1991]).
- Tambe, Newell, and Rosenbloom [Tambe *et al.*, 1990, Tambe & Rosenbloom, 1994] discuss *multiple attributes* in production rule conditions. Given an attribute test of an object, if one query to one attribute of that object can result in multiple matches, then the time bound on matching a single rule can be exponential.
- Brachman and Levesque [Brachman & Levesque, 1984] discuss a seemingly minor variation to a frame-based language FL. FL contains the RESTR construct which places restrictions on valid slot values. At first glance, RESTR seems to reduce the search space associated with processing that frame. However, Brachman and Levesque prove that the exact opposite is true. A language without RESTR (FL^-) can test if some frame subsumes another frame in, at most, polynomial time. However, the same test in FL has an upper bound of exponential time; i.e. it may be too slow to be practical.

6 Discussion

We have some good news and bad news about ontologies. First, the good news. We can see general principles for building agents that automatically support ontological engineering:

- We commit to some precise description of the the *task* of a theory written in an ontology (in this case, KB-maintainability).
- A *mutator* automatically generates variants to known problems according to the different ontological options.
- We try to perform the task over the mutations.
- We search for cases were the task fails.
- We caution against the ontological options which generate the failure.

Secondly, the bad news. We speculate that ontologies cannot be reused verbatim. Rather, once a knowledge engineer selects an ontology near to her target domain, she must then take the time to *commission* the ontology. That is, given

the tasks of the theories in her domain, she must explore aspects of the ontology looking for constructs which block those tasks. Once those constructs are found, they should not be used in knowledge acquisition. For example, in this work, we would offer a different set of temporal causal constructs for building theories that are used in short to medium length simulation runs (XNODE) to long simulation runs (IEDGE and no *state=steady* and no creators or destroyers influences and no ablers).

Lastly, we note that the bad news is not so bad. The effort of comissioning the ontology need not be prohibitive if the knowledge engineer has an ontological engineering workbench available that contains semi-automatic ontology assessment agents. This research is the first step towards building such a workbench.

References

[Benjamins & Jansweijer, 1994] Benjamins, V. & Jansweijer, W. (1994). Toward a Competence Theory of Diagnosis. *IEEE Expert*, 9(5):43–53.

[Booch *et al.*, 1997] Booch, G., Jacobsen, I., & Rumbaugh, J. (1997). *Version 1.0 of the Unified Modeling Language*. Rational. http://www.rational.com/ot/uml/1.0/index.html.

[Bossel, 1994] Bossel, H. (1994). *Modeling and Simulations*. A.K. Peters Ltd. ISBN 1-56881-033-4.

[Brachman & Levesque, 1984] Brachman, R. & Levesque, H. (1984). The Tractability of Subsumption in Frame-Based Description Languages. In *AAAI '84*, pages 34–37.

[Bratko *et al.*, 1989] Bratko, I., Mozetic, I., & Lavrac, N. (1989). *KARDIO: a Study in Deep and Qualitative Knowledge for Expert Systems*. MIT Press.

[Bylander *et al.*, 1991] Bylander, T., Allemang, D., M.C. Tanner, M., & Josephson, J. (1991). The Computational Complexity of Abduction. *Artificial Intelligence*, 49:25–60.

[Cheeseman *et al.*, 1991] Cheeseman, P., Kanefsky, B., & Taylor, W. (1991). Where the Really Hard Problems Are. In *Proceedings of IJCAI-91*, pages 331–337.

[Coiera, 1992] Coiera, E. (1992). The Qualitative Representation of Physical Systems. *The Knowledge Engineering Review*, 7:1–23.

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

[Crawford *et al.*, 1992] Crawford, J., Farquhar, A., & Kuipers, B. (1992). QPC: A Compiler from Physical Models into Qualitative Differential Equations. In Faltings, B. & Struss, P., (Eds.), *Recent Advances in Qualitative Physics*. The MIT Press.

[DeKleer & Brown, 1984] DeKleer, J. & Brown, J. (1984). A Qualitative Physics Based on Confluences. *Artificial Intelligence*, 25:7–83.

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

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

[Feldman *et al.*, 1989b] Feldman, B., Compton, P., & Smythe, G. (1989b). 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.

[Forbus, 1984] Forbus, K. (1984). Qualitative Process Theory. *Artificial Intelligence*, 24:85–168.

[Forbus, 1992] Forbus, K. (1992). Pushing the Edge of the (QP) Envelope. In Faltings, B. & Struss, P., (Eds.), *Recent Advances in Qualitative Physics*, pages 245–261. The MIT Press.

[Gabow *et al.*, 1976] Gabow, H., Maheshwari, S., & Osterweil, L. (1976). On Two Problems in the Generation of Program Test Paths. *IEEE Trans. Software Engrg*, SE-2:227–231.

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

[Iwasaki, 1989] Iwasaki, Y. (1989). Qualitative Physics. In A. Barr, P. C. & Feigenbaum, E., (Eds.), *The Handbook of Artificial Intelligence*, volume 4, pages 323–413. Addison Wesley.

[Iwasaki & Simon, 1986] Iwasaki, Y. & Simon, H. (1986). Causality in Device Behaviour. *Artificial Intelligence*, 29:3–31.

[Kuipers, 1986] Kuipers, B. (1986). Qualitative Simulation. *Artificial Intelligence*, 29:229–338.

[McIntosh & McIntosh, 1980] McIntosh, J. & McIntosh, R. (1980). *Mathematical Modeling and Computers in Endocrinology*. Springer-Verlag.

[Menzies, 1995] Menzies, T. (1995). *Principles for Generalised Testing of Knowledge Bases*. PhD thesis, University of New South Wales. Avaliable from `http://www.cse.unsw.edu.au/~timm/pub/docs/95thesis.ps.gz`.

[Menzies, 1996b] Menzies, T. (1996b). On the Practicality of Abductive Validation. In *ECAI '96*.

[Menzies, 1997] Menzies, T. (1997). Evaluation Issues for Problem Solving Methods. Submitted to Banff KA workshop, 1998. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/97eval`.

[Menzies, 1996a] Menzies, T. (September, 1996a). Applications of Abduction: Knowledge Level Modeling. *International Journal of Human Computer Studies*, 45:305–355.

[Menzies *et al.*, 1997] Menzies, T., Cohen, R., S, W., & Goss, S. (1997). Applications of Abduction: Testing Very Long Qualitative Simulations. In preperation.

[Menzies & Compton, 1997] Menzies, T. & Compton, P. (1997). Applications of Abduction: Hypothesis Testing of Neuroendocrinological Qualitative Compartmental Models. *Artificial Intelligence in Medicine*, 10:145–175.

[Menzies & Mahidadia, 1997] Menzies, T. & Mahidadia, A. (1997). Ripple-Down Rationality: A Framework for Maintaining PSMs. In *Workshop on Problem-Solving Methods for Knowledge-based Systems, IJCAI '97, August 23.*

[O'Hara & Shadbolt, 1997] O'Hara, K. & Shadbolt, N. (1997). Inerpreting Generic Structures: Expert Systems, Expertise, and Context. In Feltovich, P., Ford, K., & Hoffman, R., (Eds.), *Expertise in Context*, chapter 19, pages 449–472. MIT PRess.

[Patil *et al.*, 1981] Patil, R., Szolovitis, P., & Schwartz, W. (1981). Causal Understanding of Patient Illness in Medical Diagnosis. In *IJCAI '81*, pages 893–899.

[Rieger & Grinberg, 1977] Rieger, C. & Grinberg, M. (1977). The Declarative Representation and Procedural Simulation of Causality in Physical Mechanisms. In *IJCAI '77*, pages 250–256.

[Smith, 1996] Smith, B. (1996). Locating the Phase Transition in Binary Constraint Satisfaction Problems. *Artificial Intelligence.*

[Tambe *et al.*, 1990] Tambe, M., Newell, A., & Rosenbloom, P. (1990). The Problem of Expensive Chunks and its Solution by Restricting Expressiveness. *Machine Learning*, 5(3):299–348.

[Tambe & Rosenbloom, 1994] Tambe, M. & Rosenbloom, P. (1994). Investigating Production System Representations for Non-combinatorial Match. *Artificial Intelligence*, 68(1).

[Top & Akkermans, 1991] Top, J. & Akkermans, J. (1991). Computational and Physical Causality. In *IJCAI '91.*

[Waugh *et al.*, 1997] Waugh, S., Menzies, T., & Goss, S. (1997). Evaluating a Qualitative Reasoner. In Sattar, A., (Ed.), *Advanced Topics in Artificial Intelligence: 10th Australian Joint Conference on AI.* Springer-Verlag.

[Weiss *et al.*, 1978] Weiss, S., Kulikowski, C., & Amarel, S. (1978). A Model-Based Method for Computer-Aided Medical Decision-Making. *Artificial Intelligence*, 11.

[Yip, 1991] Yip, K. (1991). Understanding Complex Dynamics by Visual and Symbolic Reasoning. *Artificial Intelligence*, 51:179–221.