

Applications of Abduction: Testing Very Long Qualitative Simulations

Tim Menzies, Robert F. Cohen, Sam Waugh, Simon Goss

T. Menzies is with the Department of Electrical and Computer Engineering; 2356 Main Mall; Vancouver, B.C. Canada V6T1Z4 Email: timm@menzies.com; Url: www.tim.menzies.com.

Robert F. Cohen is with Algomagic Technologies, Inc. Email: rfc@algomagic.com; Url: <http://www.algomagic.com>

Sam Waugh and Simon Goss are with the Defence Science and Technology Organisation Air Operations Division, PO Box 4331, Melbourne, Australia, 3001. Email: sam.waugh,simon.goss@dsto.defence.gov.au

NOTE TO REVIEWERS

Thank you for your careful reading and excellent comments on the last draft of this paper.

With one exception (Described below), all your comments have been applied to this new draft:

1. An intuitive overview has been added to the introduction. The main result is now shown and discussed in the intro (see the new figures 1,2,3). The following text has been added to the introduction to outline our core intuition:

This paper presents a non-naive renaming strategy. This new strategy is based on the following simple intuition. Much of the search space shown in Figure 2 is the same structure, repeated over and over again. It seems at least possible that no path can be found through $T + 1$ copies that can't be found in T copies (since the space is essentially the same). If this were true, then we could reduce the search space of temporal abductive validation by not copying the structure at all.

2. There were all-too-many typos in the last draft; e.g. wrong spelling of “foreign”. This draft was checked by a professional proof-reader.
3. The meaning of the edge annotations has moved closer to the place where the edges are first used. For example, in Fig 1, “++” and “-” are defined right in the caption. The meaning of “+-” is right next to its first usage under the text describing fig11.
4. Extra notes on the relationship of “++” and “-” to the “M+” and “M-” of QSIM have been added to the related word section.

Reviewer 2 suggested augmenting the examples in section III with more intuitive labels than “a,b,c,...”. We did that, and found that it added a significant number of pages to the total size of the article since it expanded all the diagrams 9,10,11,12,13,14,15. As we understand the size restrictions at KDE, this expansion is not acceptable. So we left those diagrams as before.

Abstract

We can test a theory of “X” by checking if that theory can reproduce known behaviour of “X”. In the general case, this check for time-based simulations is only practical for short simulation runs. We show that given certain reasonable language restrictions, the complexity of this check reduces to the granularity of the measurements. That is, provided a very long simulation run is only measured infrequently, this check is feasible.

Keywords

Validation, complexity, abduction, qualitative reasoning.

I. INTRODUCTION

We need thorough methods for testing our knowledge bases (KBs). Modern knowledge acquisition (KA) theorists view KB construction as the construction of inaccurate surrogate models of reality [1,2]. Agnew, Ford & Hayes [3] comment that “expert-knowledge is comprised of context-dependent, personally constructed, highly functional but fallible abstractions”. Practitioners confirm just how inaccurate KBs can be. Silverman [4] cautions that systematic biases in expert preferences may result in incorrect/incomplete knowledge bases. Compton [5] reports expert systems in which there is always one further important addition, one more significant and essential change. Working systems can contain multiple undetected errors. Preece & Shinghal [6] document five fielded expert systems that contain numerous logical anomalies. Myers [7] reports that 51 experienced programmers could only ever find 5 of the 15 errors in a simple 63 line program, even given unlimited time and access to the source code and the executable.

Potentially inaccurate and evolving theories must be validated, lest they generate inappropriate output for certain circumstances. Testing can only demonstrate the presence of bugs (never their absence) and so must be repeated whenever new data is available or a program has changed. That is, validation is an essential, on-going process throughout the lifetime of a knowledge base. This view motivated Feldman & Compton [8], then Menzies & Compton [9], to develop QMOD/HT4: a general technique for automatically validating theories in *vague domains*. A vague domain is (i) poorly-measured; and/or (ii) lacks a definitive oracle; and/or (iii) is indeterminate/non-monotonic. Validation in such vague domains necessitates making assumptions about unmeasured variables and

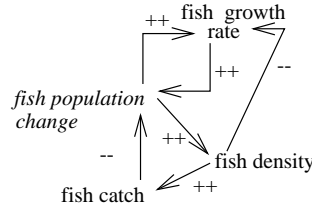


Fig. 1. A theory. A theory connected devices with two states (up \uparrow and down \downarrow). In this theory $x \xrightarrow{++} y$ denotes that $y\uparrow$ and $y\downarrow$ can be explained by $x\uparrow$ and $x\downarrow$ respectively; and $x \xrightarrow{-} y$ denotes that $y\uparrow$ and $y\downarrow$ can be explained by $x\downarrow$ and $x\uparrow$ respectively.

maintaining mutually exclusive assumptions in separate worlds. Many domains tackled by modern KA are vague; i.e. this definition of test is widely applicable.

Formally, QMOD/HT4 is abduction, which is known to be NP-hard [10]; i.e. theoretically the system is impractical since its runtimes are very likely to be exponential on theory size ($O(2^N)$). Nevertheless, abductive validation has been able to detect previously invisible but significant flaws in the theories published in the international neuroendocrinological literature [8, 9]. Further, Menzies has shown [11] that abductive validation is practical for at least the sample of fielded expert systems studied by Preece & Shinghal. However, standard abductive validation is restricted to *non-temporal* theories with the invariant that no variable can have two different values (§II). In the case of time-based simulation, this invariant is inappropriate since variables can have different values at different times.

One way to extend abductive validation to *temporal abductive validation* is to rename variables at each time point in the simulation. For example, all the variables in Figure 1 could be renamed e.g. *fishCatch1*, *fishCatch2*...*fishCatchT* where T is some time point. A naive renaming strategy would use these renamed variables to create T copies of the theory as seen in Figure 2. This naive renaming strategy incurs a severe computational cost; i.e. if a non-temporal theory has N variables and is $O(2^N)$, then for T time points, its temporal equivalent is $O(2^{N*T})$ (§III).

This paper presents a non-naive renaming strategy. This new strategy is based on the following simple intuition. Much of the search space shown in Figure 2 is the same structure, repeated over and over again. It seems at least possible that no path can be found through $T + 1$ copies that can't be found in T copies (since the space is essentially the same). If this were true, then we could reduce the search space of temporal abductive

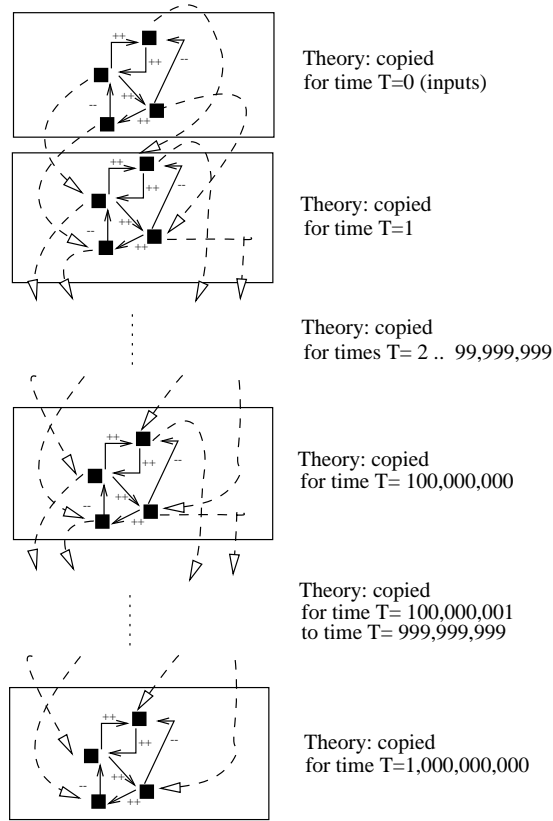


Fig. 2. The theory of Figure 1, copied $10^9 + 1$ times. The copy at time $T = 0$ is used for the inputs to the simulation. Dashed edges denote links between variables at different times. For space reasons, some of the copies not shown.

validation by not copying the structure at all.

We show below (in §III) that this intuition is incorrect, unless we carefully restrict how variables are linked in our theories. A *linking policy* is the method of connecting variables at T_i to T_{i+1} . We will show that if we apply the *implicit symmetric edge* linking policy (described below), then the search space of a theory with K states *saturates* after K renamings; i.e. if a proof does not terminate in time K , then no such proof exists (§IV). The practical implications of saturation are that, under certain circumstances, we can ignore a large subset of the renamed variables at unmeasured time points. Suppose all the variables in Figure 1 had $K = 2$ states (e.g. they were variables states: $\text{up}\uparrow$, $\text{down}\downarrow$). Suppose further we had run that model for a billion (10^9) time steps. Suppose further that we only have data from that simulation at three time points: say, initially, at $T = 10^8$ and $T = 10^9$. Without implicit symmetric edge linking, then when searching for proofs

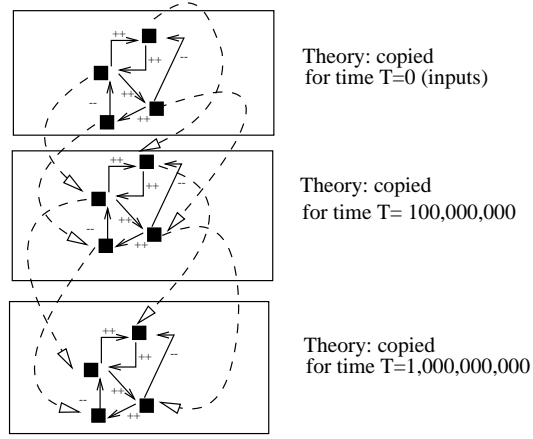


Fig. 3. Assuming the simulation is measured only three times, and the 2-state variables are connected with implicit symmetric edges, then the search space from the $10^9 + 1$ copies of Figure 2 reduces to the 3 copies shown here.

of these measurements, we would have to explore the space created by all 10^9 renamings shown in Figure 2. However, with implicit symmetric edge linking, we only need explore $\frac{3}{10^9}$ of that space, as shown in Figure 3 (§V).

Our approach requires a language that is more restrictive than that used in standard qualitative reasoning such as QSIM [12, 13]. Nevertheless, we argue that this restriction is both practical and desirable:

- §VI is an experimental demonstration that these language restrictions still permit the simulation and validation of real-world theories.
- In our related work section (§VII), we will note that standard qualitative reasoning systems cannot guarantee a tractable simulation for all models represented in that system. By comparison, we can guarantee a tractable simulation for all theories written in our language, provided that a very long simulation run is only measured infrequently.

II. NON-TEMPORAL ABDUCTIVE VALIDATION

This section contains our standard description of non-temporal abductive validation.

A. Tutorial

Abduction is the search for assumptions **A** which, when combined with some theory **T** achieves some set of goals **OUT** without causing some contradiction [14]. That is:

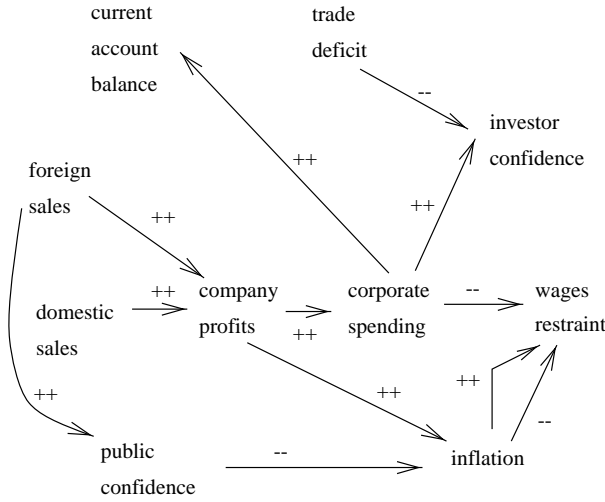


Fig. 4. A theory.

- $EQ_1: T \cup A \vdash \text{OUT}$;
- $EQ_2: T \cup A \not\vdash \perp$.

Menzies' HT4 abductive inference engine [15] caches the proof trees used to satisfy EQ_1 and EQ_2 . These are then sorted into *worlds* W : maximal consistent subsets (maximal with respect to size). In the case of multiple worlds being generated, the best world(s) are those with maximum *cover*: the intersection of that world and the OUTputs.

For example, consider the task of checking that we can achieve certain OUTputs using some INputs across the KB shown in Figure 4. We denote $x=\text{up}$ as $x\uparrow$ and $x=\text{down}$ as $x\downarrow$. In that figure, $x \overset{++}{\rightarrow} y$ denotes that $y\uparrow$ and $y\downarrow$ can be explained by $x\uparrow$ and $x\downarrow$ respectively; and $x \overset{--}{\rightarrow} y$ denotes that $y\uparrow$ and $y\downarrow$ can be explained by $x\downarrow$ and $x\uparrow$ respectively. Edges in our theories are optional inferences. The utility of using an edge is assessed via its eventual contribution to world coverage.

In the case of the observed OUTputs being $\{\text{investorConfidence}\uparrow, \text{wagesRestraint}\uparrow, \text{inflation}\downarrow\}$, and the observed INputs being $\{\text{foreignSales}\uparrow, \text{domesticSales}\downarrow\}$, HT4 can connect OUTputs back to INputs using the proofs of Table I. These proofs may contain controversial assumptions; i.e. if we can't believe that a variable can be both *up* and *down*¹, then we can declare the known values for *companyProfits* and *corporateSpending* to be controversial. Since *corporateSpending* is fully dependent on *companyProfits* (see

¹Note: in the temporal abductive case, this rule would be a variable can be *up* and *down* at the same time.

TABLE I

PROOFS FROM FIGURE 4 CONNECTING $OUT = \{\text{investorConfidence}\uparrow, \text{wagesRestraint}\uparrow, \text{inflation}\downarrow\}$ BACK TO $INPUTS = \{\text{foreignSales}\uparrow, \text{domesticSales}\downarrow\}$.

P[1]:	$\text{domesticSales}\downarrow, \text{companyProfits}\downarrow, \text{inflation}\downarrow$
P[2]:	$\text{foreignSales}\uparrow, \text{publicConfidence}\uparrow, \text{inflation}\downarrow$
P[3]:	$\text{domesticSales}\downarrow, \text{companyProfits}\downarrow, \text{corporateSpending}\downarrow, \text{wagesRestraint}\uparrow$
P[4]:	$\text{domesticSales}\downarrow, \text{companyProfits}\downarrow, \text{inflation}\downarrow, \text{wagesRestraint}\uparrow$
P[5]:	$\text{foreignSales}\uparrow, \text{publicConfidence}\uparrow, \text{inflation}\downarrow, \text{wagesRestraint}\uparrow$
P[6]:	$\text{foreignSales}\uparrow, \text{companyProfits}\uparrow, \text{corporateSpending}\uparrow, \text{investorConfidence}\uparrow$

Figure 4), the key conflicting assumptions are $\{\text{companyProfits}\uparrow, \text{companyProfits}\downarrow\}$ (denoted *base controversial assumptions* or A.b). We can use A.b to find consistent belief sets called worlds W using an approach inspired by the ATMS [16]. A proof P[i] is in W[j] if that proof does not conflict with the environment ENV[j] (a maximal consistent subset of A.b). In our example, $ENV[1] = \{\text{companyProfits}\uparrow\}$ and $ENV[2] = \{\text{companyProfits}\downarrow\}$. Hence, $W[1] = \{P[2], P[5], P[6]\}$ and $W[2] = \{P[1], P[2], P[3], P[4]\}$ (see Figure 5 and Figure 6).

HT4 defines *cover* to be size of the intersection of a world and the OUTPUT set. The cover of Figure 5 is 3 (100%) and the cover of Figure 6 is 2 (67%). Note that since there exists a world with 100% cover, then all the OUTPUTs can be explained; i.e. this theory has passed the abductive validation test.

In essence, abductive validation answers the following question: “what portions of a theory of X can reproduce the largest % of known behaviour of X?”. This algorithm will work in two hard cases:

1. Only some subset of known behaviour can be explained.
2. In the case where a theory is globally inconsistent, but contains useful portions. For example, observe in Figure 4 that the theory authors’ disagree on the connection from *inflation* to *wagesRestraint*.

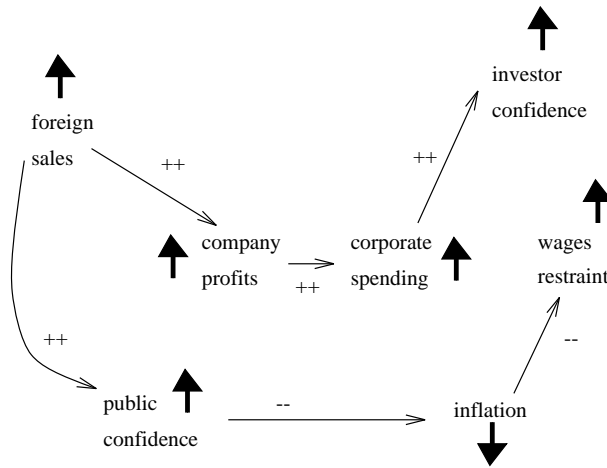


Fig. 5. World #1 is generated from Figure 4 by combining P[2], P[5], and P[6]. World #1 assumes `companyProfits`↑ and covers 100% of the known OUTputs.

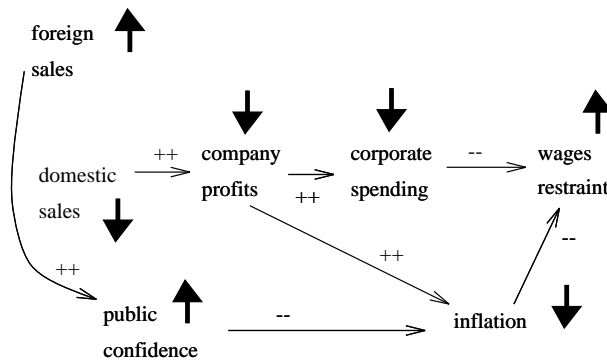


Fig. 6. World #2 is generated from Figure 4 by combining P[1], P[2], P[3], and P[4]. World #2 assumes `companyProfits`↓ and covers 67% of the known OUTputs.

Note that this inference procedure ignores certain possible inferences; e.g. in $W[1]$, `tradeDeficit`↓ and `currentAccountBalance`↑. HT4 does not compute ATMS-style *total envisionments*; i.e. all state assignments consistent with known facts. A total envisionment would have included `tradeDeficit`↓ and `currentAccountBalance`↑. Nor does HT4 compute QSIM-style *attainable envisionments* [12, 13]; i.e. the subset of total envisionments downstream of the INputs. An attainable envisionment would have included `currentAccountBalance`↑. Rather, HT4 restricts itself to the inferences that connect INputs to OUTputs. That is, HT4 only computes *relevant envisionments*; i.e. the subset of the attainable envisionments which are upstream of the OUTputs. The extra inferences

of non-relevant envisionment may result in pointless world generation. For example, if somehow `currentAccountBalance↑` was incompatible with `wagesRestraint↑`, then total or attainable envisionments would divide $W[1]$ into at least two additional worlds, each of which would contain some subset of the literals shown in Figure 5. The new additional world containing `currentAccountBalance↑` would subsequently be ignored since the new additional world with `wagesRestraint↑` would be returned during the search for maximum cover.

B. Complexity

This section describes our algorithm for solving the core problem of HT4: finding the base controversial assumption set $A.B$. This algorithm will be shown to be theoretically NP-hard, experimentally exponential, but practical for certain problems.

HT4 executes in four phases: the *facts sweep*, the *forwards sweep*, the *backwards sweep*, and the *worlds sweep*. Firstly, the *facts sweep* removes all variable assignments inconsistent with known **FACTS** (typically, $\mathbf{FACTS} = \mathbf{IN} \cup \mathbf{OUT}$). Using a hash table, the facts sweep runs in linear time.

Secondly, the *forward sweep* finds the conflicting assumption set (denoted $A.c$) as a side-effect of computing the transitive closure of \mathbf{IN} (denoted \mathbf{IN}^*). In a theory comprising a directed graph with vertices V , edges E , and fanout $F = \frac{|E|}{|V|}$, the worst-case complexity of the forwards sweep is acceptable at $O(|V|^3)$. Note that if the theory lacks invariants, then the validation process could stop at this point since the transitive closure would find the **OUT**puts reachable from the **IN**puts. However, in theories with invariants, it may be the case that we can only consistently use portions of the theory and **IN**puts to achieve some subset of the **OUT**puts.

Thirdly, the *backwards sweep* grows proofs backwards from a member of **OUT** back to **IN** while maintaining several invariants. (i) Proofs can only use members of \mathbf{IN}^* ; i.e. only those literals downstream of the **IN**puts. (ii) Proofs maintain a *forbids* set; i.e. a set of literals that are incompatible with the literals used in the proof. For example, the literals used in $P[1]$ forbid the literals $\{\text{domesticSales}\uparrow, \text{companyProfits}\uparrow, \text{inflation}\uparrow\}$. (iii) The upper-most $A.c$ found along the way is recorded as that proof's *guess*. The union of all the guesses of all the proofs will be $A.b$. (iv) A proof must not contain loops. (v) A

```

procedure worldsSweep begin
  ENV := maximalConsistentSubsets(A.b)
  for i := 1 to size(ENV) begin
    W[i] :=  $\emptyset$ ;
    for p  $\in$  P
      if    p.forbids  $\cap$  ENV[i] =  $\emptyset$ 
      then W[i] := W[i] + p;
    end
  end
end

```

Fig. 7. The worlds sweep of HT4.

proof must not contain items that contradict other items in the proof; i.e. a proof's members must not intersect with its *forbids* set. We can demonstrate informally and formally that the backwards sweep is a slow process:

- *Informally:* If the average size of a proof is $|\overline{P[i]}|$, then worst case backwards sweep is $O(|\overline{P[i]}|^F)$. To make matters worse, the backwards sweep cannot cull its search at a local propagation level. The utility of an edge may not be apparent till we have examined the search space accessed after using that edge.
- *Formally:* Bylander *et. al.* [10] showed that general abduction is NP-hard. For our particular implementation, we can repeat that prior result since we can show that satisfying invariant (v) is NP-hard. Clearly, we can find a theory to generate any directed graph. Gabow *et. al.* [17] showed that finding a directed path across a directed graph that has at most one of a set of forbidden pairs is NP-hard. Our forbidden pairs are assignments of different values to the same variable; e.g. the pairs $x\uparrow \& x\downarrow$ and $x\downarrow \& x\uparrow$ are forbidden.

Fourthly, once A.b is known, then the proofs can be sorted into worlds via the *worlds sweep*. HT4 extracts all the objects 0 referenced in A.b. A world-defining environment ENV[i] is created for each combination of objects and their values. In our example, ENV[1] = {c \uparrow } and ENV[2] = {c \downarrow }. The worlds sweep is simply two nested loops over each ENV[i] and each P[j] (see Figure 7). A proof Pj belongs in world W[i] if its *forbids*

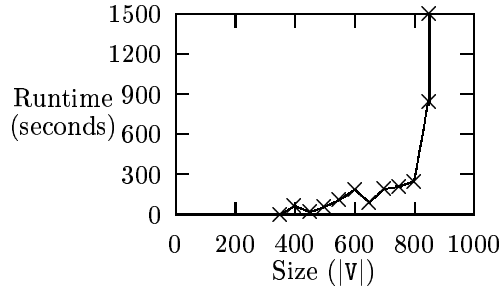


Fig. 8. Average runtimes.

set does not intersect the assumptions $\text{ENV}[i]$ that define that world. The worlds sweep is exponential at $O(|P| * |\text{ENV}|) = O((\overline{|P[i]|})^F * |\text{ENV}|)$.

Menzies [11] reports exponential runtimes of HT4 in a *mutation study*. Hundreds of theories were artificially generated by adding random vertices and edges to the Smythe '89 theory of glucose regulation [18]. For this runtime study, the fanout of the theory was kept constant. Figure 8 shows the average runtime for executing the system over 94 mutated theories and 1991 randomly chosen $\langle \text{IN}, \text{OUT} \rangle$ pairs [11]. For that study, a “give up” time of 840 seconds was built into the system. The system did not terminate for $|V| \geq 850$ in under that “give up” time (shown in Figure 8 as a vertical line); i.e. the “knee” in the exponential runtime curve kicks-in at around 800 vertices.

Nevertheless, non-temporal abductive validation is a practical validation algorithm since there exist real-worlds theories which it can validate. Of the fielded expert systems sampled by Preece & Shinghal [6], none had more than 510 literals in their dependency graphs. The neuroendocrinological theories studied by Feldman, Compton, & Menzies had less than 550 literals in their dependency graphs. However, HT4 contains an NP-hard backwards sweep followed by a exponential worlds sweep. Experimentally, exponential runtimes have been observed in one implementation. Hence, we have theoretical and experimental reasons to reject proposals which significantly increase the size of our theories; e.g. the naive renaming for temporal abductive validation discussed in the introduction and elaborated below (§III).

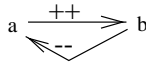


Fig. 9. A theory.

III. TEMPORAL ABDUCTIVE VALIDATION

In at least two commonly used knowledge representations, the truth value of literals can change over the lifetime of the simulation:

1. Rule-bases that are processed via a standard match-select-act loop may assert and retract facts many times during processing.
2. As the inference executes over loops in qualitative theories or topoi graphs [19], literals may be assigned different belief values at different times.

We use the term “temporal abductive validation” to denote the problem of validating theories where the belief values of literals can change during a simulation. One approach to implementing temporal abductive validation would be to create one copy of the theory for each time interval T in the simulation. The variables at time i would be created by renaming each variable X in the theory Xi for $i = 1 \dots T$. Once the renaming was completed, then we could apply some *linking policy* to make connections between $T = i$ and $T = i + 1$. For example, using the *implicit edge* linking policy (hereafter *IEDGE*), if the theory says $X \rightarrow Y$, then we would connect $X_{T=i} \rightarrow Y_{T=i+1}$. For example, for a simulation of Figure 9 over 3 time intervals, we could execute HT4 over Figure 10². The disadvantage of this approach is that the graph size would increase with the number of copies. Given the complexity results of §II-B, this is undesirable.

Note that the copies of Figure 10 repeat the structure of Figure 9 at each time interval in the simulation. Recall the introduction, a plausible intuition is that no explanation path can be found through $T + 1$ copies that can’t be found in T copies since the space is essentially the same. If this were true, then we could reduce the search space of temporal abductive validation by not copying the structure at all.

Menzies & Cohen [20] showed that this optimisation is not possible for unrestricted languages; i.e. if we allow (e.g.) *asymmetric edges* in our theories. *Symmetric edges* model

²Implementation note: internally, we could execute over one copy of the theory and use a stack variable to denote the current time.

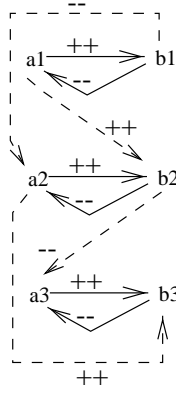


Fig. 10. Figure 9 renamed over 3 time intervals and connected using IEDGE . Dashes denote time edges.

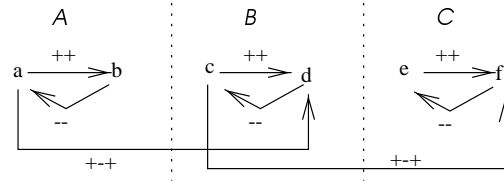


Fig. 11. A theory with asymmetric edges from a to d and from c to f .

the kinds of influences we see in simple mathematics. For example, $x \xrightarrow{++} y$ and $x \xrightarrow{--} y$ are examples of *symmetric edges* in that they make some statement about every state of the downstream vertex y . $x \xrightarrow{+-+} y$ is an example of an asymmetric edge: $y \uparrow$ could be explained by $x \uparrow$, but not vice versa. Asymmetric edges are useful for (e.g.) modeling in-flows that pour into the top of a tank and out-flows which drain from the bottom of the tank. If an in-flow increases, then the tank level could increase. However, the converse is not true since only if the tank out-flow increases will the tank level decrease. For example, Figure 11 contains two asymmetric edges ($a \xrightarrow{+-+} d$; $c \xrightarrow{+-+} f$) which, internally, has the search space of Figure 13. Figure 11 duplicates the topology of Figure 9 in the regions A, B, C with an extra link from the top-left vertex of one region to the top-right vertex of the next region.

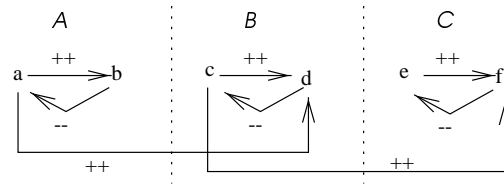


Fig. 12. Figure 11 with asymmetric edges replaced by symmetric edges.

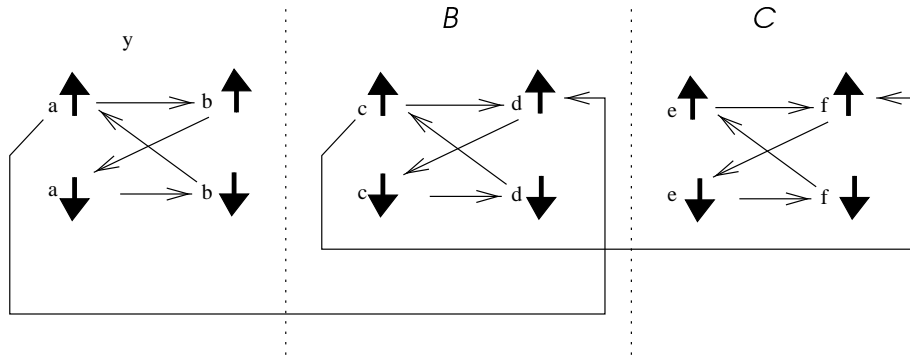


Fig. 13. The search space within Figure 11.

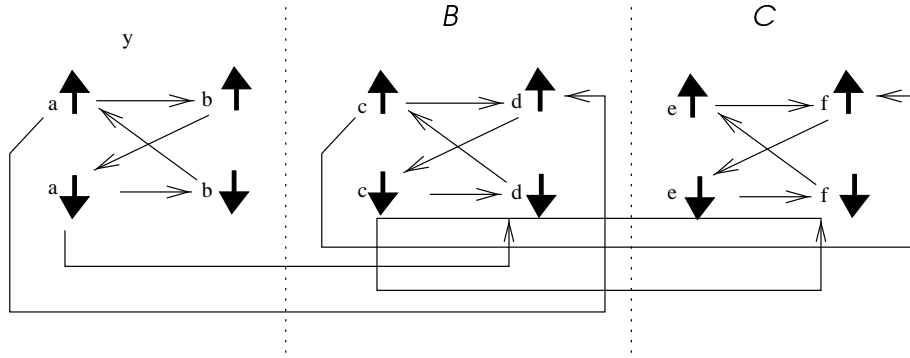


Fig. 14. The search space within Figure 12.

A path from $b↑$ to $e↑$ will take at 3 time intervals to cross from top-left to top-right in each of the regions A, B, C . By repeating A, B, C more times, we can generate dependency graphs which would require any number of intermediaries to traverse. This example suggests that between each measured time interval we may need many intermediary copies.

However, if we restrict our language appropriately, then something close to our above plausible intuition is true. While exploring examples like Figure 11, we noted that if we restrict ourselves to only symmetric edges, then we could not find an example where proofs took longer than two time intervals. For example, Figure 12 replaces the asymmetric edges of Figure 11 with symmetric edges. Figure 12 expands into the search space of Figure 14 in which we can connect $b↑$ to $e↑$ in two time intervals (one from $b↑$ to $f↓$, then two from $f↓$ to $e↑$). In the next section, we will demonstrate that this “2 time intervals is enough” is a general result.

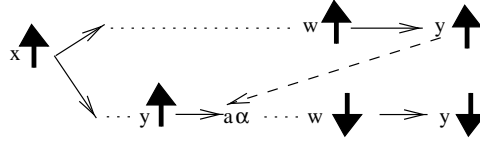


Fig. 15. Proofs in two time intervals or less. The dashed line from $y\uparrow$ to $a\alpha$ is a time edge. Dotted edges denote any number of non-time edges.

IV. PROOF

This section describes the main result of this paper; i.e. given a theory comprising K -state devices connected by symmetric edges, then if a proof cannot be found in K time intervals, it can never be found at all. Roughly speaking, if every edge offers a comment on all the states of its downstream vertices (i.e. they are symmetric), then the state space rapidly *saturates*. That is, the granularity of the time axis reduces to the number of states in the theory. Hence, all the connectable points in the state space can be found very quickly. The formal proof for the K -state case is intricate. Hence, we first show an informal proof of the 2-state case before presenting the K -state proof.

A. Informal Proof For 2-State Devices

Consider a proof over theory using two state devices (say, up and down, denoted \uparrow and \downarrow respectively) and symmetric edges. This proof has a start (e.g. $x\uparrow$) and a goal. If that goal can be reached without crossing anything that contradicts the goal (hereafter, the *simplest case*), then this proof can be achieved in one time interval. This case is the top path of Figure 15.

Now consider the case in which the goal (e.g. $y\downarrow$) can only be reached via a conflicting value (e.g. $y\uparrow$, see the bottom path of Figure 15). The only case in which such a proof can terminate is when there is a path from the conflicting value to the goal value; e.g. $y\uparrow$ to $y\downarrow$). If such a path exists, then with EDGE linking, there must be a time edge from the conflicting value to its neighbor; e.g. $y\uparrow$ to $a\alpha$ where α denotes some value assignment to a (either $a\uparrow$ or $a\downarrow$). This means that we can use the simplest case to get to the conflicting value in one time interval and then we can use the time edge to move on to the goal (see the dashed edge in Figure 15). Note that moving on to the goal (via $a\alpha$ to $w\downarrow$ to $y\downarrow$) will take no more than one more time interval since the only literal that could block the

process (e.g. $y\uparrow$) has already been used (this is proved formally below).

The only other interesting case is that this proof needs to cross other conflicting values (e.g. $w\uparrow$ and $w\downarrow$). Using the simplest case, we can cross one of these conflicting values before the time edge via the base case, and the other one afterwards (see Figure 15). Hence, these other conflicting values will not further delay the proof.

By symmetry, this example can be repeated for proofs from $x\downarrow$ to $y\uparrow$ or $y\downarrow$, and for proofs from y to x . Hence, if a proof exists, it must be found in at least two time intervals.

B. Formal Proof for K -State Devices

Formally, a *theory digraph* is a directed graph G consisting of vertices representing *variables* and edges representing connections between the states of the variables. Each variable x can have K states, x_1, \dots, x_K . A directed edge e of G from x to y represents a collection of dependencies of the states of y on the states of x . For theory digraph G , we use these dependencies to construct a *dependency digraph* G_D . For each vertex x of G , there is a vertex x_s in G_D for each state of x . There is an edge from vertex x_s to vertex y_t if y_t is dependant on x_s in some relationship represented by an edge of G . For simplicity, in this paper we refer to vertices in a theory digraph as *variables* and vertices in a dependency digraph as *states*. For an edge e of a theory digraph G , we define $image(e)$ to be the collection of edges of G_D generated by e (see Fig. 16). Similarly, for a path P of G , $image(P)$ is the subgraph formed by the union of the images of the edges of P .

A directed edge e in G from x to y is *symmetric* if the edges in $image(e)$ form a bijection between the the states of x and the states of y (see Fig. 16). In this section, we consider only theories where all edges are symmetric.

Consider states u_q and v_r in dependency digraph G_D . A *simple proof* of v_r from u_q , $\Pi^s(u_q, v_r)$, is a directed path from u_q to v_r such that for any variable z in G , at most one state of z is on $\Pi^s(u_q, v_r)$.

A *proof* of state y_t from state x_s , $\Pi(x_s, y_t)$ is an ordered collection of simple proofs such that x_s is the start of the first path, y_t is the end of the final path, and there is an edge in G_D from the final state of each simple proof to the start state of its successor (see Fig 17). Note that for a state z of G , multiple states of z can be contained in proof $\Pi(x_s, y_t)$, but at most one state of z can be contained in the same simple proof of $\Pi(x_s, y_t)$.

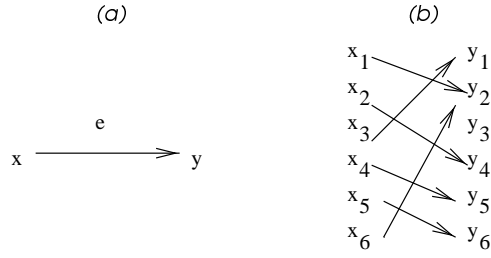


Fig. 16. The relationship between an edges of a theory digraph and its image in the associated dependency digraph. (a) Directed edge e from variable x to variable y in a 6-state theory. (b) The edges in $image(e)$.

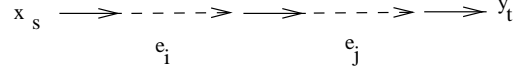


Fig. 17. A proof of y_t from x_s . Edges e_i and e_j are time edges and are shown as dotted lines. Solid lines are simple proofs containing one or more non-time edges.

We think of each simple proof of proof $\Pi(x_s, y_t)$ as occurring in its own time quantum. The time of a proof, $time(\Pi(x_s, y_t))$ is the number of simple proofs in $\Pi(x_s, y_t)$. We call the edges connecting the simple proofs *time edges*. The minimum proof time, $minTime(x_s, y_t)$, is the minimum time of any proof of y_t from x_s . If there is no proof of y_t from x_s , then $minTime(x_s, y_t)$ is undefined.

Consider a directed path P in G from vertex x to vertex y . Define $reachable(P, x_s)$ to be the collection of states of y that are reachable from x_s in $image(P)$. We also define path P' to be a *prefix* of path P if P' is a subpath of P and P and P' share the same first vertex.

Lemma 1: Let P be a directed path in theory digraph G from variable x to variable y and P' be a prefix of P . Then $|reachable(P', x_s)| \leq |reachable(P, x_s)|$.

Proof: The proof is by a simple inductive argument on the length of P . If path P has length 0, then $x = y$ and only x_s is reachable from x_s so $|reachable(P, x_s)| = 1$ and the only prefix is P itself so the lemma holds.

Now suppose the lemma is true for all paths of length k or less. Suppose P has length $k + 1$. Let vertex w be the predecessor of y on P and let P' be the path from x to w formed by removing the final edge e of P . Symmetric edges imply that the edges in $image(e)$ form a bijection between the states of w and the states of y . Therefore, for each

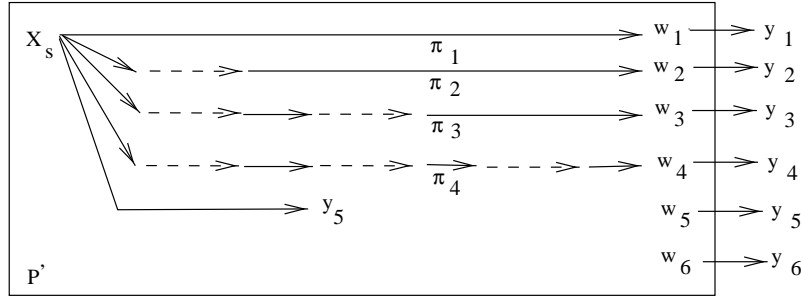


Fig. 18. Illustration of the proof of lemma 1. There are paths to states y_1 , y_2 , y_3 , and y_4 through the states of $|reachable(P')|$. $|reachable(P', x_s)| < |reachable(P, x_s)|$ since state y_w is reachable from some prefix of P' . Dashed lines are time edges; solid lines are one or more non-time edges.

w_u in $reachable(P', x_s)$ there is a unique state of y reachable from x_s via a path through w_u , so $|reachable(P', x_s)| \leq |reachable(P, x_s)|$ (see Fig. 18). Since any prefix of P is either P itself or a prefix of P' the lemma is proved.

Note that it is possible for $|reachable(P', x_s)| < |reachable(P, x_s)|$ since P may contain cycles and other states of y may have been reached in some other prefix of P . ■

Lemma 2: Let P be a directed path in theory digraph G from variable x to variable y and x_s be a state of x with $|reachable(P, x_s)| = r$. For each integer i in $1 \dots r$, we can find a unique state y_j of y such that $minTime(x_s, y_j) \leq i$ when only considering proofs with all edges in $image(P)$.

This lemma is the key to our result. Before proving the lemma, consider the following example. Figure 18 represents $image(P)$ for some path P from variable x to variable y . Path P' is the prefix of P formed by removing the final edge e from w to y .

Suppose we have determined that $|reachable(P', x_s)| = 4$ and for each $1 \leq i \leq 4$ we have $minTime(x_s, w_i) \leq i$. We iterate finding states of y to satisfy the lemma:

- *Iteration 1:* Since there are no states of y on Π_1 , we can extend Π_1 by e_1 to demonstrate that $minTime(x_s, y_1) = 1$.
- *Iteration 2:* State Y_3 is on the final simple proof of proof Π_2 , so we can conclude that $minTime(x_s, y_3) \leq 2$.
- *Iteration 3:* Since we can consider edge e_2 a time edge, we demonstrate that $minTime(x_s, y_2) \leq 3$.
- *Iteration 4:* State Y_2 is on the final simple proof of proof Π_4 . We can use the already

determined proof of y_2 to demonstrate that $\minTime(x_s, y_4) \leq \minTime(x_s, y_2) + 1 \leq 4$.

- *Iteration 5:* Since a proof of y_4 was found earlier (by induction), we have that $\minTime(x_s, y_4) \leq |\text{reachable}(P', x_s)| = 4 < 5$.

Proof: Let P be a path from vertex x to vertex y . The proof is by induction on the length of P . If P has length 0, then $x = y$ and only x_s is reachable from x_s so $|\text{reachable}(P, x_s)| = 1$ and $\minTime(x_s, x_s) = 1$.

Now suppose the lemma is true for all paths of length k or less. Suppose P has length $k + 1$. Let vertex w be the predecessor of y on P and P' be the path from x to w formed by removing the final edge e of P .

We partition the states of y into three categories:

- Y_1 : the states of y that are connected from a state in $\text{reachable}(P', x_s)$ by an edge in $\text{image}(e)$.
- Y_2 : the states in $\text{reachable}(P, x_s)$ not in Y_1
- Y_3 : the states not in $\text{reachable}(P, x_s)$.

Assume without loss of generality, that the states of w are ordered such that for $i < |\text{reachable}(P', x_s)|$, $\minTime(x_s, w_i) \leq i$. Such an ordering is possible by the inductive hypothesis. Assume also that the edges of $\text{image}(e)$ are labeled e_1, \dots, e_K and the states of y are numbered such that:

- edge e_i connects state w_i to state y_i ,
- $y_{|Y_1|+1}, \dots, y_{|\text{reachable}(P, x_s)|}$ are the states of Y_2 in any order.
- $y_{|\text{reachable}(P, x_s)|+1}, \dots, y_K$ are the states of Y_3 in any order.

Note that our definition implies that $y_1, \dots, y_{|Y_1|}$ are the states of set Y_1 .

We now show how we bound the values of $\minTime(x_s, y_i)$ in such a way to satisfy the lemma. Let *count* indicate the number of states of y labeled at anytime in our algorithm. For each value $1 \leq \text{count} \leq |\text{reachable}(P, x_s)|$ we need to find a unique state y_i such that $\minTime(y_i) \leq \text{count}$:

While $\text{count} < |\text{reachable}(P, x_s)|$ do

- Increment *count*
- Choose the smallest i such that y_i is in either set Y_1 or set Y_2 and we have not already bounded $\minTime(x_s, y_i)$ for path P . By our structure, we know that $i \leq \text{count}$.

- If y_i in Y_1
 - If $i < count$ since we know by induction that $minTime(x_s, w_i) \leq i$, if we consider edge e_i a time edge, we get $minTime(x_s, y_i) = i + 1 \leq count$.
 - else let Π be a proof of state w_i from x_s with $time(\Pi) \leq i$. Let Π^s be the final simple proof of Π . If Π^s contains state $y_j \neq y_i$:
 - * if we have already determined a bound for $minTime(x_s, y_j)$ for path P , then we know that $minTime(x_s, y_j) < count$. Let v_ℓ be the successor of the last occurrence of y_j on Π^s connected to y_j by edge e_ℓ . We form a proof of y_i as follows. Use a proof Π_j of y_j with $time(\Pi_j) < count$. Consider edge e_ℓ a time edge. Then the suffix of Π^s from v_ℓ to w_i plus edge e_i form a simple proof (since Π^s is a simple proof), and the only state of y on this path is y_i , so $minTime(x_s, y_i) \leq count$.
 - * else we can truncate proof Π at state v_j , so $minTime(x_s, y_j) = i \leq count$.
 - else the only possible state of y in simple proof Π^s is y_i . Therefore, the path formed by adding edge e_i to Π^s is a simple proof, so we get $minTime(x_s, y_i) = i \leq count$.
- else state y_i is in set Y_2 . Therefore, there is a prefix P'' of P' such that y_i is in $reachable(P'', x_s)$. By lemma 1, we know that $|reachable(P'', x_s)| \leq |reachable(P', x_s)|$. By our construction, $i > |Y_1| = |reachable(P', x_s)|$. We get, by induction, $minTime(x_s, y_i) \leq |reachable(P'', x_s)| \leq |reachable(P', x_s)| < count$.

■

Theorem 1: Consider a theory digraph G with K states per variable. If there is a path from vertex x_s to vertex y_t in dependency graph G_D , then $minTime(x_s, y_t) \leq K$.

Proof: Find a path P in G with y_t in $reachable(P, x_s)$. Then, as an immediate consequence of lemma 2, $minTime(x_s, y_t) \leq K$.

■

V. IMPLICATIONS

We can use the above proof as an optimisation technique as follows. Suppose we had a K -state device which was run for T_i time intervals and was measured at T_j time points. By “running” we mean that a search space was created with T_i renamings of the variables in the device. These renamings would be linked as per the connections in the original device *plus* implicit edge linking between variables from adjacent time points. In search for proofs of measurements, we would not need to explore further than K time edges away

from the measurement before we could declare a proof impossible.

Note that it would be a mistake to only generate K renamings of the search space. A proof from $T = a$ back to $T = c$ must be consistent with intermediary measurements at $T = b$ ($a < b < c$). Hence, the backwards sweep must build this proof from $T = c$, through $T = b$, then back to $T = a$. Hence, when generating the renamed variables, we would need to represent the search space at *and* between measured time points; i.e. $(K * T_j) - 1$ renamings. In the case where K was small and $T_j \ll T_i$, then this is still a significant reduction in the search space.

Returning to our example in the introduction, suppose:

- We had a $K = 2$ -state device with symmetric `IEDGE` linking which has been running for a billion time steps; i.e. $T_i = 10^9$ seconds.
- The number of literals in the device are less than the size limits shown in Figure 8; i.e. non-temporal abductive validation across this device was known to be tractable.
- We only collected measurements at three time points (say, 1 and $5 * 10^8$ and 10^9); i.e. $T_j = 3$.

Let the space complexity for non-temporal abductive validation across this device be X . If we had not proved saturation for our device, then the search for these proofs would have to explore 10^9 renamings; i.e. the space complexity would be $10^9 * X$. However, since we have proved saturation, 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 $5 * 10^8$. If that proof cannot terminate in the space 1 and $5 * 10^8$, then it will never terminate. That is, when building this proof, we could ignore the search space that used the renamings from 2 to $5 * 10^8 - 1$. A similar argument could be made for proofs from $5 * 10^8$ to 10^9 . In practice, we would only need to search the space created for the three measured time points. That is, the space requirements for this temporal validation problem would only be $2 * X$.

VI. EXPERIMENTS

The previous section discussed a general theoretical result relating to temporal abductive validation. This section describes a specific experimental result relating to studies on one model. We motivate this section as follows:

- In order to prove saturation, we have had to severely restrict our representation lan-

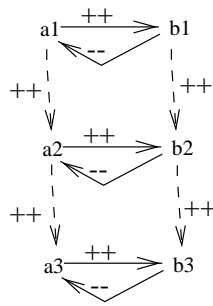


Fig. 19. Figure 9 renamed over 3 time intervals and connected using INODE.

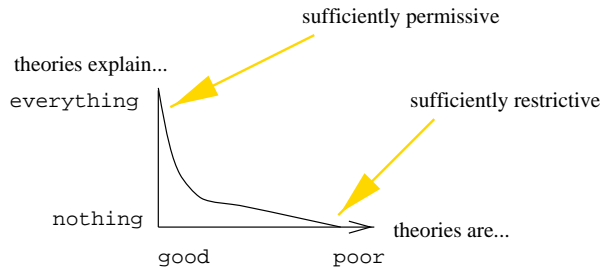


Fig. 20. Visualising a successful linking policy. A linking policy should be both *permissive* to all good theories to explain correct behaviour and *restrictive* enough to prevent poor theories explaining any behaviour.

guage. A reasonable objection to our approach is that these restrictions render our approach impractical. In this section, we show that it is not necessarily true that our approach is excessively restrictive since, at least in the example offered below, a practical system was developed.

- The experiment described below is a general framework for comparing the testability of different qualitative representations.
- An earlier version of this paper [20] offered a theoretical conclusion that 3 time copies were sufficient for the *implicit node* linking policy (hereafter, INODE) using two-state variables linked by symmetric edges. With implicit node linking, variables at time $T = i$ were linked to variables at time $T = i + 1$ (see Figure 19). However, subsequent experiments with that policy [21] showed that IEDGE could not distinguish good theories from poor theories. This section describes this experimental rejection of INODE and the experimental confirmation of the utility of IEDGE.

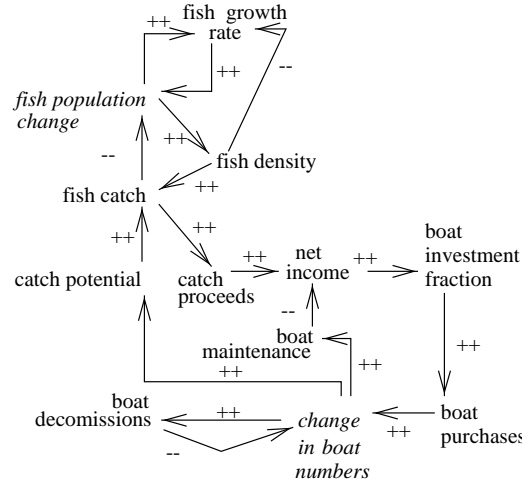


Fig. 21. The fisheries model. Adapted from [23, pp135-141]. Variables in italics were used in the `xnode` study (see text).

A. Experimental Design

In order to assess the practical merits of a linking policy, we need to step back and make a statement about the context in which such a policy would be used. Menzies & Goss [22] describe the *gray-box modeling problem* in which operators need to guess the inner workings of a black-box simulator using only minimal knowledge of that simulator's input-output. The operator records their guess in an approximate gray-box notation such as Figure 4. For our purposes, we say that a linking policy is adequate if it supports gray-box modeling. In particular, we say that a linking policy should be *permissive* enough to permit the generation of proofs for known correct behaviour and *restrictive* enough to block proofs of known incorrect behaviour. Further, we need to be able to quickly recognise if an operator's guesses move towards the correct theory.

Figure 20 is a succinct visual representation of an adequate linking policy. Given a range of theories which degrade from **good** to **poor**, we like to see curve of Figure 20; i.e. we quickly get feedback that we can explain progressively less and less of the behaviour of the theory.

Our test engine has four sub-routines: (1) representative model selection; (2) data generation; (3) model mutation; and (4) option comparison.

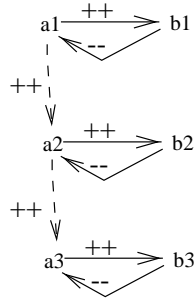


Fig. 22. Figure 9 renamed over 3 time intervals and connected using `XNODE` where the user has indicated that `a` is an explicit time node. Dashed lines indicate time edges.

(1) *Representative model selection*: The “fisheries simulation model” (Figure 21) is similar to theories developed by Goss in his domain. It includes feedback loops; qualitative states; and measurable entities. A precise mathematical expression of this model is available [23, pp135-141].

(2) *Data generation*: The selected quantitative model was run 15 times over five time steps to generate numeric test data using different input parameters to create an array of quantitative observations `measure[1..15]`. From each comparison of `measure[i]` with `measure[j]` ($i < j \leq 15$), 105 entries were written to an array of qualitative observations `changes[1..105]`. 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 `{fdens=up}` 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}`.

(3) *Model mutation*: This process must be repeated for a large number of representative theories from a domain. As these are hard to find in practice, we generate them using a variant of the mutation strategy used by Menzies (Figure 8). In this new mutator, a random sample of `X` statements in the qualitative form of the known representative model are corrupted. 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 Figure 20. We corrupted the model by flipping the annotation on an edge (e.g. `++` to `--` or vice versa). The corruption-model-mutator picks its edges to corrupt at random, and we repeat the corruption a statistically significant number of times (20 repeats).

(4) *Option exploration*: The two linking policies discussed above (`IEDGE`, `INODE`) were

compared with another policy we have proposed elsewhere [21]. The *explicit node* linking policy (hereafter, `xnode`) acts like `inode` except that domain experts explicitly label which nodes are to be connected across time. For the three linking policies (`iedge`, `inode`, `xnode`) and for each mutated model, we created six model copies `copy[0..5]`. `copy[i]` was connected to `copy[i+1]` as follows: if `iedge` was being used, then copies were connected as in Figure 10; otherwise, if `inode` was being used, then copies were connected as in Figure 19; otherwise, if `xnode` was being used, then copies were connected as in Figure 22. For the `xnode` policy, the variables shown in italics in Figure 21 were used as the explicit time nodes. `Change` inputs were mapped into `copy[0]`. `Change` outputs were mapped into some `copy[1..5]`. The success of each run was assessed using the generated data, by recording the % of the *explicable outputs* i.e. those outputs that the model could connect back to inputs. Returning to Figure 20, `everything=100%` explicable and `nothing=0%` explicable.

Proofs for PUTputs at time $T = 5$ must be consistent with proofs from $T = 1..4$. Hence, all the proofs must be built together (see `run_qualitative_model` in Figure 23). For this study, we only collected % explicable figures for OUTputs at time $T = 5$.

B. Results

Figure 24 shows the results of applying the test engine to the fisheries model. All policies could yield explanations for at least 20% of data, even for very poor theories. We attribute these *residual explanations* to the indeterminacy of qualitative theories.

The results show that two of the linking policies (`xnode` and `iedge`) were close to the ideal curve of Figure 20. `inode` was observed to be quite removed from the ideal. `inode` is clearly not sufficiently restrictive a linking policy since even with all edges corrupted, this linking policy allowed poor theories to explain correct behaviour. Hence, based on this experiment, we now reject our previous recommendations [20] regarding the use of `iedge`.

`xnode` was closest to the ideal curve of Figure 20. However, we must not uncritically endorse the `xnode` linking policy. Recall that the basis of this policy was that users could pick some subset of the vertices and mark them as explicit time traversal nodes. Our above proof of saturation of K -state devices in K time ticks depended on the regularities found in the search space found in `iedged` linked theories. The search space of an explicitly

Inputs:

- 1) the quantitative fisheries model M0
- 2) the qualitative fisheries model M1 with
 - edges E := edge[1..17]
- 3) T := 5 max time ticks
- 4) linkingPolicies := [iedge, inode, xnode]
- 5) Repeats := 20

Outputs: Xplicable, Runtime

```

measure[1..15] := run_quantitative_model(T,M0)
change[1..105] := comparisons(measure)
for policy ∈ linkingPolicies begin
  for corrupted:=0 to |E| begin
    for r:=1 to Repeats begin
      M2 :=corruptSomeEdgesChooosenAtRandom(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]
        startTime := timeNow()
        Xplained[1..T] := run_qualitative_model(copy,In,Out)
        Runtime[policy,r,corrupted,i] := timeNow() - startTime
        Xplicable[policy,r,corrupted,i] :=
          |Xplained[5]|*100/|Out[5]|
      end
    end
    if corrupted=0 or corrupted=|E| then goto :nextE
  end
  :nextE # skip further random edge corruptions of 0 or E edges
end
end

```

Fig. 23. Experimental design for assessing linking options

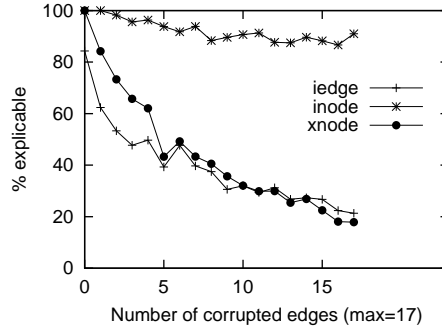


Fig. 24. Comparing implicit edge linking (IEDGE) with implicit node linking (INODE) with explicit node linking (XNODE). 2.6×10^6 theories were tested ranging from Figure 21 with 0 edges corrupted) to Figure 21 with all 17 edges were corrupted.

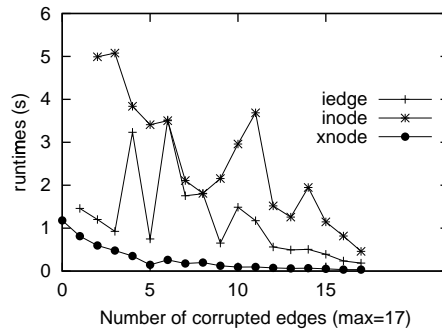


Fig. 25. Average runtimes in seconds for the experiments of Figure 24.

linked theory may be highly irregular because users could link only a small subset of nodes across time. Since we cannot demonstrate saturation for `XNODE` linking, then a simulation of (e.g.) 10^9 time ticks, the `XNODE` validator must search through a space equal in size to 10^9 copies of the theory. Hence, we can only recommend `XNODE` for short simulations such as the five time steps simulated in Figure 24.

Figure 24 shows us that `IEDGE`'s maximum explicable rate is 82%; i.e. `IEDGE` is not fully permissive. In our enthusiasm to cull the search space in a qualitative simulation, we have also culled some valid behaviours. Nevertheless, our results show that `IEDGE` can serve as a validation device for fisheries. Note that for `IEDGE` after only a third of the model being mutated, only around half the outputs are inexplicable. This is a nice result: we get clear, early indications if we are straying from a good model. Further, since we can prove that `IEDGE` supports saturation, then, unlike `XNODE`, we can show that `IEDGE` can be used for

validating very long simulation runs.

The average runtimes in seconds for each trial are shown in Figure 25. `IEDGE` and `XNODE` were much faster than the rejected linking policy, `INODE`. One interesting feature of Figure 25 is that as the models grow more corrupted, it becomes faster to determine what outputs are explicable. This is another nice result: we can reject nonsense faster.

VII. RELATED WORK

Menzies & Compton [24] have argued that the limits to validation are also the limits to modeling since we should not build what we cannot test. Levesque and Brachman observed that seemingly minor changes in a frame-based language dramatically altered the tractability of subsumption [25]. Here we have shown that seemingly minor changes in the temporal linking policy dramatically altered the tractability of validation. The implications for other representations are not yet clear. However, it is possible that if other representations are to support tractable validation, then must be constrained in a manner analogous to symmetric `IEDGE` linking.

Elsewhere [9, 26] we have extensively discussed the connection of this work to standard qualitative reasoning (QR); non-monotonic reasoning such as default logic and assumption-based truth maintenance systems (ATMS); and ATMS-based verification and validation tools from the V&V community. We include some summary notes below.

Default logic: An HT4 world is not an extension of a default logic theory [27]. Extensions are closed under deduction; i.e. they contain the attainable envisionments.

ATMS: As mentioned above, we generate different envisionments to the ATMS (§II-A). Further, while the ATMS computes its environments incrementally, HT4 is a batch process.

V&V: To the best of our knowledge, our work is the first detailed analysis of the computational complexity of validating theories used for time-based simulations. The validation of the runtime dynamic properties of rules has been studied by Preece *et. al.* [28]. Detailed complexity results for the verification of rule-based systems have been reported by Levy & Rousset [29]. Other ATMS-style validation approaches are all for non-temporal theories [30, 31]. None of these studies explore loops within the dependency graph of a rule based system. Such loops would require a temporal analysis.

Standard QR: Standard QR is based around theorem provers that process qualitative

differential equations; i.e. a piece-wise well-approximated low-order linear equation or a first-order non-linear differential equation whose numeric values are replaced by one of three qualitative states: up, down, or steady [32]. Examples of standard QR include QSIM [12] and QPT. Non-standard QR variants include first-order logic qualitative modeling [33], and non-linear QR [34]. Our approach is similar to standard QR since it is based on well-approximated low-order linear equations. However, it has certain key differences. For example, our symmetric edges are different to the $M+$ and $M-$ of QSIM. The QSIM connections allow for the automatic inference of landmark states where the sign of a variable's rate of changes from positive to negative (or visa versa). We do not permit the auto-deduction of such states: all our states must be encoded manually. Also, our language is optimizes for tractability, not expressability. Hence, we can't encode the higher-order constraints often discussed in the QSIM literature. In our defense, we note that often the QSIM literature uses those constraints to tame intractable chatter. We have no need to tame intractability since that comes for free with our language choice.

Another major difference to the standard QR literature is that we have sought a minimal graph-theoretic framework for our representation. As a result, the internal data structures of our approach are very uniform. This uniformity enables the kind of complexity analysis described above. Also, our approach can be viewed as a multiple-world qualitative version of time averaging [35]. Time averaging studies the long-term properties of a model under feedback. Such long-term reasoning is possible only for certain structures within a system of equations. Our technique, on the other hand, can discuss the long-term properties of all constructs in our language. To achieve this, we have had to adopt a more restrictive language than that used in (e.g.) QSIM. Despite these restrictions, we have shown in §VI that our language is still expressive enough to model and validate real-world QR theories such as Figure 21.

A fundamental property of qualitative systems is their indeterminacy. For example, recall that the qualitative model of Figure 4 could not tell if `companyProfits` would go up or down. In standard QR, one world is branched for each possibility: `companyProfits↑`, `companyProfits↓` and `companyProfits` remaining steady. When extended over several levels in a network, this can lead to an intractable branching of behaviour. Meta-knowledge

can be used to tame some of this indeterminacy. For example, the Waltz filtering of the QSIM [12] system ruled out a transition of the first derivative of a variable from increasing to decreasing without first going through a zero state. In practice, however, many possible behaviours will still be generated [36] and must be somehow handled by the program calling the qualitative simulator. Clancy and Kuipers observe that...

Intractable branching due to irrelevant distinctions is one of the major factors hindering the application of qualitative reasoning techniques to large real-world problems [37].

As of the time of this writing, the current best-proposal for taming intractable branching from the QSIM community is the DEC-SIM simulator in which the user divides the theory into several partitions [37]. These partitions are then simulated as separate units. While DEC-SIM has been able to offer richer simulations than basic QSIM, DEC-SIM’s authors comment that “DEC-SIM cannot guarantee a tractable simulation for any model.” By comparison, we can guarantee a tractable simulation for theories written using symmetric edges and IEDGE linking and whose non-temporal abductive validation task was tractable.

VIII. CONCLUSIONS

Representations are usually assessed via their expressibility, completeness, correctness, and tractability. Here we have explored a fifth criteria: testability. We have found that seemingly minor choices in the representation language can have a significant impact on the tractability of validation. In summary, we have explored a language comprising K -state variables connected by symmetric edges and joined across time by implicit edge linking. Using a graph-theoretic analysis, we have shown that such a language *saturates* in K time steps; i.e. if proofs do not terminate after K time steps, then they will never terminate. This observation can be used to dramatically cull the search space of qualitative simulations running for very long time periods. Without knowledge of saturation, if a theory is run for T_i time steps, then the size of the search space may be proportional to $O(T_i)$. However, given that language saturates, and if that simulation is only measured at T_j time points, then we can reduce that space to something much smaller ($\frac{(K*T_j)-1}{T_i}$).

Our approach requires that (i) non-temporal abduction of some theory is tractable; and (ii) when the non-temporal abductive validation problem is converted to a temporal

abductive validation problem, then it is linked using symmetric `IEDGE` edges. The resulting language is more restrictive than that used in standard qualitative reasoning such as `QSIM`. Despite these restrictions, we have shown in §VI that the language is still expressive enough to model and validate real-world QR theories such as the fisheries model. Further, we can make a clear statement about the tractability of validating all theories written in our restrictive language, where as other QR systems (e.g. `DECIM`) cannot.

ACKNOWLEDGEMENTS

Paul Compton (UNSW) first offered the intuition that we did not need to search every time copy of a theory since each copy just reproduces the search space of its predecessor. While this intuition proved not to be true in the general case, it inspired our search for special cases where it was true. Kingsley Jones (DSTO) first proposed the mutator used in §VI-A.

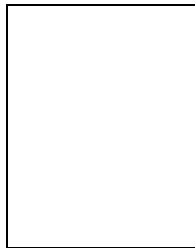
REFERENCES

- [1] R. Davis, H. Shrobe, and P. Szolovits, “What is a Knowledge Representation?,” *AI Magazine*, pp. 17–33, Spring 1993.
- [2] B.J. Wielinga, A.T. Schreiber, and J.A. Breuker, “KADS: a Modeling Approach to Knowledge Engineering,” *Knowledge Acquisition*, vol. 4, pp. 1–162, 1 1992.
- [3] N.M. Agnew, K.M. Ford, and P.J. Hayes, “Expertise in context: Personally constructed, socially elected, and reality-relevant?,” *International Journal of Expert Systems*, vol. 7, 1 1993.
- [4] B.G. Silverman, “Survey of expert critiquing systems: Practical and theoretical frontiers,” *Communications of the ACM*, vol. 35, pp. 106–127, 4 1992.
- [5] P. Compton, K. Horn, J.R. Quinlan, and L. Lazarus, “Maintaining an expert system,” in *Applications of Expert Systems*, J.R. Quinlan, Ed. 1989, pp. 366–385, Addison Wesley.
- [6] A.D. Preece and R. Shinghal, “Verifying knowledge bases by anomaly detection: An experience report,” in *ECAI '92*, 1992.
- [7] G.J. Myers, “A controlled experiment in program testing and code walkthroughs/inspections,” *Communications of the ACM*, vol. 21, pp. 760–768, 9, September 1977.
- [8] 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*, 1989, pp. 319–331.
- [9] T.J. Menzies and P. Compton, “Applications of abduction: Hypothesis testing of neuroendocrinological qualitative compartmental models,” *Artificial Intelligence in Medicine*, vol. 10, pp. 145–175, 1997, Available from <http://www.cse.unsw.edu.au/~timm/pub/docs/96aim.ps.gz>.
- [10] T. Bylander, D. Allemang, M.C. M.C. Tanner, and J.R. Josephson, “The Computational Complexity of Abduction,” *Artificial Intelligence*, vol. 49, pp. 25–60, 1991.

- [11] T.J. Menzies, "On the practicality of abductive validation," in *ECAI '96*, 1996, Available from <http://www.cse.unsw.edu.au/~timm/pub/docs/96abvalid.ps.gz>.
- [12] B. Kuipers, "Qualitative simulation," *Artificial Intelligence*, vol. 29, pp. 229–338, 1986.
- [13] B. Kuipers, "Qualitative simulation: then and now," *Artificial Intelligence*, vol. 59, pp. 133–140, 1993.
- [14] K. Eshghi, "A Tractable Class of Abductive Problems," in *IJCAI '93*, 1993, vol. 1, pp. 3–8.
- [15] T.J. Menzies, "Applications of abduction: Knowledge level modeling," *International Journal of Human Computer Studies*, vol. 45, pp. 305–355, 1996, Available from <http://www.cse.unsw.edu.au/~timm/pub/docs/96abkl1.ps.gz>.
- [16] J. DeKleer, "An Assumption-Based TMS," *Artificial Intelligence*, vol. 28, pp. 163–196, 1986.
- [17] H.N. Gabow, S.N. Maheshwari, and L. Osterweil, "On two problems in the generation of program test paths," *IEEE Trans. Software Engrg*, vol. SE-2, pp. 227–231, 1976.
- [18] G.A. Smythe, "Brain-hypothalamus, Pituitary and the Endocrine Pancreas," *The Endocrine Pancreas*, 1989.
- [19] R. Dieng, O. Corby, and S. Lapalut, "Acquisition and exploitation of gradual knowledge," *International Journal of Human-Computer Studies*, vol. 42, pp. 465–499, 1995.
- [20] T.J. Menzies and R.E. Cohen, "A graph-theoretic optimisation of temporal abductive validation," in *European Symposium on the Validation and Verification of Knowledge Based Systems, Leuven, Belgium*, 1997, Available from <http://www.cse.unsw.edu.au/~timm/pub/docs/97eurovav.ps.gz>.
- [21] S. Waugh, T.J. Menzies, and S. Goss, "Evaluating a qualitative reasoner," in *Advanced Topics in Artificial Intelligence: 10th Australian Joint Conference on AI*, Abdul Sattar, Ed. 1997, Springer-Verlag.
- [22] T.J. Menzies and S. Goss, "Applications of abduction #3: "black-box" to "gray-box" model," in *AI in Defence Workshop, Australian AI'95*, also *Technical Report TR95-31, Department of Software Development, Monash University*, 1995.
- [23] H. Bossel, *Modeling and Simulations*, A.K. Peters Ltd, 1994, ISBN 1-56881-033-4.
- [24] 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.
- [25] R.J. Brachman and H.J. Levesque, "The tractability of subsumption in frame-based description languages," in *AAAI '84*, 1984, pp. 34–37.
- [26] T.J. Menzies, *Principles for Generalised Testing of Knowledge Bases*, Ph.D. thesis, University of New South Wales. Available from <http://www.cse.unsw.edu.au/~timm/pub/docs/95thesis.ps.gz>, 1995.
- [27] R. Reiter, "A Logic for Default Reasoning," *Artificial Intelligence*, vol. 13, pp. 81–132, 1980.
- [28] A. D. Preece, C. Grossner, and T. Radhakrishnan, "Validating dynamic properties of rule based systems," *Int. J. Human-Computer Studies*, vol. 44, pp. 145–169, 1996.
- [29] A.Y. Levy and M. Rousset, "Verification of knowledge bases using containment checking," in *AAAI '96*, 1996.
- [30] N. Zlatereva, "Truth maintenance systems and their application for verifying expert system knowledge bases," *Artificial Intelligence Review*, vol. 6, 1992.
- [31] A. Ginsberg, "A new Approach to Checking Knowledge Bases for Inconsistency and Redundancy," in *Proc. 3rd Annual Expert Systems in Government Conference*, 1987, pp. 102–111.
- [32] Y. Iwasaki, "Qualitative physics," in *The Handbook of Artificial Intelligence*, P.R. Cohen A. Barr and E.A. Feigenbaum, Eds., vol. 4, pp. 323–413. Addison Wesley, 1989.

- [33] I. Bratko, I. Mozetic, and N. Lavrac, *KARDIO: a Study in Deep and Qualitative Knowledge for Expert Systems*, MIT Press, 1989.
- [34] K.M. Yip, “Understanding complex dynamics by visual and symbolic reasoning,” *Artificial Intelligence*, vol. 51, pp. 179–221, 1991.
- [35] R. Levins and C.J. Puccia, *Qualitative Modeling of Complex Systems: An Introduction to Loop Analysis and Time Averaging*, Harvard University Press, Cambridge, Mass., 1985.
- [36] P. Fouche and B. Kuipers, “An assessment of current qualitative simulation techniques,” in *Recent Advances in Qualitative Physics*, B. Faltings and P. Struss, Eds. 1992, pp. 263–278, The MIT Press.
- [37] D.J. Clancy and B.K. Kuipers, “Model decomposition and simulation: A component based qualitative simulation algorithm,” in *AAAI-97*, 1997.

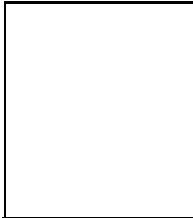
seen below



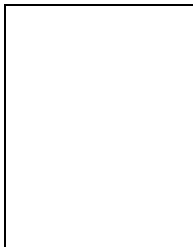
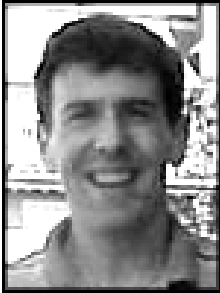
Tim Menzies is an Assistant Professorship at the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada. In his research, he explores how quirks in human cognition effect the process of software and knowledge engineering. He holds a Ph.D. in artificial intelligence (1995), a masters of cognitive science (1988) and a computer science undergrad degree (1985), all from the University of New South Wales, Sydney, Australia. In his career, Dr. Menzies has worked with NASA on software testing and with commercial organizations on object-oriented systems and expert systems. Dr Menzies is

the author of over 90 journal/conference/ and workshop articles (see <http://tim.menzies.com/papers.html>).

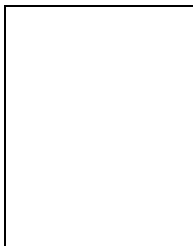




Robert F. Cohen Robert Cohen is President of Algomagic Technologies, Inc, an internet security company, and a software consultant. He holds a PhD and M.S. in Computer Science from Brown University, M.S. in Computer Science from Boston University, and B.A. in Mathematics from Brandeis University. He lives in Boston, Massachusetts.



Sam Waugh Sam Waugh received the BSc and PhD degrees in Computer Science from the University of Tasmania, Australia in 1991 and 1997, respectively. He is currently a Research Scientist with the Air Operations Division of the Defence Science and Technology Organisation, Melbourne, Australia. His research interests include machine learning and software engineering.



Simon Goss Simon Goss holds BSc(Hons) (1978) and PhD (1985) degrees in physical chemistry from La Trobe University, and a Grad Dip KBS (1991) from RMIT. He is a Senior Research Scientist in Air Operations Division of DSTO. He has been local chair of the AI SIG of the ACS since 1991 and chaired national conferences in Cognitive Science and international workshops in modeling Situation Awareness. In previous existences he has worked as a programmer-engineer in industrial control, a market-driven consultant, a physical scientist and a journalist. He is a member of IEEE, AAAI, ACM, ACS and AORS. His research interests lie in applied AI in the context of representation, software life cycle, and optimisation, in modeling and simulation of complex social systems for operations research.

