Lower Bounds on the Size of Test Data Sets

Tim Menzies[1], Sam Waugh[2]

[1] Artificial Intelligence Department, School of Computer Science and Engineering
University of NSW, Australia, 2052; timm@cse.unsw.edu.au
[2] Defence Science and Technology Organisation, Air Operations Division, Melbourne,
Australia, 3001; sam.waugh@dsto.defence.gov.au

Abstract. Practically speaking, how small can a test suite be and still be of value? In the context of temporal graph-theoretic abductive validation, the answer to this question is very language-dependent. Seemingly trivial variations in a language can have a significant impact on how large a test suite must be. This paper is hence a cautionary note to those who invent languages and ontologies without experimentally testing the practicality of those languages.

1 Introduction

Validating a theory is hard work. Validation is complicated even further in poorly-measured domains. In such domains, the cost of data collection prevents us collecting all the observations we desire. Such observations could be used to validate that a theory of X can reproduce known behaviour of X. Such observations are lacking in many domains; e.g. economics and neuroendocrinology. The (in)famous *Limits to Growth* study attempted to predict the international effects of continued economic growth [7]. Less than 0.1 percent of the data required for the theories was available [3]. Data collections in neuroendocrinology can be just as sparse since data collection in that domain is very expensive. In one extreme example, 300,000 sheeps brains had to be filtered to extract 1.0 milligrams of purified thyroptin-releasing hormone [6].

Techniques exist for automatically generating test data sets. For example, the dependency network of a system can be used to determine inputs that will exercise all branches of the system. Sophisticated non-monotonic techniques can be used to separate inputs into sensible subsets [5,18]. However, note that once an input suite is inferred, an expert still has to decide what are the appropriate outputs for those inputs. This may be a significant analysis task and, in practice, may only be practical for small systems.

For all the above reasons, it is difficult and expensive to find or build test data sets containing valid pairs of inputs and outputs. Given this, practioners often need to rationalise the process of building test data sets. The value of building bigger test data sets must be weighed up against the cost of their construction. To avoid wasting money, practioners must build test data sets big enough to be useful, but no bigger.

This paper explores *how big is big enough?*; i.e. practically speaking, what are reasonable lower bounds on the size of a test suite. Size will be expressed as the percentage of variables in a theory which are not measured in the test data set. The lower bound on size will be found as follows. Assuming our temporal graph-theoretic abductive validation procedure [8,11,12,17], we will reduce test suite size until we can no longer distinguish good theories from bad theories. It will be found that seemingly minor variations in a language can have a significant impact on this lower bound on test suite size. This paper is hence a cautionary note to those who invent languages and ontologies without experimentally testing the practicality of those languages.

This article is structured as follows. Before we describe our experiments, the abductive validation framework is explained: first the kind of theories it process; next the details of abductive validation. Four language variants of a temporal extension to this validation procedure are defined: XNODE, INODE, XEDGE, IEDGE. An experiment is described in which theories written in these four variants are executed using fewer and fewer measurements. The lower practical bound on the size of the test data sets will be found to be crucially dependent on minor variations in the language used. For example, XNODE is only a small variant on the IEDGE language. However, XNODE is practical down to 70 percent unmeasured while IEDGE fails after 40 percent unmeasured.

The theoretical framework of this article has been presented before (e.g. [8, 11,12,17]). The new contribution of this article is the data reduction experiments and the observation that languages react very differently to data reduction.

2 Theories

This section describes the types of theories used in this analysis. The next section describes a validation procedure which executes over these theories.

This analysis assumes that theories fall into the following framework:

- Theories contain a finite number of variables, each with a finite number of N mutually exclusive states.
- Theories are written in some language L and a language-specific translator can convert theories into a directed dependency graph connecting and/or-nodes. Loops in this depenency graph may implies extra edges from comparisons at time I to time J.
- To use and-nodes in a proof, all the parents of that node must also appear in that proof. To use or-nodes in a proof, only one of the parents need appear in that proof.
- Internally, or-nodes are time-stamped comparisons. Such comparisons record our belief that a variable at some time has some state. For example, *age@0 above 10* says that at time 0, we believe that *age* is over 10.
- Certain pairs of comparisons are illegal; e.g. *age@0 above 10* contradicts *age@0 below 8*.

- A test data set documenting required behaviour is available. This data set comprises pairs of inputs and outputs where each input or output is a comparison.
- Testing is a validation process which searches for consistent pathways from inputs to outputs across the dependency connections (see next section).

Many symbol-level knowledge bases can be expressed in the above form. The dependency graph of a propositional rule base used in a match-select-act cycle can be mapped into the above structure (literals in rule left-hand-sides at time *I* can be connected to right-hand-side literals at time *I+1*). Qualitative equations can also be expressed in this form. For example, *A=3B*2C-5D* would generate influences from right-hand-side variables to the left-hand-side-variables (and also for all valid rearrangements of the equation such as *D=(3B*2C-A)/5*). Most generally, any horn clause that can be partially evaluated to a ground state would satisfy the above description (subgoals connect via an and-node to the head).

Parts of knowledge-level models can also be expressed in the above form. If a problem solving method ever needs to access variables near some variable of interest (e.g. in fault-localisation during model-based diagnosis), then that problem-solving method would be traversing the dependency network described above. In the special case where a KADS knowledge source is expressed in a rule base, then the whole knowledge-level model can be reduced to such a dependency graph. Elsewhere, Menzies and Mahidadia [13] argue that many problem solving methods can be modelled as choice operators controlling the traversal of the above dependency network (a proposal similar to that implemented in SOAR [14]).

3 Temporal Graph-Theoretic Abductive Validation

3.1 Graph-Theoretic Abductive Validation

Abduction is a demonstration that a theory, plus some assumptions, can reach some goal without causing contradictions. If contradictions can occur, abduction must create *worlds*: maximal consistent sets of beliefs. If multiple such worlds can be generated, then a BEST assessment operator selects the preferred world(s). For more details on abduction, see [2, 10]. For an example of abduction, see below.

Graph-theoretic abduction implements abduction using the above dependency network. Consistent pathways (ordered sets of edges with no illegal pairs of comparisons) are found between output goals back to known inputs. Pathways that cross unmeasured variables must make assumptions. Worlds are generated by collecting maximal subsets of these pathways with compatible assumptions.

Graph-theoretic abductive validation uses a BEST operator that returns the worlds with the largest number of outputs. Intuitively, this procedure is searching for the assumptions that let us explain the most of our known behaviour. As an example, let us search a theory of economics for the assumptions that let us

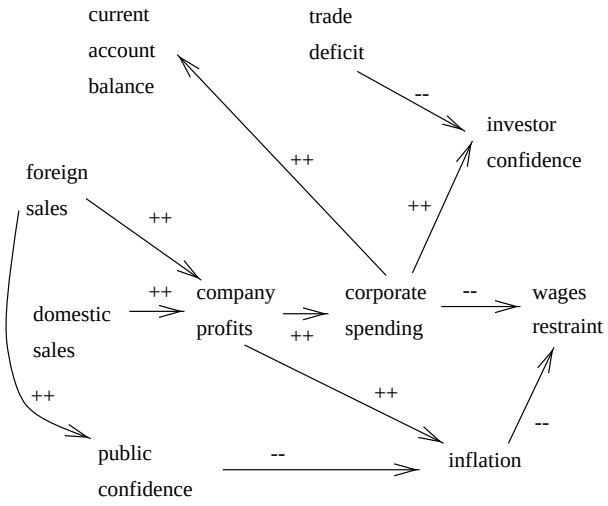

Fig. 1. An economics theory.

explain the most number of outputs (We shall use the theory in Figure 1.) In the language of that theory, variables have three states: *up*, *down* or *steady*. These values model the sign of the first derivative of these variables and model the rate of change in each value. Dependencies between them can be created as follows. The direct connection between *foreignSales* and *companyProfits* (denoted with plus signs) means that *companyProfits* being *up* or *down* should be connected back to *foreignSales* being *up* or *down* respectively. The inverse connection between *publicConfidence* and *inflation* (denoted with minus signs) means that *inflation* being *up* or *down* should be connected back to *publicConfidence* being *down* or *up* respectively. Also, in this language, competing upstream influences can cancel out to explain a steady. There are two upstream influences to *companyProfits*. Dependencies are created from *companyProfits=steady* back to an and-node with parents (e.g.) *foreignSales=up* and *domesticSales=down*.

In the case where the inputs are *foreignSales=up, domesticSales=down* and the outputs are *investorConfidence=up, inflation=down, wageRestraint=up*, there are six pathways connecting inputs to outputs;

- PATH1: foreignSales=up, companyProfits=up, corporateSpending=up, investorConfidence=up
- PATH2: domesticSales=down, companyProfits=down, corporateSpending=down wageRestraint=up
- PATH3: domesticSales=down, companyProfits=down,inflation=down
- PATH4: domesticSales=down, companyProfits=down, inflation=down, wagesRestraint=up
- PATH5: foreignSales=up, publicConfidence=up, inflation=down
- PATH6: foreignSales=up, publicConfidence=up, inflation=down, wageRestraint=up

Note that these paths contain contradictory assumptions; e.g. *companyProfits=up* in PATH1 and *companyProfits=down* in PATH2. If we sort these paths into the biggest compatible subsets, we get the worlds shown in Figure 2 and Figure 3. Applying our BEST criteria, we see that the world that assumes

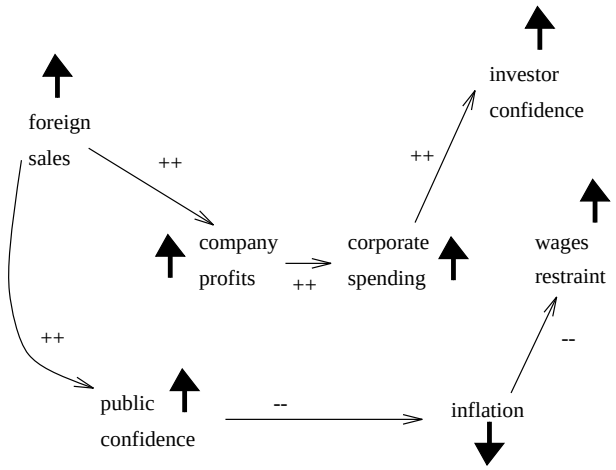

Fig. 2. A set of consistent pathways: PATH1 and PATH5 and PATH6.

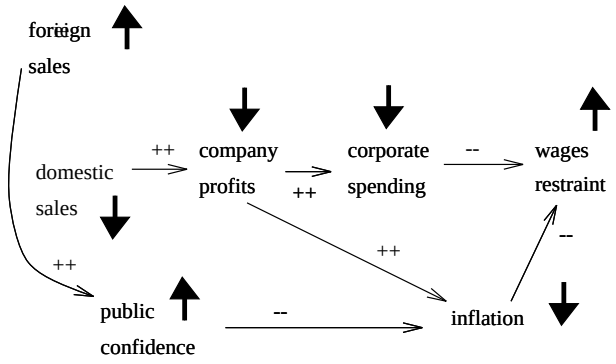

Fig. 3. Another set of consistent pathways: PATH2 and PATH3 and PATH5 and PATH6.

companyProfits=up lets us explain all of our outputs. That is, this theory has passed the abductive validation test. This process has found numerous inexplicable outputs in theories of neuroendocrinology published in international refereed journals [4, 8, 12]. The diagrams from those papers were expressed in the qualitative language of our economics example. Interestingly, the faults were found

using data taken from the papers that proposed those theories. Also, the faults had never been detected before, even by the reviewers of those journals. Further, when experts reviewed the detected faults, they found them exciting and insightful to their domain [16].

3.2 Temporal Graph-Theoretic Abductive Validation

The above example had no feedback loops. In theories with feedback loops, it is possible that a literal can be assigned multiple values over the life time of the simulation. To handle time, we add a time stamp to the definition of a literal; e.g. *population* could be renamed to *population@1, population@2 ... population@T* where T is some time point.

How are we to connect literals at time I to literals at time J? Depending on how we answer this question, we can define variants on a qualitative simulation language. Consider the theory containing two edges: *direct(A,B)* and *inverse(B,A)*. If we execute this theory over three time steps, we could search one of the spaces illustrated in Figure 4. In the *implicit node linking* language

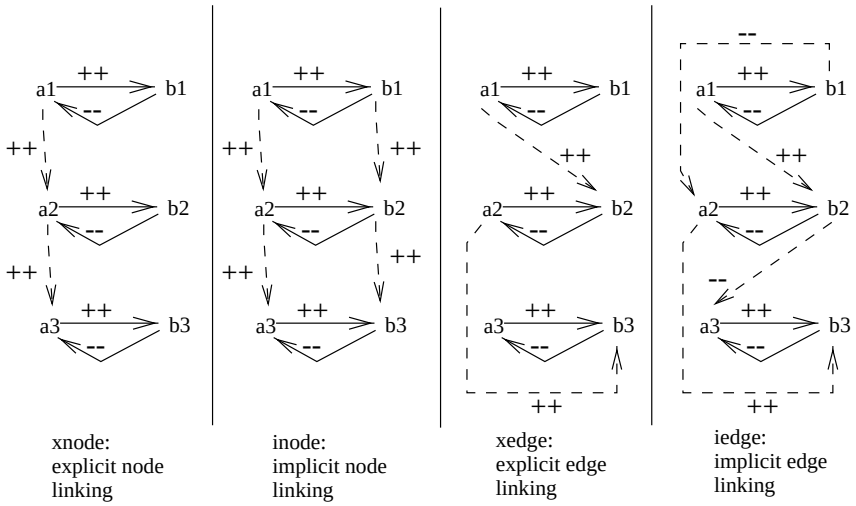

Fig. 4. *Direct(A,B)* and *inverse(B,A)* renamed over 3 time intervals using different time linking policies. Dashed lines indicate time traversal edges.

(or INODE), we cross time on every node; i.e. every comparison at time I is connected to the same comparison at time *I+1*. In the *implicit edge linking* language (or IEDGE), we cross time on every edge. 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 *explicit edge linking* language (or XEDGE), we only cross time on the edges explicitly denoted as time edges by the user (in this case, the direct link from A to B).

Once the search space has been defined, it can be compiled into the dependency graphs and tested using graph-theoretic abductive validation.

4 Limits to Temporal Graph-Theoretic Abductive Validation

In the context of the above architecture, what can we say about how much test data is enough? To answer this question, we need one more definition. We say that a validation device is adequate if it can distinguish good theories from bad theories. If we can identify precisely the point where we lose adequacy, then we have found a limit to that validation device. To operationalise this theory in the context of *how much data is enough*, we need to validate good and bad theories using different sizes of test data sets. This section describes how this can be done. First, a range of theories must be generated ranging from good theories to bad theories. Next, input-output sets must be generated. Lastly, validation must be attempted with more and more of the theory unmeasured.

4.1 Generating Theories

To generate theories, we started with the quantitative equations of a fisheries system using equations from [1] (pages 135-141). Next, we built a qualitative form of the fisheries model as shown in Figure 5. Note that this fisheries model

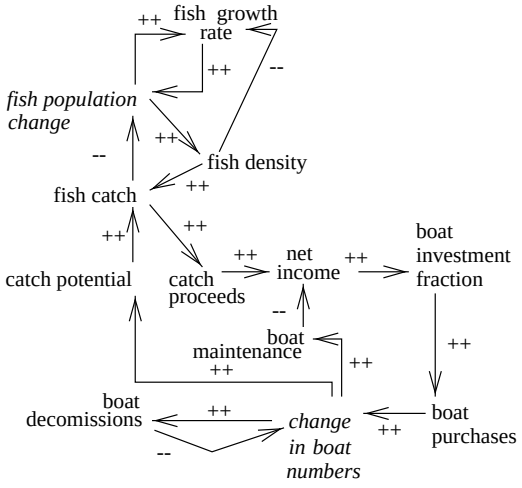

Fig. 5. The fisheries model. Adapted from [1] (pp135-141). Variables in italics are the *first derivative variables* used in the XNODE and XEDGE study.

is ambiguous concerning how to handle time. We must add in a temporal causal

interpretation (e.g. XNODE, XEDGE, INODE, or IEDGE) in order to handle the feedback loops. When using XNODE or XEDGE, we must somehow assign our explicit time traversal nodes. XNODE used the *first derivative variables* which show time rate of change. Example first derivative variables in the fisheries models are *fish population change* and *change in boat numbers*. XEDGE used the edges leaving a first derivative variable.

To generate a range of theories, we corrupted some portion of the qualitative version of the fisheries model. Fisheries has 17 edges. These edges were corrupted as follows. The annotations of between 0 to 17 edges edges in fisheries were flipped (inverse to direct, or visa versa), chosen at random. Once the model was mutated, it was then copied over several time steps and connected via one of the XNODE, XEDGE, INODE, or IEDGE temporal linking policies.

Note that as the number of edges mutated increases from 0 to 17, the mutated model becomes less and less like the original model. That is: at *mutations=0* we are processing the correct fisheries model; at *mutations=17* we are processing a very incorrect fisheries model; at *mutations=2..16* we are processing progressively worse fisheries models.

4.2 Generating Data

To generate data, we ran the quantitative fisheries model 15 times. Next, we generated input and output *up*s and *down*s by comparing all pairs of the measurements in the 15 runs (105 such pairs exist). Inputs were always observations found in the first copy of the model. Outputs were always observations not found in this first copy.

Once we had the data, we threw some of it away. U percent of the output from was discarded to produce 10 variants of the data with U at 0, 10, 20, 30, 40, 50, 60, 70, 80, and 90 percent unmeasured.

4.3 Validation Results

This section shows the results from graph-theoretic abductive validation running with different test suite sizes. Test suite size was expressed as what percentage of the variables in the fishing were measured and included in the output set. Between 0 to 17 edges were corrupted 20 times to create 360 new models. This process was repeated 10 times each time U was increased (i.e. resulting in 3600 models). These were exercised for both XNODE, XEDGE, IEDGE, INODE using the qualitative 105 data sets created above; i.e. 3600*105*4=1,512,000 runs. The results are shown in Figure 6 and Figure 7.

To read these results, note that the x-axis shows a progression from a correct fishery model (at 0 edges corrupted) to a very incorrect fisheries model (at 17 edges corrupted). The y-axis shows the percent explicable found by graph-theoretic abductive validation. In order to distinguish a good model from a bad model, this x-y plot should start high on the y-axis, and fall off to a low figure on the right-hand-side of the x-axis; e.g. the *U=0* plot for XNODE. If the x-y

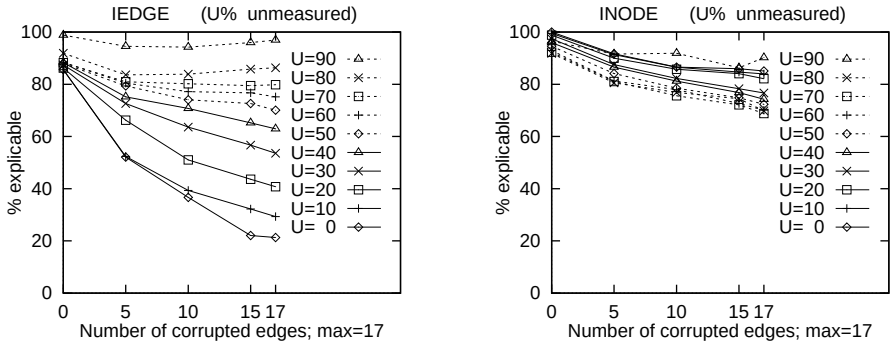

Fig. 6. Implicit linking: corrupting 0 to 17 edges, running validation with less and less data.

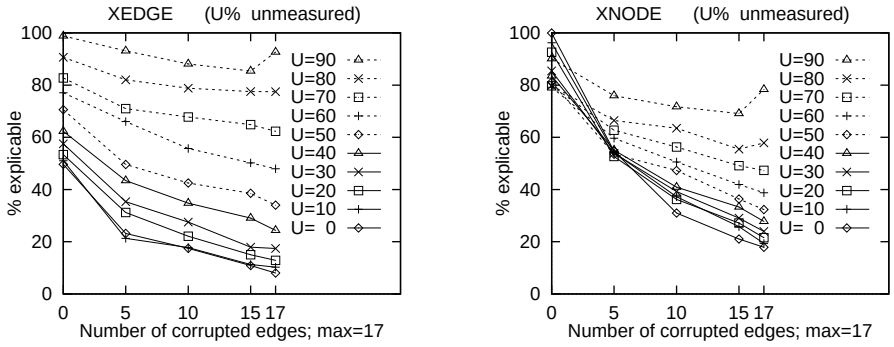

Fig. 7. Explicit linking: Corrupting 0 to 17 edges, running validation with less and less data.

plot remains flat, then the validation procedure is given the same score to good models and bad models; e.g. the *U=40* plot for IEDGE.

Several effects can be read from these graphs:

- In all cases, as the percentage unmeasured in the theory increased, the x-y plots flattened out. That is, with less and less data, it becomes harder and harder to distinguish a good model from a bad model.
- Consider the *U=0* case. As reported previously [17], some time linking policies are clearly inferior. XEDGE can explain, at best, only half the data. INODE can barely distinguish good from bad models: lowest explicable percentage is 88 percent. XEDGE and INODE are poor candidates for a validation system in this framework.
- The remaining adequate languages (XNODE and IEDGE) react differently to increasing the percentage unmeasured. IEDGE is not recommended above *U=40*. However, XNODE is adequate for distinguishing good from bad models down to *U=70*.

5 Discussion

Seemingly trivial variants to a language can have a major impact on the minimum practical size of a test suite. The difference between the definition of INODE and XNODE is very small. Yet experimentally it has been shown here that INODE is an impractical validation language while XNODE lets us use only very small test data sets (up to 70 percent unmeasured).

How general are these results? This analysis assumes graph-theoretic abductive validation. It was argued above that this process is relevant to the validation of a range of symbol-level and knowledge-level models. The results above come only from the fisheries model. However, in this study, nearly 400,000 mutations of that model were generated and validated. Furthermore, the methodology used here for testing limits to validation is quite general. After defining the core parameters of your validation procedure and theories, build numerous variants of the procedure and theories and look for the cases where the validation engine fails. (This approach to experimentally exploring an algorithm is analogous to that used in the satisfiability community [15].) Menzies has argued elsewhere [9] that the knowledge engineering field is in urgent need of such empirical evaluations. For example, this paper is a cautionary note to those who invent languages and ontologies without experimentally testing the practicality of those languages. Ontologies define a language and the choice of language can have significant implications; e.g. the minimum amount of data needed to validate a theory written in that language.

References

1. H. Bossel. *Modeling and Simulations.* A.K. Peters Ltd, 1994. ISBN 1-56881-033-4.
2. T. Bylander, D. Allemang, M.C. M.C. Tanner, and J.R. Josephson. The Computational Complexity of Abduction. *Artificial Intelligence*, 49:25–60, 1991.

3. H. S. Coles. *Thinking About the Future: A Critique of the Limits to Growth.* Sussex University Press, 1974.
4. B. Feldman, P. Compton, and G. Smythe. Towards Hypothesis Testing: JUSTIN, Prototype System Using Justification in Context. In *Proceedings of the Joint Australian Conference on Artificial Intelligence, AI '89*, pages 319–331, 1989.
5. A. Ginsberg. A new Approach to Checking Knowledge Bases for Inconsistency and Redundancy. In *Proc. 3rd Annual Expert Systems in Government Conference*, pages 102–111, 1987.
6. D.T. Krieger. The Hypothalmus and Neuroendocrinology. In D.T. Krieger and J.C. Hughes, editors, *Neuroendocrinology*, pages 3–122. Sinauer Associates, Inc., 1980.
7. D.H. Meadows, D.L. Meadows, J. Randers, and W.W. Behrens. *The Limits to Growth.* Potomac Associates, 1972.
8. 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`.
9. T.J. Menzies. Evaluation Issues for Problem Solving Methods, 1998. Banff KA workshop, 1998. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/97eval`.
10. T.J. Menzies. Applications of Abduction: Knowledge Level Modeling. *International Journal of Human Computer Studies*, 45:305–355, September, 1996. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/96abkl1.ps.gz`.
11. 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`.
12. T.J. Menzies and P. Compton. Applications of Abduction: Hypothesis Testing of Neuroendocrinological Qualitative Compartmental Models. *Artificial Intelligence in Medicine*, 10:145–175, 1997. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/96aim.ps.gz`.
13. T.J. Menzies and A. Mahidadia. Ripple-Down Rationality: A Framework for Maintaining PSMs. In *Workshop on Problem-Solving Methods for Knowledge-based Systems, IJCAI '97, August 23.*, 1997. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/97rdra.ps.gz`.
14. P.S. Rosenbloom, J.E. Laird, and A. Newell. *The SOAR Papers.* The MIT Press, 1993.
15. B. Smith. Locating the Phase Transition in Binary Constraint Satisfaction Problems. *Artificial Intelligence*, 1996.
16. G.A. Smythe. Concerning errors in the Smythe '87 model., 1992. Personal communication.
17. S. Waugh, T.J. Menzies, and S. Goss. Evaluating a Qualitative Reasoner. In Abdul Sattar, editor, *Advanced Topics in Artificial Intelligence: 10th Australian Joint Conference on AI.* Springer-Verlag, 1997.
18. N. Zlatereva. Truth Mainteance Systems and Their Application for Verifying Expert System Knowledge Bases. *Artificial Intelligence Review*, 6, 1992.