

AGENTS AND LIGHTWEIGHT USE OF LOGIC, COMBINED SIMULATION LOGIC AND NEURAL AGENT NODES

Kevin D. Reilly
Computer & Information Science
Behavioral Neuroscience – Psychology
Civitan International Research Center
University of Alabama at Birmingham
reilly@cis.uab.edu

KEYWORDS

agents, logic programming formalism, lightweight use of logic, neural networks, psychology and decisions

ABSTRACT

Promoting research on agents, particularly in modeling and simulation contexts, is a part of this paper's purpose. Particular promotions are sought with respect to characterizing the notion and the role of formal reasoning. After definitional material, we present a formally defined notion of a programming system or style to support "symbolic simulation," designed, e.g., to join numerical simulation in a combined or hybrid system. The presentation is related to a movement in "lightweight use of logic in conceptual modeling." Abstractions that emerge in these discussions are viewed in agent supporting contexts. Discourse moves to decision making (using a table format) and another role for logic programming. Properties emerging in this context are mentioned, followed by discussion of agent style migration of the logic problem description to a "neural network" node (a creating and processing node). Property preservation and NN enhancements precede a final return from the NN node of data and suggested processing to the (originating) logic node.

INTRODUCTION

In previous papers (Reilly et al., 1995; Barrett et al., 1995) we thought it helpful to work on the *definition* of agent. It is not so important that a firm definition emerge; what is important is to address the richness in candidate properties put forward for agents (by advocates and practitioners) and relating them to (other) styles of (distributed) processing, e.g., "autonomous objects" (Morse et al., 1997). The meeting of any given study's claims with core and extra-core features in "thought out" definitions is part of the issue as are future orientations, in a context where we expect a greater increase in agent schemes.

AGENT

We argued that *three* definitional schemes are worthy of consideration, two of which are empirically (practice) based and one theoretically based (Reilly et

al., 2001). (The latter can readily be expanded, but it is outside our present scope and currently, according to Sowa (2000) not making inroads (as he might have it).

Empirical Approach I The first case among the empirical schemes is the most popular. Typically, a single study (or a few agent espousing studies) is (are) performed and a definition presented. In cases when only one study establishes the definition, it has been widely noted that the definition is often distinctly biased to the study. (And, in some of these cases, it is even hard to recognize just what the agent connection is!) When authors have performed several studies and are positioned to be well aware of other work, such as reflected in (Wooldridge and Jennings, 1995), one of whose authors is a guru in the field, a more refined definition emerges. Franklin & Gaesser (1996) is perhaps in this same context. The problem is, however erudite, the core properties do not seem to coincide as well as we like. Trying to reconcile them, however, may be profitable (and interesting).

Wooldridge and Jennings' (1995) core is: autonomous social reactive and proactive. Franklin & Gaesser (1996) present: an agent is environmentally situated; senses its environment; acts on it, effecting future sensing; operates over time, and is dependent on an agenda of its own. The latter definition seems not to include the social aspect and autonomy is a corollary. Similarly, though "having its own agenda" smacks of "proactive," it is not clear that these terms are co-extensive. Mobility, often thought of because of "KnowBot stories," is not in the core of either of these definitions, as the autonomous object pursuers point out (Morse et al., 1997). Franklin and Gaesser are interested in carrying their definitions into building taxonomies of agents.

Empirical Approach II The dilemma that the "experts" can't agree suggests search for other approaches. Not previously done, to our knowledge, is analysis based on existing agent-claiming work, sifting through a relatively large number of studies and gathering the attributes, without (and perhaps with) prejudice. The final product, a matrix of *study vs. attribute*, can be submitted to multivariate statistical analyses to determine principal or core elements.

Sowa's relatively long list (Sowa, 2000) augmented by some of the above and other sources,

allows us to offer a sample of what this may entail: autonomous goal-oriented collaborative flexible self-starting temporal continuity character communicative adaptive mobile veracity benevolence rational intentional desire obligation commitment deliberation flexibility selectivity and robust. This includes both "is" and "has" attributes. This method suffers from (still) having a synonym problem (refer to the co-extensive property noted above), but it might be easier to work around through approaches which take varying degrees of conservative to liberal handling of putative synonyms (i.e. the "with prejudice" just mentioned).

Theoretical Approaches The third, theoretical, approach typically starts from formal logic. Such a logic may well be modal. For example, a line of work relating to planning has emerged, e.g., (Cohen and Levesque, 1990)), formalizing "rational agents" which *plan*, incorporating a Belief-Desires-Intentions (BDI) plan scheme due to (Bratman, 1987).

Lightweight use of logic (LWUL) approaches (Robinson and Agusti, 1999) may represent a way to providing schemes which meet most needs. In this kind of view, we may formalize only a portion of a system and use it toward flexible ends, e.g., seeking means to handle co-extensiveness in agent definitions' attributes. For example, relating proactivity, self-starting, and having its own agenda could be a target. The discussion immediately following provides a view of some work which we view as a LWUL application.

Symbolic Simulation Formalism In Reilly et al., 1995 and Barrett et al., 1995), we sought to formally characterize a combined continuous, discrete and symbolic simulation language (or system or, even a style). Since the analysis, as we'll see, progressively departs from strident formalism to loose formulations, we dub it a lightweight use of logic (LWUL).

"Heavy weight" features stand out in the symbolic portion, at its base. A definition of "symbolic" simulation was a hurdle with (previous paper) reviewers; the formalism takes a step toward jumping this hurdle. It provides a core simulation system and presents processing styles on which to build.

Formalism "weakening" occurs as we progress from rigorously defined constructs to ones "piggy backed" on them, still rigorous if no "stray entries" creep in. Strays are a very real problem in this context since the formalism is programmed on machine. Technically, the formalism is an operational semantics based on an abstract machine (defined in logic programs). It is used practically, however, where supplementing is used with these supplements perhaps working their way "backwards" in less than fully careful work. Admittedly undefined constructs, interpreted as "yet to be defined" are not seen as "problematic" outside this potential debility.

Heavyweight Start The basic idea is to define a core of commands, "machine" instructions for an abstract machine. An operational semantics is defined for this machine running on a Prolog definitional scheme due to Moss (1981). These lower level instructions, once defined, become the basis for several higher level constructions, which are also well defined. Putatively, this latter step is "heavy weight" but it is to be understood that the higher level construction's formalization is once removed from the abstract machine, introducing potential difficulty, e.g., creating a need for a compiler and its potential intrusions. We nevertheless call this portion of our scheme, Barrel-F, "F" for "formal." It contrasts with Barrel-E, "E" for "exploratory."

Becoming LW The situation becomes increasingly lightweight from this point on, in part because we make judgments, without providing proofs, that new constructs can readily be defined --- and *even* that some need *not* be defined, because it is "obvious" that it would be easy to do so! The set of higher-level instructions formally defined was not sufficient (or at least not entirely convenient) for some of the practical applications of Barrel-F.

Using a system, which mixes 1) fully formally defined constructions, 2) ones "once removed" but based on the core and worked out on the abstract machine, and 3) others, alleged to be consistent with the latter, ultimately constitutes a species of LWUL. Such an exercise, nevertheless, proves useful in defining (a notion of) symbolic computing that can be related to others' claims on the same topic and can thus help in the definitional work.

Finally, the discrete and continuous system portions of the overall system were based on a more ad hoc formalism (Hooper and Reilly, 1982 and Hooper and Reilly, 1983). LW in flavor in that arguments were confined to only a portion of a combined discrete and continuous simulator (putatively, the most important part, the "strategy"). The advisor on the project advocated a formal approach that would be totally consistent with the work we present in the next section of this paper, i.e., a decision table theoretic rendition and embedded in logic programming. This part of the work did not materialize, in part because it introduced some awkward features, but perhaps, more so, because the formulation used clung more tightly, and thus more obviously, to its predecessor work (due to Ziegler, in his theory book (Zeigler, 1976) or a related paper).

Note that this foray into formalism and the one in the paragraphs above are "theory and practice" efforts. Theory is deemed a worthy matter principally for characterizing and guiding a practical work. (The studies, of course, could be starting points for further theorizing, but we have not elected to press this angle.) The simulators, which arose in practice from

this base, form a combined continuous, discrete and symbolic simulator.

Agent Redux In our paper (Reilly et al., 2001) we cited that some of the confusion about agents stems from confusion about Software Engineering lifecycle stages. A distinction between specifications and implementation is helpful. Some authors define agents relative to specifications whereas others, to implementation details or implementation styles. A statement to the effect that agents should not be restricted to software agents is cited as an example; we noted that this possibly betrays an inordinate orientation to implementation (at a deficit for the more important realm of specification) and, for example, introduces a bias against the notion that some agent scheme (specification) might well be implemented by one group in software and in firmware and/or hardware by another group.

The above formulation seeks to avoid this kind of limitation. It is important to note that a system may be specified in agent terms, and yet, due to limitations in today's machines, be implemented with several "concessions" to non-agent elements. The opposite often occurs when an agent "style" is imposed on a problem that really may not naturally be agent-like. (How many such systems are there --- given that humans are so very agent-like in life --- can they devise systems which are "foreign" to their thinking?)

The former (concessions) case has analogies in neural networks, where, e.g., a NN that calls for an extremum may well use a conventional algorithm, though extrema calculations can in principle be done in NN-ware. The issue is purely one of efficiency in a particular software environment.

A final note is that as the system we defined "moves" from totally formal to less so, it leaves in its wake a series of model abstractions. These can form a basis for agency schemes, which include, e.g., among their premises, a need to obtain answers from systems with rigorous foundations (perhaps along with respecting less formally derived answers on any of several grounds, e.g., spontaneity?).

CASE STUDY

In this case study we approach a problem of model abstraction and model enhancement attained in a multiple model situation. We proceed by example.

Decision Tables The study starts from the decision table based on Montalbano (1974) (see Figure 1). This kind of table is called extended entry. It can be reformulated so that the upper right and lower right quadrants have entries of Y or N or - (for "Don't care"); such a table is called limited entry. In practice, extended entries are considered more user friendly and limited more machinable. Additional

merits can be cited in relation to storage, movement over a network, and so on.

Car Make		reo		reo		cord		cord		dues'		dues'	
Car Cond'n		good		poor		good		poor		good		poor	
Commission		5%		15%		5%		10%		7.5%		17.5%	
ShopWork		2hrs		2days		2hrs		1day		4hrs		2days	
Manager OK		no		yes		no		no		yes		yes	

Figure 1: Salesman Conditions-Actions

Reilly et al. (1987) and Salah and Reilly (1987) discuss logic programming representations of this kind of system and related ones. Included are 1) storage in relational databases, 2) a form of approximate reasoning addressing rule formulations in which rules fire if k of n antecedents are satisfied and 3) (multiple) representation schemes using both predicate and term representations (Kowalski, 1979). A key point relative to this final matter is that predicate representations of data cause the control program (the interpreter) to access the data so that the programmer need not write data access code; in a term (functional) representation, the programmer must write access code.

Logic Program These considerations have ramifications for agent style computing, and we shall meet one or more of them in the following pages. Our immediate concern will be a new tact, in which we port the information content of our table to a neural network, and, in consequence, achieve a broader and more powerful form of computation, useful say, in simulations like those in the RoboKid project (Anumolu et al., 1997, Bray et al., 1997, Reilly and Bray, 2000c, Reilly and Bray, 2000d) and in training and education in inter-disciplinary research entailed by such a project (Reilly and Bray, 2000a and Reilly and Bray, 2000b). Decision table processing, due in part to its facile representation in logic forms, plays a role in the Barrel system (Reilly et al., 1995 and Barrett et al., 1995) and was used in a hybrid or combined numerical-symbolic simulation by Reilly and Oliver (1988). Particularly acute is the matter of computations. A logic programming solution in Prolog, e.g., does not admit function evaluation *on access* of an implication.

Output computations, however, are elementary, allowing for statistic computations, e.g., of percent profit in the table listed in Figure 1. It also allows development of "table dominant systems," in which whole systems are characterized by sequences of tables, connected through, in the decision table case, the action entries. All said and done, we propose here a Prolog representation, one relation per table rule:

```
table(reo,good,5%,2hrs,no)
table(reo,poor,15%,2days,yes)
table(cord,good,5%,2hrs,no)
table(cord,poor,10%,1day,no)
table(duesenberg,good,7.5%,4hrs,yes)
table(duesenberg,poor,17.5%,2days,yes)
```

This schema provides typical queries (initial denials, mathematically), which follow the "original" functional form of the decision table. I.e. a query of the form

```
table(duesenberg,poor,Commisn,ShopW,ManOK)
```

instantiates the variables, Commisn, ShopW and ManOK to the values 17.5%, 2-days, and yes, as called for by the table. However, we can also provide queries which have in them values that belong to the action portion of the table, such as:

```
table(CarMake,Shape,17.5%,2-days,yes)
```

and receive, as output, what is recognized by the table as "input," i.e., the values of the rule that contains these (output) actions within its (input) conditions. More than one rule may be output also, since Prolog processes relationally (i.e. is not restricted to functions). And, finally, we can mix and match, with some values from the table's "input" and some from the table's "output." The control program (the interpreter, compiler) then "fills in the blanks. So, an initial denial such as

```
table(CarMake,poor,Comm,ShopW,yes)
```

returns {reo,15%,2-days} and {duesenberg,17.5%,2-days} for its instantiated variables.

The property we have just presented may be called "input-output indifference." It is an important property such that for any for which it obtains it implies that a solution in the "forward" direction also yields one in the "reverse" direction. This holds for scientifically important cases of functions, whose solutions frequently, on inversion, become relations.

The astute observer probably is well aware of the limitations --- what Kowalski's book (1979) calls "intolerable non-determinism," which occurs all too frequently in problem domains. (Hint: try a real/genuine inverse to a trigonometric function (not a gerrymandered "principal domain" function). Limits or no limits, we take this input-output indifference seriously as we move to an agent based operation in which the content of the table (logic formulation) is transmitted to a neural network structure, which can add to our processing capabilities what amounts to a significant intelligent processing capability, achievable in the logic programming solution only with much additional code (and predetermined statistical processing schemes).

Agent Communication Code in the previous section can readily be processed to transmit the key system content in a more abbreviated form. The tuple organization can readily be processed at a node we now describe. We take an implementer's point of view.

Localist Neural Net The (recipient) node manages a set of files that underlie a particular implementation of a localized neural net popularized under the name, Interactive Activation and Competition (IAC), by McClelland and Rumelhart (1998). This model is used extensively in cognitive processing models (Grainger and Jacobs, 1998) and serves well as a form of "intelligent" database. The implementation sports an interactive form of processing for users as well as mechanisms whereby (output) files can be generated for yet further processing; we resist any attempt to get drawn into some of the potential here --- it goes beyond our current scope! This (receiving) node first analyses the data to determine requirements for interactive processing ('on screen' display, etc.) and for generating data to send to other nodes, including perhaps the originating (logic) node. Scripts are generated for developing more permanent and/or exportable data; these might be generated automatically for some more or less standard data processing protocols.

IAC NN The structure for the model may be a pure or vanilla IAC form or may be one of a number of IAC-like forms (forms which alter way in some of the distinctly defined IAC). The choices may be viewed as model abstractions, relative to the initial logic forms (or vice versa) and relative to one another (Reilly et al., 2001). Due to scope limits, we do (only) an IAC model.

Each tuple of the logic formula is given a unique key; such a key can be formed as in relational database theory or formed intuitively, e.g., rule1 ... rule6, in this simple case. These keys are represented as neurons and are incorporated into a "pool" of "instance nodes." Similarly, we group the values {reo, cord, duesenberg}, {good, poor}, and the output values into separate pools. Note that these sets represent mutually exclusive values. I.e., a car cannot be both a cord and a reo. A manager's OK is required or it is not. (We can of course fuzzify things, perhaps through so-called linguistic values --- and (perhaps) do limited "outside" fuzzy processing, reminiscent in many ways of what we commented upon in connection with the logic models in the previous section.)

Positive & Negative Links These constructions give us, here, five additional pools over and above the instance pool. Following IAC strictures, we lace the instance node of each tuple to every value in its (logic formulated) tuple by links of value 1. Back links of value 1 are also put in place --- key to the retaining the input-output indifference of the (originating) logic formulation. Additionally, in a move that represents a form of signal enhancement, and precisely directed to emphasizing the inherent mutual exclusivity, links of -1 are laced among all nodes in each and every pool. Thus, positive links emerge as "between pools" connections and negative or "inhibitory" links emerge

as "within pool" connections. All other connections are zero, including the important case that each node does not, in a strict IAC model, communicate with itself. The reader may note that, at least as an abstraction, the model produces a matrix with a zero diagonal. It is symmetric (due to mutual links between nodes). Processing is bi-directional and input-output indifferent. Distinctly appearing sub-matrices arise in response to the mutually inhibiting nodes of each pool.

Reverberation Processing is normally "reverberating," i.e., the model goes through cycles in which each node communicates activation (including decrementing action) to every node to which it is linked. Notice that such reverberation implies repeated matrix multiplication, which in the absence of some controlling element, tends to increase values without bound. To prevent this, non-linear transformations are applied to output vectors; in this case, e.g., a "squasher" keeps activations on a range, say [-1,1]. Matrix multiplication and non-linear transformation are the "heart and soul" of neural network processing.

"Intelligent" Processing Typically, these machinations give the model multiple forms of response. In some (we might call them "regular") cases the model first gives an "information retrieval" answer comparable to what is obtained in the logic case (again noting that I-O indifference is retained). Later, the model performs a kind of multivariate correlation analysis on its content. This gives rise to a variety of "intelligent" responses from which we can ascertain responses such as how "close" (similar) rules are to each other (and among one another). For example, if two rules sport a similarity, like the 2-hours of shopwork required, no manager ok required, etc. it will not be missed! Consider the case:

```
table(reo,good,5\%,2-hours,no)
table(cord,good,5\%,2-hours,no)
```

Notice that some of these "intelligent" responses can be generated within the logical formulation, but, typically, only after a considerable amount of exploration (which must be appropriate to the application) and a sizable amount of code; i.e., the code must contain within it multivariate correlation schemes. This seems to be a course of processing we prefer to avoid. We think again of a Kowalski comparison between "finding" and "showing" (Kowalski, 1979), it being the case that "showing" is generally much easier than "finding" a solution.

Agent Theme An interesting prospect lies in having the IAC send some of its results back to the logic node and providing "show" opportunities for the logic formulation. Logic code in such a case is constrained (or directed) and thus can be easier than what a general attack would require. We close with this, completing of a cycle of processing, shared communication and results of the pair of agents.

SUMMARY

In this paper we first introduce a revised view of agent computing (relative to our previous work). We next discussed "light weight use of logic" (LWUL). We discussed our previous attempt at defining a notion of symbolic simulation to be joined with numerical (discrete and continuous) simulation. This notion entails a form of processing which includes logic and (character and structure) pattern matching at its base. The associated definitional scheme is formal (operational semantics on an abstract machine, which also was used to self-define Prolog and to define other (toy) languages). It is thus logic based at its roots. Our development proceeded through stages of relaxing such that it truly became an exercise in LWUL. The varying levels of formality in this sequence of processors provide model abstraction opportunities and, fittingly, support an agent style of processing within simulation in a form that is perhaps not usually contemplated. In our final sections, we concentrated on application of logic programming (Prolog) to decision making (with casual reference to some psychology). We scoped a few problems not covered in the paper, but our main emphasis fell on how, in agent style, content of the tables can be communicated to a node which "specializes" in a form of NN which both retains some of the strong properties of the decision tables (and logic formulation) and proffers an extended form of processing. Finally, and once again in an agent purview, results generated in both the logic and neural net systems can be shared and methods of processing can be suggested via communicating "show" computation opportunities, say, by the neural net to the logic programs.

Acknowledgements This work is partially supported by a University of Alabama at Birmingham Mental Retardation Research Center Grant (MRRC) NICHD P30HD38985-02 (Core D - Longitudinal Studies and Statistical Modeling).

REFERENCES

- Anumolu, V., N. W. Bray, and K. D. Reilly. Neural network models of strategy development in children. *Neural Networks*, 10(1):7-24, 1997.
- Barrett, J. H., Reilly, K. D., Tarnig, J. and R. M. Hyatt. A computerized formal means to reason about components in simulation models and environments -- Part II: System development methodology. *Trans. Soc. Comp. Simulation*, 12(3):191-244, 1995.
- Bratman, M. *Intentions, Plans and Practical Reason*. Harvard U. Press, Cambridge, MA, 1987.
- Bray, N., Reilly, K., Villa, M, and L. Grupe. Neural network models and mechanisms of strategy development. *Developmental Review*, 17:526-566, 1997.

- Cohen, P. and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3):213-261, 1990.
- Franklin, S. and A. Graesser. Is it an agent or just a program?: A taxonomy for autonomous agents. In Wooldridge, M. J. and N. R. Jennings, editors, *Proc. Third International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag, New York, NY, 1996. 11 pp.
- Grainger, J. and A. M. Jacobs, editors. *Localist Connectionist Approaches to Cognition*. L. Erlbaum, Mahwah, NJ, 1998.
- Hooper, J. W. and K. D. Reilly. An algorithmic analysis of simulation strategies. *Int'l. J. Comp. & Info. Sciences*, 11(2), 1982.
- Hooper, J. W. and K. D. Reilly. The 'GPSS-GASP Combined' (GGC) system. *Int'l. J. Comp. & Info. Sciences*, 12(2):111-136, 1983.
- Kowalski, R. *Logic for Problem Solving*. Elsevier Press, New York, NY, 1979.
- McClelland, J. L. and D. E. Rumelhart. *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises*. MIT Press, Cambridge, MA, 1988.
- Montalbano, M.. *Decision Tables*. Science Research Associates, Chicago, IL, 1974.
- Morse, K., D. Cutts, J. Hancock, and S. A. Lubbers. Feasibility and functionality of autonomous objects in the HLA. *Incomplete reference*.
- Moss, C. D. S. *The Formal Description of Programming Languages using Predicate Logic*. PhD thesis, Imperial College, London, Dept. of Computing, 1981.
- Reilly, K. D., Barrett, J. H. Tarnag, J. and R. M. Hyatt. A computerized formal means to reason about components in simulation models and environments -- Part I: A logic-based approach. *Trans. Soc. Comp. Simulation*, 12(2):101-178, 1995.
- Reilly, K. D. and N. W. Bray. Goal sketch modeling with neural networks. In Callaos N., Pham, T., Kudo, M., Funabiki, N., and D. Andina, editors, *Proc. SCI 2000 (World Multiconference on Systemics, Cybernetics, and Informatics) Vol. 3 Virtual Engineering and Emergent Computing*, pages 318-323. International Institute of Informatics and Systemics, 2000.
- Reilly, K. D. and N. W. Bray. Human learning models and data collection over the "Long Haul." In: Khosrow-Pour, Mehdi, editor, *Proc. IRMA 2000*, pages 1033--1034. Information Resource Management Association, 2000.
- Reilly, K. D. and N. W. Bray. Prolegomenon to computer-based great epistemologies education. In: Khosrow-Pour, Mehdi, editor, *Proc. IRMA 2000*, pages 1035-1036. Information Resource Management Association, 2000.
- Reilly, K. D. and N. W. Bray. Trends in simulation and modeling and the RoboKid Project. In Sanchez, B., Hammel II, R., Soriano, M., and P. Tiako, editors, *Proc. SCI 2000 (World Multiconference on Systemics, Cybernetics, and Informatics) Vol. 9 Industrial Systems*, pages 50-55. International Institute of Informatics and Systemics, 2000.
- Reilly, K. D. and J. Oliver. A neural control element in a control systems application. In *Proc. First Int'l Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 507-513. Association for Computing Machinery, 1988.
- Reilly, K. D., Salah, A. I. and C. C. Yang. A logic programming perspective on decision table theory and practice. *Data and Knowledge Engineering*, 2:191-212, 1987.
- Reilly, K. D., A. Sprague, and A. Fanning. Biologically inspired simulation: Agent metaphors and immune systems. In *Proc. 2001 Summer Simulation Conference*, Orlando, FL, San Diego, CA, 2001. Society for Computer Simulation, Int'l. In press.
- Reilly, K. D., Sprague, A. and C. L. Plachco. Agents and model abstractions in intelligent systems simulations. In *Proc. 2001 Summer Simulation Conference*, Orlando, FL, San Diego, CA, 2001. Society for Computer Simulation, Int'l. In press.
- Robertson, D. and J. Agusti. *Software Blueprints: Lightweight Uses of Logic in Conceptual Modeling*. Addison-Wesley, Reading, MA, 1999.
- Salah, A. I. and K. D. Reilly. A reduction methodology for a differential diagnosis expert system. *Int'l J. Approximate Reasoning*, 1:131-139, 1987.
- Sowa, J.. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole, Pacific Grove, CA, 2000.
- Wooldridge, M. J. and N. R. Jennings. *Intelligent Agents - Theories, Architectures and Languages*. Springer-Verlag, New York, 1995.
- Zeigler, B. P.. *Theory of Modelling and Simulation*. J. Wiley, New York, 1976. Reissued by Krieger Pub. Co., Malabar, FL, 1985.