Section 5. Mathematical Foundations of AI

5.1. Algorithms

RAISING EFFICIENCY OF COMBINATORIAL ALGORITHMS BY RANDOMIZED PARALLELIZATION

Arkadij D. Zakrevskij

Abstract: *A new approach is proposed to deal with some hard combinatorial optimization problems, which admit a certain reformulation. Considering such a problem, several similar problems are prepared differing in initial data but having the same set of solutions. They are solved in parallel until one of them will be solved, and that solution is accepted. Notwithstanding the evident overhead, the whole run-time could be significantly reduced due to dispersion of velocities of combinatorial search in regarded cases. The efficiency of this approach is investigated on the concrete problem of finding short solutions of non-deterministic system of linear logical equations.*

Keywords: *combinatorial problems, combinatorial search, parallel computations, randomization, run-time, acceleration.*

Introduction

There exist a variety of various hard combinatorial optimization problems, which could be more quickly solved when changing each of them for a selection of other problems solved in parallel. These new problems could seem quite different, but nevertheless they can be made equivalent to the initial problem, that means they will have the same set of solutions. The overhead expenses for constructing these sets of new problems and dealing with them can be compensated with interest by the essential acceleration of the search for solution of the considered problem.

That idea is demonstrated below by the problem of finding a short solution of an undefined system of linear logical equations.

Let us regard a system of m *linear* Boolean equations with n *variables*

$$\begin{aligned} &a_1^1x_1 \oplus a_1^2x_2 \oplus \ldots \oplus a_1^nx_n = y_1 ,\\ &a_2^1x_1 \oplus a_2^2x_2 \oplus \ldots \oplus a_2^nx_n = y_2 ,\\ &\qquad\qquad \ldots \\ &a_m^1x_1 \oplus a_m^2x_2 \oplus \ldots \oplus a_m^nx_n = y_m , \end{aligned} \qquad (1)$$

where a_i^j are the coefficients of the system, x_j – unknown variables ($x_j \in \{0, 1\}$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$), $\oplus$ is the EXOR operator.

This system can be represented in a compact matrix form:

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{y} , \qquad (2)$$

where $\boldsymbol{A}$ is a Boolean $m \times n$ coefficient matrix, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ – Boolean vectors. AND operator is used as an internal one, and EXOR as the external one for the matrix multiplication.

$$\bigoplus_{j=1}^{n} a_i^j x_j = y_i . \quad (3)$$

Solutions (roots) of this system are the sets of variable values, for which all equations will be satisfied. In this case EXOR sum of the columns of matrix ***A***, corresponding to the component-wise sum by modulo 2 of the columns of ***A***, which correspond to the variables having value 1, should be equal to the column ***y***. The system can be deterministic (having the single solution), non-deterministic (several solutions exist) or contradictory, where there are no solutions [1, 2]. Let us assume that $n > m$ and all rows of ***A*** are linearly independent. Then the number of possible solutions is equal to 2^{n-m}.

A great property of the linear equations is that they can be solved in regard of any variable. An effective Gauss method of variable exclusion is based on this property [3] and allows finding the only solution for a deterministic system or the set of all possible solutions for a non-deterministic one. But it is not sufficient when some optimal solution should be selected from that set which could be very large.

Looking for Short Solutions of Systems of Linear Logical Equations

Solving a non-deterministic system of linear logical equations, it is necessary sometimes to find a solution with minimum number of variables taking value 1 – such a solution is called *shortest*. That is important in many practical cases, for instance, when logic circuits in AND/EXOR basic are synthesized [4-6] or when some problems of information security are considered.

Evidently, a shortest solution could be found by means of selecting from matrix ***A*** one by one all different combinations of columns, consisting first of 1 column, then of 2 columns, etc. and examining each of them to see if their sum equals vector ***y***. As soon as it happens, the current combination is accepted as the sought-for solution. That moment could be forecasted. If the weight of the shortest solution (the number of 1s in vector ***x***) is w, the number N of checked combinations is defined approximately by the formula (where C_n^i is the number of i-element subsets taken from a set of n elements).

$$N = \sum_{I=0}^{w} C_n^i \quad (4)$$

and could be very large, as is demonstrated below.

It was shown [7], that the expected weight γ of the shortest solution of an SLLE with parameters m and n can be estimated before finding the solution itself. We represent this weight as a function $\gamma(m, n)$. First we find the mathematical expectation $\alpha(m, n, k)$ of the number of solutions with weight k. We assume that the considered system was randomly generated, which means that each element of ***A*** takes value 1 with the probability 0.5 and any two elements are independent of each other. Then the probability that a randomly selected column subset in matrix ***A*** is a solution equals 2^{-m} (probability that two randomly generated Boolean vectors of size m are equal). Since the number of all such subsets having k elements equals $C_n^k = n! / ((n-k)!\, k!)$, we get:

$$\alpha(m, n, k) = C_n^k 2^{-m}. \quad (5)$$

Similarly, we denote as $\beta(m, n, k)$ the expected number of the solutions with weight not greater than k:

$$\beta(m, n, k) = \sum_{i=0}^{k} C_n^i 2^{-m}. \quad (6)$$

Now, the expected weight γ of the shortest solution can be estimated well enough by the maximal value of k, for which $\beta(m, n, k) < 1$:

$$\gamma(m, n) = k, \quad \text{where } \beta(m, n, k) < 1 \le \beta(m, n, k+1). \quad (7)$$

For example, for a system of 40 equations with 70 variables the expected weight γ equals 10, and reaches 31 when $m = 100$ and $n = 130$.

In the last case, the described above simple algorithm should check about 10^{30} combinations, according to formula (2) in which $w = \gamma$. Examining them with the speed of one million combinations per second we need about **30 000 000 000 000 000 years** to find the solution. Practically impossible!

A great acceleration can be achieved by using Gaussian method of variables exclusion [3] developed for solving systems of linear equations with real variables and adjusted for Boolean variables [5]. It enables to avoid checking all subsets of columns from ***A*** which have up to *w* columns, when only one of 2^{n-m} regarded combinations presents some roots of the system.

Its main idea consists in transforming the extended matrix of the system (matrix ***A*** with the added column ***y***) to the *canonical form*. A maximal subset of *m* linear independent columns (does not matter which one) is selected from ***A*** and by means of equivalent matrix transformations (adding one row to another) is transformed to **I**-matrix, with 1s only on the main diagonal. That part is called a basic, the rest $n - m$ columns of the transformed matrix ***A*** constitute a *remainder*. The column y is changed by that, too.

According to this method the subsets of the remainder are regarded, i.e. combinations selected from the set which has only $n - m$ columns (not all *n* columns!). It is easy to show that every of these combinations enables to get a solution of the considered system (any sum of its elements can be supplemented with some columns from matrix **I** to make it equal to ***y***). When we are looking for a shortest solution using this method, we have to consider different subsets of columns from the remainder, find the solution for each such subset and select a subset, which generates the shortest solution. If it is known that the weight of the shortest solution is not greater than *w*, then the level of search (the cardinality of inspected subsets) is restricted by *w*. Note that if $w \geq n - m$, then all 2^{n-m} subsets must be searched through.

Now, for the same example ($m = 100$ and $n = 130$), $N \cong 10^9$, which means that the run-time of Gaussian method is about only **17 minutes**.

Unfortunately (for practical application), it rises very quickly when increasing parameters *m* and *n*. For example, when $m = 625$ and $n = 700$, the number of checked combinations equals 2^{75} and the run-time surpasses one milliard years.

It could be significantly reduced when we know that there exists a short solution with weight *w* less than γ. For example, the following situation could be imagined: a random matrix ***A*** is generated, then *w* columns are selected by random, and their component-wise modulo 2 sum is defined as vector ***y***. In that case the searching (checking different solutions one by one in order to find the shortest) could be interrupted as soon as the weight *v* of current solution satisfies the inequality $v < \gamma$ and we conclude that the sought-for solution is found. This idea lies in the base of the recognition method [8, 9].

The Level of Search and its Dispersion

Suppose, a short solution exists presented by *w* columns of matrix ***A*** (their sum equals vector ***y***). Let *p* of them belong to the remainder. Then this solution will be found when such subsets of columns from the remainder are checked which have exactly *p* elements. In other words, the solution will be found on the *level of search* L when this one equals *p*.

That quantity depends, first, on the regarded example (how matrix ***A*** was generated) and, second, on the way of getting the remainder (how *m* linearly independent columns were selected from initial matrix ***A***, which determines splitting that matrix into two parts, basic and remainder).

An experiment was conducted, where 10 different random matrices ***A*** were generated with parameters $n = 500$ and $m = 430$. In each of them 30 different subsets of linearly independent columns were selected and corresponding canonical forms were constructed, which produced 300 variants of the remainder. A set of 75 columns was chosen from ***A*** by random constituting a short solution - their sum was accepted as vector ***y***.

All intersections of that set with remainders were found. The cardinalities of these intersections (numbers of columns in them) define the level of search *L* on which the considered short solution will be found. The values of *L* for all 300 variants are shown in Table 1. Its columns correspond to different examples (*i*) of system and the rows correspond to different remainders in them (*j*) selected at random. As one can see, this quantity is subjected to a great dispersion.

The values of run-time (using PC COMPAC Presario, 1000 MH) corresponding to different levels are presented in Table 2, measured in seconds (*s*), minutes (*m*), hours (*h*), days (*d*) and years (*y*). The distribution of levels is shown on the right side of the table, *N* indicating the number of entries with value *L* in Table 1.

Table 1. Dispersion of level L in space of examples and remainders

j \ *i*	1	2	3	4	5	6	7	8	9	10
1	10	13	12	12	16	11	13	10	12	13
2	8	9	13	9	9	10	11	10	7	9
3	10	10	11	9	11	13	11	10	12	14
4	10	8	11	11	10	8	12	10	13	11
5	10	11	14	6	11	8	17	15	10	6
6	12	6	7	10	12	11	11	12	8	16
7	9	11	8	9	10	12	8	9	8	17
8	8	15	6	13	14	16	15	12	12	12
9	10	13	9	5	14	11	13	6	16	10
10	14	15	9	8	11	10	8	6	13	12
11	14	12	10	12	6	10	12	15	13	9
12	8	9	12	16	16	7	18	11	9	12
13	9	11	6	13	15	9	9	10	10	9
14	14	12	9	14	6	9	7	13	11	13
15	9	5	13	12	11	10	7	13	9	16
16	13	16	6	15	10	7	4	10	11	13
17	6	10	15	13	11	11	13	11	9	12
18	9	9	10	11	10	6	14	8	10	12
19	5	15	7	11	10	16	9	12	9	13
20	11	11	7	8	7	7	11	9	13	10
21	8	12	10	8	8	13	9	6	12	8
22	15	8	11	8	16	6	12	11	15	5
23	10	12	7	14	11	12	13	13	9	9
24	10	9	10	12	4	10	12	11	10	10
25	9	11	10	10	11	11	14	9	7	19
26	11	14	8	10	11	5	14	8	12	10
27	11	12	14	9	10	7	14	11	11	8
28	18	12	13	12	6	8	8	10	12	11
29	12	12	7	9	14	12	7	8	9	12
30	10	7	6	11	13	8	11	15	11	18

Table 2. Dependence of run-time *T* on level *L*

L	*T*	*N*	
4	1.6s	2	11
5	22s	5	11111
6	4m	16	1111111111111111
7	38m	16	1111111111111111
8	5h	27	111111111111111111111111111
9	1.5d	36	111111111111111111111111111111111111
10	10d	44	11111111111111111111111111111111111111111111
11	55d	44	11111111111111111111111111111111111111111111
12	280d	39	111111111111111111111111111111111111111
13	3.6y	27	111111111111111111111111111
14	15y	16	1111111111111111
15	59y	12	111111111111
16	214y	10	1111111111
17	713y	2	11
18	2213y	3	111
19	6394y	1	1

Note that fulfilled experiments were virtual ones: the run-time was not measured directly but was forecasted by the method suggested in [10] and ensuring a rather good approximation.

It logically follows from these tables that efficient algorithms for finding short solutions may be constructed which solve the problem in quasi-parallel mode using a set of q canonical forms of the system ($\boldsymbol{A}$, $\boldsymbol{z}$) with different basics selected at random. Using these forms one by one, such an algorithm consecutively increases the cardinality of checked combinations of columns taken from the remainders and terminates when finding the short solution.

Experiments

Some results of the program virtual implementation of that algorithm for different values of q are shown in Table 3. Thirty examples of random systems with parameters n = 1000, m = 900 and the weight of the short solution w = 100 have been generated and solved. The following denotation is used in the table: **No** is the number of the regarded example, **N** is the number of form where its solution was found, **L** is the level at which it was found, and **T** is the time spent for it.

Table 3. Results of solving non-deterministic systems of linear logical equations with parameters n = 1000, m = 900, w = 100.

	q = 1			q = 10			q = 30			q = 300		
No	**N**	**L**	**T**	**N**	**L**	**T**	**N**	**L**	**T**	**N**	**L**	**T**
1	1	8	7d	9	6	16h	9	6	18h	33	4	19m
2	1	9	69d	8	6	14h	16	5	2h	254	2	4m
3	1	10	2y	7	7	7d	28	6	2d	234	2	4m
4	1	13	776y	3	6	5h	3	6	8h	117	3	5m
5	1	3	4s	1	3	4s	1	3	5s	1	3	4m
6	1	10	2y	5	7	5d	26	5	3h	56	2	4m
7	1	12	112y	9	4	3m	9	4	3m	57	3	5m
8	1	8	7d	10	5	1h	10	5	1h	106	3	5m
9	1	12	112y	10	5	1h	28	4	10m	141	3	6m
10	1	9	69d	2	6	4h	2	6	6h	95	2	4m
11	1	13	776y	7	8	82d	15	4	5m	50	2	4m
12	1	13	776y	9	8	104d	14	5	2h	117	3	5m
13	1	10	2y	8	6	14h	8	6	16h	35	3	4m
14	1	6	58m	1	6	2h	1	6	4h	39	3	4m
15	1	6	58m	1	6	2h	11	5	1h	134	3	6m
16	1	14	4954y	2	8	27d	28	4	10m	205	3	7m
17	1	6	58m	1	6	2h	14	5	2h	49	3	4m
18	1	10	2y	2	7	2d	27	5	3h	285	2	4m
19	1	13	776y	8	7	8d	8	7	9d	84	3	5m
20	1	10	2y	2	6	4h	2	6	6h	203	4	1,3h
21	1	8	7d	7	5	45m	7	5	52m	93	4	39m
22	1	7	13h	10	6	17h	27	4	9m	190	3	6m
23	1	12	112y	2	5	13m	16	2	4s	16	2	4m
24	1	15	29185y	4	7	4d	24	6	2d	226	2	4m
25	1	10	2y	2	6	4h	2	6	6h	46	3	4m
26	1	13	776y	9	7	9d	16	5	2h	112	4	45m
27	1	10	2y	6	8	71d	16	4	6m	281	3	8m
28	1	12	112y	7	8	82d	18	6	1d	87	3	5m
29	1	12	112y	6	4	2m	6	4	2m	254	2	4m
30	1	12	112y	7	8	82d	22	6	2d	31	4	18m
The sum:			38704y			1,3y			20d			5,3h

Note that value 1 of parameter q corresponds to the pure Gaussian method dealing with one canonical form of the system. Changing it for pseudo-parallel algorithm on the base of 300 forms we accelerate finding the short solution on an average in 46 million times. The immense acceleration!

Conclusion

A new approach to solving hard combinatorial optimization problems is suggested, demonstrated on the problem of finding a short solution of a non-deterministic system of linear logical equations. Its idea is in changing the regarded problem for a set of other similar problems equivalent to the given one and solving them in parallel. The run-time could be considerably reduced by that, possibly in many millions times, as computer experiments show.

Acknowledgements

The work had been performed thanks to the partly support of ISTC (Project B-986) and the Fond of Fundamental Researches of Belarus.

Bibliography

1. Kostrikin A., Manin Y. Linear algebra and geometry. Translated from the second Russian (1986) edition by M. E. Alferieff. Revised reprint of the 1989 English edition. Gordon and Breach, 1997.
2. Lankaster P. Theory of matrices, Academic Press, New York – London, 1969.
3. Gauss C.F.Beitrage zur Theorie der algebraischen Gleichungen, Gött, 1849.
4. Sasao T. Representations of logic functions using EXOR operators. – In "Representations of discrete functions" (ed. by T. Sasao and M. Fujita) – Kluwer Academic Publishers, Boston/London/Dordrecht, 1996, pp. 29-54.
5. Zakrevskij A.D. Looking for shortest solutions of systems of linear logical equations: theory and applications in logic design. - 2.Workshop "Boolesche Probleme", 19./20. September 1996, Freiberg/Sachsen, pp. 63-69.
6. Zakrevskij A.D., Toropov N.R. Polynomial representation of partial Boolean functions and systems. – Minsk, Institute of technical cybernetics, Belarusian NAS, 2001. (In Russian).
7. Zakrevskij A.D., Vasilkova I.V. Fast algorithm to find the shortest solution of the system of linear logical equations. - Problems of logical design. - Minsk, 2002. – Pp. 5-12. (In Russian).
8. Zakrevskij A.D. Randomization of a parallel algorithm for solving undefined systems of linear logical equations. – Proceedings of the International Workshop on Discrete-Event System Design – DESDes'04. – University of Zielona Gora Press, Poland, 2004, pp. 97-102.
9. Zakrevskij A.D. Efficient methods for finding shortest solutions of systems of linear logical equations. – Control Sciences, 2003, No 4, pp. 16-22. (In Russian).
10. Zakrevskij A.D., Vasilkova I.V. (2003). Forecasting the run-time of combinatorial algorithms implementation. -Methods of logical design, issue 2. Minsk: UIIP of NAS of Belarus, pp. 26-32. (In Russian).

Author's Information

Arkadij Zakrevskij – United Institute of Informatics Problems of the NAS of Belarus, Surganov Str. 6, 220012 Minsk, Belarus; e-mail: zakr@newman.bas-net.by

SPECIFYING AGENT INTERACTION PROTOCOLS WITH PARALLEL CONTROL ALGORITHMS

Dmitry Cheremisinov, Liudmila Cheremisinova

Abstract. *The purpose of the paper is to explore the possibility of applying existing formal theories of description of distributed and concurrent systems to interaction protocols for real-time multi-agent systems. In particular it is shown how the language, proposed for description of parallel logical control algorithms and rooted in the Petri net formalism, can be used for the modelling of complex concurrent conversations between agents in a multi-agent system. It is demonstrated with a known example of English auction on how to specify an agent interaction protocol using considered means.*

Keywords: *multi-agent system, interaction protocols, parallel control algorithm*

Introduction

Agents are becoming one of the most important topics in distributed and autonomous decentralized systems, and there are increasing attempts to use agent technologies to develop large-scale commercial and industrial software systems. Over the last decade, the specification, design, verification and application of a particular type of agents, called BDI (belief, desire, intention) agents [1], have received a great deal of attention. BDI agents are systems that are situated in changing environment receive continuous perceptual input and take actions to affect

their environment based on their internal state. In particular, there are many efforts aimed at developing agent-oriented designs that are typically structured as multi-agent systems (MASs). MAS is a computational system in which two or more agents interact or work together to perform a set of tasks or to achieve a set of goals [2]. Agents of a MAS interact with others toward their common specific objective or individual benefit. Agent interactions are established through exchanging messages that specify the desired performatives of other agents (such as notice, request) and declarative representations of the content of messages.

MAS is usually specified as a concurrent system based on the notion of autonomous, reactive and internally-motivated agents acting in a decentralized environment. One key reason of the growth of interest in MAS is that the idea of an agent as an autonomous system, capable of interacting with other agents in order to satisfy its design objectives, is a naturally appealing one for software designers. Agents are used in an increasingly wide variety of applications such as distributed and autonomous decentralized systems. The complexity of MASs suggests a pressing need for system modelling techniques to support reliable, maintainable and extensible design. Although there are many efforts aimed at developing such MASs, there is sparse research on formal specification and design of such systems.

Agent system can operate if the agents are able to exchange information in the form of messages and if they have a common understanding of the possible types of messages that are connected with message “content”. This shared understanding is referred to as an ontology [3]. A great deal of agent-based research has devoted to the development of techniques for representing ontology’s. Multi-agent conversations are built upon two components: agent communication language and interaction protocol. There are a number of agent communication languages, such as Knowledge Query and Manipulation Language (KQML) [4] and the Foundation for Intelligent Physical Agents (FIPA) ACL [5] and others designed for special purposes and that are like mentioned ones. These agent communication languages specify a domain specific vocabulary (ontology) and the individual messages that can be exchanged between agents.

Interaction protocols [5] specify the sequences in which these messages should be arranged in agent interactions. A group of rational agents complies with an interaction protocol in order to engage in task-oriented sequences of message exchange. Thus, when an agent sends a message, it can expect a receiver's response to be among a set of messages indicated by the protocol and the interaction history. With a common interpretation of the protocol, each member of the group can also use the rules of the interaction in order to satisfy its own goals. In other words protocol constrains the sequences of allowed messages for each agent at any stage during a communicative interaction (dialogue), i.e. it describes some standard pattern messages exchanged between agents need to follow. Protocol plays a central role in agent communication. It specifies the rules of interaction between communicating agents and the first thing should be done by the designer developing any particular real-time system is to impose interaction protocol.

This paper explores the possibility of applying existing formal theories of description of distributed and concurrent systems to interaction protocols for real-time multi-agent systems. In particular it is shown how the language PRALU [6, 7], proposed for description of parallel logical control algorithms and rooted in the Petri net formalism, can be used to describe agent interaction protocols. The described approach can be used for the modelling of complex, concurrent conversations between agents in a multi-agent system. It can be used to define protocols for complex conversations composed of a great number of simpler conversations. With the language PRALU it is possible to express graphically the concurrent characteristics of a conversation, to capture the state of a complex conversation during runtime, and to reuse described conversation structure for processing multiple concurrent messages. It is demonstrated with a known example of English auction [5] on how to specify an agent interaction protocol using considered means. Finally, using PRALU language we can verify some key behavioral properties of our protocol description that is facilitated by the use of existing software for the language PRALU [6, 7, 8].

Agent Interaction Protocols

The application domains of MASs are getting more and more complex. Firstly, many current application domains of MASs require agents to work in changing environment (or world) that acts on or is acted on by the system. A closed system is one that has no environment; it is completely self-contained in contrast to an open (uncertain) system, which interacts with its environment. Any real system is open. The MAS must decide what to do and develop a strategy in order to achieve its assigned goals. For this, the MAS must have a representation or a

model of the environment in which it evolves. The environment is composed of situations. A situation is the complete state of the world at an instant of time.

The application of multi-agent systems to real-time environments can provide new solutions to very complex and restrictive systems such as real-time systems. A suitable method for real-time multi-agent system development must take into account the intrinsic characteristics of systems of this type. As a rule they are distributed, concurrent systems with adaptive and intelligent behaviour. For agent-based systems to operate effectively, they must understand messages that have a common ontology underlying them. Understanding messages that refer to ontology can require a considerable amount of reasoning on the part of the agents, and this can affect system performance.

BDI agents are systems that are situated in a changing environment, receive continuous perceptual input, and take actions to affect their environment, all based on their internal mental state. Within the BDI architecture agents are associated with beliefs (typically about the environment and other agents), desires or goals to achieve, and intentions or plans to act upon to achieve its desires. In practical terms, beliefs represent the information an agent has about the state of the environment. It is updated appropriately after each action. The desires denote the objectives to be accomplished, including what priorities are associated with the various objectives. Intentions reflect the actions that must be fulfilled to achieve the goal (the rules to be fired).

To reduce the search space of possible responses to agent messages, interaction protocols can be employed. They specify a limited range of responses that are appropriate to specific message types for a given protocol. When an agent is involved in a conversation that uses an interaction protocol, it maintains a representation of the protocol that keeps track of the current state of the conversation. After a message is received or sent, it updates the state of the conversation in this representation.

By the very nature of protocols as public conventions, it is desirable to use a formal language to represent them. When agents are involved in interactions where no concurrency is allowed, conversation protocols are traditionally specified as deterministic finite automata (DFA) of which there are numerous examples in the literature. DFA consists of a set of states, an input alphabet and a transition function, which maps every pair of state and input to next state. In the context of interaction protocols [9], the transitions specify the communicative actions to be used by the various agents involved in a conversation. A protocol based on such a DFA representation determines a class of well-formed conversations. Conversations that are defined in this way have a fixed structure that can be laid down using some kind of graphical representation.

Protocols can be represented as well in a variety of other ways. The simplest is a message flow diagram, as used by FIPA [5]. More complex protocols will be better represented using a UML sequence (Uniffied Modelling Language) [10] and AUML [11], interaction diagram, statechart [12] and Colored Petri Net (CPN) [13]. UML is one of the currently most popular graphical design languages that is de facto standard for the description of software systems. AUML extends UML sequence diagrams to represent asynchronous exchange of messages between agents. The advantage of AUML is its visual representation. Statecharts [12] are an extension of conventional DFAs. However, expressing protocols of realistic complexity using statecharts or AUML requires substantial efforts for developing, debugging and understanding.

A CPN model of a system describes the states, which the system may be in, and the transitions between these states. CPNs provide an appropriate mathematical formalism for the description, construction and analysis of distributed and concurrent systems. CPNs can express a great range of interactions in graphical representations and well-defined semantics, and allow formal analysis and transformations [14]. By using CPNs, an agent interaction protocol can be modeled as a net of components, which are carriers of the protocol structure. Using CPNs to model agent interaction protocol, the states of an agent interaction are represented by CPN places. Each place has an associated type determining the kind of data that the place may contain. Data exchanged between agents are represented by tokens, and the colors of tokens indicate the data value of the tokens. The interaction policies of a protocol are carried by CPN transitions and their associated arcs. A transition is enabled if all of its input places have tokens, and the colors of these tokens can satisfy constraints that are specified on the arcs. A transition can be *fire*d, which means the actions of this transition can occur, when this transition is enabled. When a transition occurs, it consumes all the input tokens as computing parameters, conducts conversation policy and adds new tokens into all of its output places. After a transition occurs, the state (marking) of a protocol has been changed and a protocol will be in terminal state when there is no enabled or fired transition.

There are a number of works of using Petri Nets or CPNs to model agent interaction protocols [15, 16], there have been also some works on the investigation of protocols' flexibility, robustness and extensibility [17]. It is becoming consensus [14] that a CPN is one of the best ways to model agent interaction protocols. However the notion of an agent executing an action with Petri Net is not explicit in the notation [18]. Different PNs can be assigned to each agent role, raising questions about how the entire protocol is inferred and the reachability and consistency of shared places. If a single Petri net is partitioned for each role, this leads to a complex diagram where a partition is required for each agent identified. Furthermore, alternative actions and states such as either agree or reject but not both, cannot be expressed in standard Petri nets.

Keeping in mind complex systems characterized by complex interaction, asynchronism and concurrency we propose to use for the purposes of their description a special language PRALU [6, 7] having its background in the Petri net theory (expanded nets of free choice – EFC–nets investigated by Hack [19]) but possessing special means for keeping track of the current states of the conversation, receiving messages and initiating responses. The formal language PRALU combines properties of "cause-effect" models with Petri nets. It is intended for a wide application in engineering practice and is well suited for representation of the interactions involved in concurrent system, synchronization among them and then it is simple enough for understanding. The language PRALU supports hierarchical description of the algorithms that is especially important in the case of complex systems. At last, powerful software has been developed that provides correctness verifying, simulation, hardware and software PRALU-description's implementation [7, 8]. The review of obtained results it can be found in [8].

PRALU Language

Any algorithm in PRALU consists of sequences of operations to be executed in some pre-determined order. There exist two basic kinds of operations used in PRALU: acting operations "$\rightarrow A$" and waiting operations "$- p$". Action operation changes the state of the object under control, whereas a waiting operation is passive waiting for some event without affecting anything. In simple case A and p are conjunctive terms, so acting and waiting operations can be interpreted as waiting for event $p = 1$ and producing event $A = 1$. But A can be understood too as a formulae defining operations to be performed and p as a predicate defining condition to be verified [20]. For example, acting and waiting operations could be specified by the expressions such as

"$- (a > b + c)$" and "$\rightarrow (a := b + c)$".

The sequences consisting of action and waiting operations are considered to be linear algorithms. For instance, the following expression means: wait for p and execute A, execute B, then wait for q and execute C:

$$-p \rightarrow A \rightarrow B - q \rightarrow C$$

In general, a logical control algorithm can be presented as an unordered set of chains α_j in the form

$$\mu_j : - p_j L_j \rightarrow \nu_j ,$$

where L_j is a linear algorithm, μ_j and ν_j denote the initial and the terminal chain labels represented by some subsets of integers from the set $M = \{1, 2, ..., m\}$: $\mu_j, \nu_j \subset M$ and the expression "$\rightarrow \nu_i$" presents the transition operation: to the chains with labels from ν_j.

Chains can be fulfilled both serially and in parallel. The order in which they should be fulfilled is determined by the variable starting set $N_t \subseteq M$ (its initial value $N_0 = \{1\}$ as a rule): a chain $\alpha_j = \mu_j : - p_j L_j \rightarrow \nu_j$ (that was passive) is activated if $\mu_j \subseteq N_t$ and $p_j = 1$. After executing the operations of the algorithm L_j, N_t gets a new value $N_{t+1} = (N_t \setminus \mu_j) \cup \nu_j$. The algorithm can finish when some terminal value of N is reached (one-element as a rule), at which time all chains became passive. But the algorithms can also be cyclic; they are widely used when describing production processes.

When the conditions $\mu_j \subseteq N_t$ and $p_j = 1$ are satisfied for several chains simultaneously these chains will be fulfilled concurrently. On the contrary chains with the same initial labels are alternative (only one of them can be fulfilled at a time), they are united in a sentence with the same label as will be shown below.

Thus PRALU allows concurrent and alternative branching, as well as merging concurrent and converging alternative branches. These possibilities are illustrated with the following examples of simplified fragments [20]:

Concurrent branching	Merging concurrent branching	Alternative branching	Converging alternative branching
1: ...→2.3	2: ...→4	1: – *a*...→2	2: ...→4
2: ...	3: ...→5	– $\bar{a}$ ...→3	3: ...→4
3: ..	4.5: ...		4: ...

In PRALU there are two syntactic constraints on chains that restrict concurrent and alternative brunching. If some chains are united in the same sentence (they have equal initial labels) they should have orthogonal predicates in the waiting operations opening the chains:

$$(i \neq j)\ \&\ (\mu_i \cap \mu_j \neq \varnothing) \rightarrow (p_i\ \&\ p_j = 0).$$

The other constraint is similar to the corresponding condition specific for extended nets of free choice (Hack [19]):

$$(i \neq j)\ \&\ (\mu_i \cap \mu_j \neq \varnothing) \rightarrow (\mu_i = \mu_j).$$

PRALU language has some more useful properties that come in handy for description of complex interaction protocols.

1. PRALU algorithms can be expressed both in graphical and symbolic forms.

2. PRALU language permits hierarchical descriptions. The two terminal algorithms (having the only terminal label) may be used as blocks (invoked as complex acting operations) in hierarchical algorithms.

3. In PRALU there exist some additional interesting operations that can be useful for description of interaction protocols. Those are suppression operations and some arithmetic operations. The first ones provide response on special events that can take place outside or within control system. Suppression operation ("→∗", "→∗γ", "→′γ") interrupts the execution all concurrently executed chains of the algorithm or those ones mentioned in γ, (or do not mentioned in γ – in the case of "→′γ"). This operation breaks the normal algorithm flow. Among arithmetic operations it ought to be mention timeout operations (waiting for *n* unit times "– *n*") and counting operation that counts event occurrences.

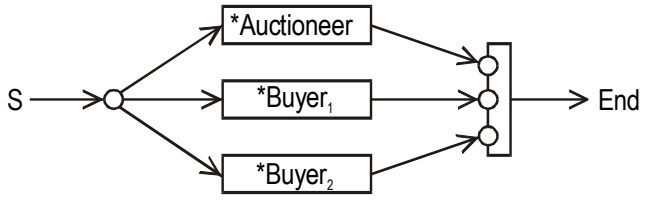
Specifying Protocols in PRALU

For an example of an interaction protocol, consider an English auction [5]. The auctioneer seeks to find the market price of a good by initially proposing a price below that of the supposed market value and then gradually raising the price. Each time the price is announced, the auctioneer waits to see if any buyers will signal their willingness to pay the proposed price. As soon as one buyer indicates that it will accept the price, the auctioneer issues a new call for bids with an incremented price. The auction continues until no buyers are prepared to pay the proposed price, at which point the auction ends. If the last price that was accepted by a buyer exceeds the auctioneer's (privately known) reservation price, the good is sold to that buyer for the agreed price. If the last accepted price is less than the reservation price, the good is not sold.

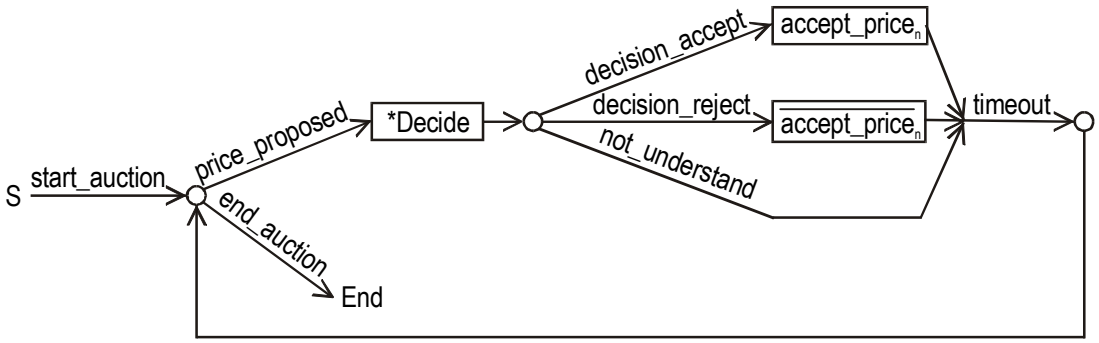
In the case of the auction there are participants of two types: the initiator of the conversation – Auctioneer, and others – Buyers. So, we have two kinds of interaction protocols – those of Auctioneer and of Buyers. The last participants are peer and should be described with identical interaction protocols modeled in our case by the operation "Buyer".

Interaction protocol as a whole can be represented in PRALU as three complex acting operations – blocks. PRALU-blocks are exchanging with values of logical (binary) variables, only such variables are mentioned in them. Each block has some sets of input and output that are enumerated in brackets following the block name (the other variables of a block are its internal). Initialization of a complex acting operation is depicted by the fragment such as "→∗Buyer". The operation Buyer exists in as many copies as the number of participants of the auction, so the copies of the operation differ in their indexes only.

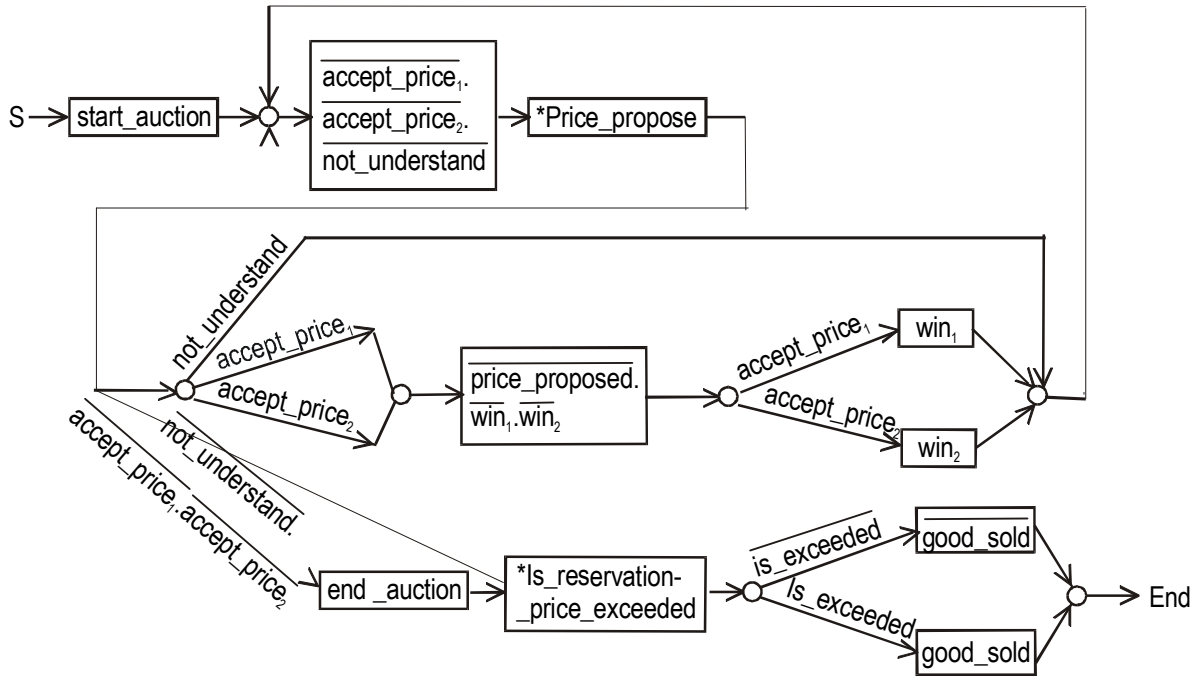
The modelling of the process of auction begins with the execution of "Main_process" triggering event that initiates the interaction protocol execution. Here the processes Auctioneer and Buyer_ns are executed concurrently. For the sake of simplicity we limit the number of buyers to two (in principle it can be simply increased). The process Auctioneer starts with sending the first message (start_auction) that is waited by others participants to continue communication.



Main_process ()



Buyer*n* (start_auction, price_proposed, end_auction / accept_price*n*, not_understand)



Auctioneer (accept_price1, accept_price2, not_understand / start_auction, price_proposed, end_auction)

Fig. 1. English auction interaction protocol in PRALU

Below PRALU description of the auction interaction protocol is shown. Here we cite the only block Buyer*n*, but for real application (intending to simulate the process of auction, for example) we should have as many proper copies as it has been used (in our case – two). Fig. 1 depicts graphical schemes of three mentioned PRALU-blocks: Main_process, Auctioneer and Buyer*n*.

Main_process ()

1: →2.3.4
2: →∗Auctioneer →5
3: →∗$Buyer_1$ →6
4: →∗$Buyer_2$ →7
5.6.7: →**.**

$Buyer_n$ (start_auction, price_proposed, end_auction / $accept\_price_n$, not_understand)

1: –start_auction →2
2: –price_proposed →∗Decide(/decision_accept,decision_reject) →3
–end_auction →**.**
3: –decision_accept → $accept\_price_n$ →4
–decision_reject →′$accept\_price_n$ →4
–not_understand →4
4: –timeout →2

Auctioneer ($accept\_price_1$, $accept\_price_2$, not_understand / start_auction, price_proposed, end_auction)

1: →start_auction →2
2: →′$accept\_price_1$.′$accept\_price_1$.′not_understand →∗Price_propose(/price_proposed) →3
3: –not_understand →2
– $accept\_price_1$ →4
– $accept\_price_2$ →4
–′not_understand.′$accept\_price_1$.′$accept\_price_2$ →end_auction
→∗Is_reservation_price_exceeded(/is_exceeded) →6
4: →′price_proposed.′win_1.′win_2 →5
5: – $accept\_price_1$ →win1 →2
– $accept\_price_2$ →win2 →2
6: –is_exceeded →good_sold →7
–′ is_exceeded →′good_sold →7
7: →**.**

It is assumed that all unformalized operations are referred to as acting operations that set values of logical variables concerned with them. For example, Buyer's operation "Decide" decides for accepting or rejecting the announced price. Depending on adopted decision, it outputs true value of logical variable "decision_accept" or "decision_reject". In a similar, Auctioneers operation "Price_propose" proposes an initial price or increments the charged price outputting true value of logical variable "price_proposed"; the operation "Is_reservation_price_exceeded" verifies if the price accepted by a buyer exceeds the auctioneer's reservation price outputting true or false value of logical variable "is_exceeded".

The operation "–timeout" (where timeout is integer number) means waiting for timeout unit times before doing something followed it. The operation "→**."** is interpreted as the transition to an end of a process described by the block. When the processes of Auctioneer and all Buyers reach their end in the Main_process the transition to its end is executed.

Conclusion

This paper has addressed the need for formalized and more expressive logical and graphical methodologies for specifying (and then validating) interaction protocols in multi-agent systems. Towards this, it was proposed to use the formal language PRALU intended for the representation of complex interactions involved in concurrent system, being in need of synchronization among these interactions. It was demonstrated as well how PRALU algorithms could be used for the specification of multi-agent interaction protocols by the example of English auction. In favour of using the language PRALU is the existence of a great deal of methods and software developed for simulation and logical design of PRALU algorithms as well as for their hardware and software implementation.

References

1. B. Burmeister and K. Sundermeyer, "Cooperative problem-solving guided by intensions and perception", edited by E. Werner and Y. Demazean, Decentralized A.I. 3, Amsterdam, The Netherlands, North Holland, 1992.
2. V. Lesser, "Cooperative Multiagent Systems: A Personal View of the State of the Art", IEEE Trans. Knowledge and Data Engineering, vol. 11, no 1, pp. 133–142, 1999.
3. T. R. Gruber, "A Translation Approach to Portable Ontologies", Knowledge Acquisition, 1993, vol. 5, no 2, pp. 199-220, 1993
4. Finin, Y. Labrou and J. Mayfield, "KQML as an agent communication language", edited by J.M. Bradshaw, Software Agents, MIT Press, pp. 291–316, 1997.
4. ARPA Knowledge Sharing Initiative. Specification of the KQML agent-communication language. ARPA Knowledge Sharing Initiative, External Interfaces Working Group, July 1993.
5. Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org.
6. A.D. Zakrevskij, V.K. Vasiljonok, "The formal description of logical control algorithms for designing discrete systems", Electronnoje modelirovanie, vol. 6, no 4, pp. 79–84, 1984 (in Russian).
7. A.D. Zakrevskij, "Parallel algorithms for logical control", Minsk, Institute of Engineering Cybernetics of NAS of Belarus, 202 p., 1999 (in Russian).
8. L.D. Cheremisinova, "Realization of parallel algorithms for logical control", Minsk, Institute of Engineering Cybernetics of NAS of Belarus, 246 p., 2002 (in Russian).
9. M. Greaves, and J. Bradshaw J., "Specifying and Implementing Conversation Policies", Autonomous Agents '99 Workshop, Seattle, WA, May 1999.
10. G. Booch, J. Rumbaugh, and I. Jacobson, "*The Unified Modelling Language User Guide*", Addison Wesley, 1999.
11. B. Bauer, J.P. Müller, J. Odell, "Agent UML: A Formalism for Specifying Multiagent Interaction," *Agent-Oriented Software Engineering* , edited by P. Ciancarini and M. Wooldridge, Springer-Verlag, Berlin, pp. 91–103, 2001.
12. D. Harel and M. Politi, "Modelling reactive systems with statecharts", McGraw-Hill, 1998.
13. K. Jensen, "Colored Petri Nets – Basic Concepts, Analysis Methods and Practical Use", vol. 1: Basic Concepts, Springer-Verlag, Berlin, 1992.
14. Quan Bai, Minjie Zhang and Khin Than Win, "A Colored Petri Net Based Approach for Multi-agent Interactions", 2nd International Conference on Autonomous Robots and Agents, Palmerston North, New Zealand, December 13-15, pp. 152–157, 2004
15. R. Cost, "Modelling Agent Conversations with Coloured Petri Nets", Working Notes of the Workshop on Specifying and Implementing Conversation Policies, Seattle,Washington, pp. 59–66, 1999.
16. D. Poutakidis, L. Padgham, and M. Winikoff, "Debugging Multi-Agent Systems Using Design Artefacts: The Case of Interaction Protocols", Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi Agent Systems, Bologna, Italy, pp. 960–967, 2002.
17. J. Hutchison and M. Winikoff, "Flexibility and Robustness in Agent Interaction Protocols", Proceedings of the 1st International Workshop on Challenges in Open Agent Systems, Bologna, Italy, 2002.
18. S. Paurobally, J. Cunningham, "Achieving Common Interaction Protocols in Open Agent Environments", *AAMAS* '02, Melbourne, Australia, 2002.
19. M. Hack, "Analysis of production schemata by Petri nets", Project MAC-94, Cambridge, 1972.
20. A. Zakrevskij, V. Sklyarov, "The Specification and Design of Parallel Logical Control Devices", Proceedings of PDPTA'2000, June, Las Vegas, USA, pp. 1635–1641, 2000.

Authors' Information

Dmitry Cheremisinov, **Liudmila Cheremisinova** – The United Institute of Informatics Problems of National Academy of Sciences of Belarus, Surganov str., 6, Minsk, 220012, Belarus, Tel.: (10-375-17) 284-20-76, e-mail: cher@newman.bas-net.by, cld@newman.bas-net.by

ОБ ОДНОЙ МОДИФИКАЦИИ TSS-АЛГОРИТМА

Руслан А. Багрий

Аннотация: *В работе рассматривается одна модификация TSS–алгоритма, который генерирует минимальное порождающее множество решений системы однородных линейных диофантовых уравнений над множеством натуральных чисел.*

Ключевые слова: *Система линейных диофантовых уравнений, базис решений, минимальное порождающее множество.*

Введение

Системы линейных диофантовых уравнений используются во многих разделах современной науки. Важное место среди таких систем занимают системы линейных однородных диофантовых уравнений (СЛОДУ), поскольку методы решения неоднородных систем уравнений или неравенств сводятся к решению СЛОДУ. Методы решения СЛОДУ и систем линейных неоднородных диофантовых уравнений (СЛНДУ) в области натуральных чисел используются в таких областях как искусственный интеллект, логическое программирование[Lloyd J., 1987], компьютерная алгебра[Buchberger B, Kollins J, Loos R., 1986], автоматизация доказательств теорем, унификация, сети Петри [Murata, 1989] и т. п.

Методи решения системы линейных диофантовых уравнений условно можно разделить на методы построения базиса множества всех решений СЛОДУ и методы построения минимального порождающего множества решений СЛОДУ. Известно несколько методов и алгоритмов решения задачи построения базиса и наиболее часто используемым является метод Контенжан-Дэви. В некоторых случаях СЛОДУ может быть охарактеризована с помощью минимального порождающего множества решений, которое в работе [Krivoi S., 1999] было названо TSS (от английского ***Truncated Set of Solutions***).

В данной статье рассматривается одна модификация TSS-метода и соответствующего алгоритма и ее сопоставление с алгоритмом Контенжан-Вэви.

1 Необходимые сведения и определения

Системой линейных диофантовых уравнений (СЛДУ) называется система вида

$$S = \begin{cases} a_{11}x_1 + ... + a_{1q}x_q = b_1; \\ a_{21}x_1 + ... + a_{2q}x_q = b_2; \\ ... \quad ... \quad ... \quad ... \quad ... \\ a_{p1}x_1 + ... + a_{pq}x_q = b_p; \end{cases} \tag{1}$$

где $a_{ij}, b_j \in Z$, $x_j \in N$ и i = 1,2,...,p, j = 1,2,...q.

Единичные вектора из множества N^q $e_1 = (1,0,...,0)$, $e_2 = (0,1,...,0)$,..., $e_q = (0,0,...,1)$ называются векторами канонического базиса множества N^q. Решением СЛДУ (1) называется вектор $c=(c_1,c_2, \dots , c_q)$, где $c_i \in N$, $I = 1,2,...,q$, если выполняется условие $\forall I \in [1,p]$ $a_{i1}c_1 + ... + a_{iq}c_q \equiv b_i$.

Если $\forall I \in [1,p]$ $b_I = 0$, то система называется однородной. СЛОДУ всегда имеет решение вида (0,0,...,0), называемое нулевым или тривиальным. Всякое решение СЛОДУ, отличное от нулевого, называется нетривиальным. СЛОДУ называется совместной, если она имеет хотя бы одно нетривиальное решение. Введем на множестве N^q отношение порядка $\leq$, которое определяется таким образом: если $x=(x_1,x_2, \dots , x_q)$, $y=(y_1,y_2, \dots , y_q) \in N^q$, то $x \leq y$ тогда и только тогда, когда для всех I = 1,2,...,p $x_i \leq y_i$.

Это отношение является частичным порядком и относительно этого порядка множество минимальных решений СЛОДУ является базисом. Известно, что это множество всегда конечно и задача сводится к его построению.

2.1 Краткая характеристика алгоритма Контенжан-Дэви

Метод Контенжан-Дэви[Contenjean E., Devie H., 1994] был первым из алгоритмов построения базиса множества всех решений СЛДОДУ, который не использовал идею сокращения числа неизвестных и уравнений. Этот алгоритм использовал идею метода Фортенбахера[Clausen M. Fortenbacher A., 1989], который строился для построения базиса множества решений единственного диофантового уравнения (ЛОДУ). Идея Фортенбахера состоит в том, чтобы искать минимальные базисные решения, начиная с канонических векторов N^q. При этом если некоторый текущий вектор $x=(x_1,x_2, \dots , x_q)$ еще не является решением, то условие наращивания компонент текущего вектора выполняется с соблюдением такого условия (условие Фортенбахера):

(C_1): увеличивать на 1 то x_j, для которого $a(x)\cdot a(e_j)<0$, где $x = (x_1,x_2, \dots , x_q)$, $a(e_j) = a_j$, где e_j – вектор канонического базиса.

Метод Контенжан-Дави обобщает условие Фортенбахера для СЛОДУ до такого:

(C_p): увеличивать на 1 то x_j, для которого $a(x)\cdot a(e_j)<0$, где $\cdot$ означает скалярное произведение векторов-столбцов $a(x)$ и $a(e_j)$.

Преимуществами алгоритма Контенжан-Дэви есть свойство инкрементальности, возможность применения не только к СЛОДУ, но и СЛНДУ и отсутствие необходимости предварительных расчетов.

Недостатками алгоритма есть его малая эффективность в случае СЛОДУ с большими величинами коэффициентов a_{ij}, а количество уравнений экспоненциально влияет на количество необходимых итераций для поиска решения и как следствие на время решения.

2.2 Краткая характеристика TSS-алгоритма

В ряде задач нет необходимости искать весь базис решений системы, а достаточно убедиться в том, что система совместна. Совместность СЛОДУ можно охарактеризовать с помощью построения ее минимального порождающего множества решений. Минимальное порождающее множество решений генерируется TSS-алгоритмом (описание этого алгоритма можно найти в [Krivoi S., 1999, Krivoi S., 2003]) и по этому множеству осуществляется проверка СЛОДУ на совместность. Работа TSS-алгоритма сводится к выполнению следующих шагов:

- для заданной СЛОДУ $L_1(x) = 0$ & $L_2(x) = 0$ & … & $L_j(x) = 0$, множество канонических векторов N^q разбиваются на три группы M^0, M^+, M^- таким образом, что $M^0 = \{e|L_1(e)=0\}$, $M^+ = \{e|L_1(e)>0\}$, $M^- = \{e|L_1(e)<0\}$; если множество $M^0 \cup M^+$ или $M^0 \cup M^-$ пустое, то уравнение не имеет нетривиальных решений в множестве натуральных чисел, иначе множество решений уравнения $L_1(x) = 0$, находять по следующей формуле:

$$M_1 = M^0 \cup \{y_{ij} \mid y_{ij} = -L_1(e_i)\cdot e_j + L_1(e_j)\cdot e_i, e_j \in M^+, e_i \in M^-\} \quad (2)$$

- полученное множество M_1 с помощью функции $L_2(x)$ тоже разбивается на три группы – $M^0 = \{e|L_2(e)=0\}$, $M^+ = \{e|L_2(e)>0\}$, $M^- = \{e|L_2(e)<0\}$;
- далее повторяются все действия, аналогичные предыдущему уравнению.
- алгоритм заканчивает работу, когда все уравнения данной СЛОДУ будут обработаны и элементы множества M_j составляют TSS этой СЛОДУ.

Основное свойство TSS выражается такой теоремой [6, 7].

Теорема 1. Пусть M и M_j - множество всех решений СЛОДУ и ее TSS, соответственно. Тогда для любого $x \in M$ и $x \notin M_j$ имеет место представление

$$tx = a_1e_1 + a_2e_2 + ... + a_ke_k,$$

где a_i, $t > 1 \in N$, $e_i \in M$.

Из этой теоремы и способа построения TSS вытекает, что TSS совпадает с базисом всего множества решений данной СЛОДУ, если |TSS| = 1.

Из способа построения элементов множества M_j вытекает такое утверждение

Теорема 2. TSS СЛОДУ состоит из единственного решения, если СЛОДУ включает p линейно независимых уравнения, $q - p \leq 1$ и множество $M_i^0 = \varnothing$ для всех $i \in [1,p]$, где q – количество неизвестных.

На эффективность работы данного метода мало влияют значения коэффициентов при неизвестных. Данный метод лишен тех недостатков свойственных предыдущему методу, но в отличие от него не дает полного базиса. Это ограничение позволяет использовать его только в задачах, где поиск всего базиса не требуется.

Таким образом, каждый из алгоритмов, существующих на данный момент времени, может использоваться только в каких-то определенных задачах обусловленных ограничениями самого алгоритма. Следовательно, расширив границы, при которых усеченное множество TSS алгоритма будет совпадать с полным минимальным базисом, мы тем самым, увеличим круг задач, в которых он может использоваться. Рассмотрим одну простую модификацию TSS-алгоритма, который основывается на теореме 2.

3 Модификация TSS-алгоритма

Как следует из вышесказанного, TSS-алгоритм в процессе своей работы оперирует с каждым из уравнений СЛОДУ и тремя множествами M^0, M^+, M (первое множество M^0 уже включает решения очередного уравнения). Комбинирование элементов из последних двух множеств в объединении с первым множеством и дает TSS данной СЛОДУ. Рассмотрим на примере алгоритм работы и проанализируем полученный результат.

Пусть система уравнений будет следующая:

$$\begin{cases} L_1(x) = 4x_1 + 2x_2 - 3x_3 - 2x_4 - x_5 = 0 \\ L_2(x) = 2x_1 - x_2 - 4x_3 + x_4 - 5x_5 = 0 \\ L_3(x) = 0x_1 + 2x_2 + 4x_3 + 0x_4 - 9x_5 = 0 \end{cases} \quad (3)$$

Рассмотрим уравнение $L_1(x)$. Множество канонических векторов N^q в данном случае разбиваются на три группы: $M^0 = \varnothing$, $M^+ = \{(1,0,0,0,0), (0,1,0,0,0)\}$, $M^- = \{(0,0,1,0,0), (0,0,0,1,0), (0,0,0,0,1)\}$. После комбинирования элементов этих множества по формуле (2) получаем такое TSS, состоящее из 6 векторов:

$$e_1 = (3,0,4,0,0), e_2 = (0,3,2,0,0), e_3 = (1,0,0,2,0),$$
$$e_4 = (0,1,0,1,0), e_5 = (1,0,0,0,4), e_6 = (0,1,0,0,2).$$

При анализе полученных результатов очевиден, тот факт, что количество компонент в каждом векторе не равных нулю будет равно двум, поскольку только два вектора из множества канонических векторов комбинировались и формировали решение.

Идея данной модификации состоит в комбинировании не двух векторов принадлежащие к разным множествам M^+ и M^-, а к комбинированию трех векторов, два из которых принадлежат одному множеству, а третий другому. Тогда будем получать решения, в которых количество ненулевых компонент будет больше двух. Это равносильно решению линейного диофантового уравнения с тремя неизвестными, а, следовательно, необходим алгоритм нахождения базиса множества решений такого уравнения, который для данного случая был бы наиболее эффективным.

Для нахождения базиса уравнений вида $ax + by + cz = 0$ был использован расширенный алгоритм Евклида. Этот алгоритм вычисляет наибольший общий делитель двух чисел, а также позволяет решать

уравнения вида $ax+by=c$, где a,b,c – целые, и получить частные решения x_0 и y_0. Общий вид решений имеет вид $x = x_0+kb$, $y = y_0-ka$, где k – произвольное любое целое число.

Исходя из алгоритма Евклида, опишем порядок нахождения минимального базиса для уравнения вида $ax + by + cz = 0$:

1. Преобразуем уравнение по правилу:
   - если $\mathrm{sgn}(a) = \mathrm{sgn}(b)$, то уравнение примет вид $by + cz = -ax$;
   - если $\mathrm{sgn}(a) = \mathrm{sgn}(c)$, то уравнение примет вид $ax + by = -cz$;
   - если $\mathrm{sgn}(b) = \mathrm{sgn}(c)$, то уравнение примет вид $ax + cz = -by$;
2. Примем значение неизвестного в правой части уравнения равной 0.
3. Решим полученное уравнение при помощи алгоритма Евклида.
4. Увеличим неизвестную в правой части уравнения на 1.
5. Будем повторять пункты 3-4 до тех пор, пока значение одного из неизвестных в левой части не будет равно нулю.

Рассмотрим работу алгоритма на примере.

Возьмем два вектора из множества M^+ предыдущего примера – (1,0,0,0,0) и (0,1,0,0,0), один из множества M^- (0,0,1,0,0), то необходимо решить уравнение $4x_1 + 2x_2 - 3x_3 = 0$. Следуя правилу преобразования, приведем уравнения к виду $2x_2 - 3x_3 = -4x_1$. Порядок получения базиса сведен в таблицу:

Уравнение	Вектор-решение (x_1,x_2,x_3)
$2x_2 - 3x_3 = -4 \cdot 0$	(0,3,2)
$2x_2 - 3x_3 = -4 \cdot 1$	(1,1,2)
$2x_2 - 3x_3 = -4 \cdot 2$	(2,2,6)
$2x_2 - 3x_3 = -4 \cdot 3$	(3,0,4)

Третий вектор-решение (2,2,6) не относится к базису, так как все его компоненты больше или равны второго вектора (1,1,2).

Использую расширенный алгоритм TSS, к усеченному множеству решений системы (3) добавляется еще 5 векторов.

$$e_7 = (1,1,2,0,0), e_8 = (2,0,2,1,0), e_9 = (1,0,0,1,2), e_{10} = (0,2,1,0,1), e_{11} = (1,0,1,0,1).$$

В данном случае расширенное усеченное множество векторов совпало с полным базисом решений, но в большинстве случаев такое совпадение редкость.

Дальнейшее решение системы полностью соответствует алгоритму TSS за исключением того, что векторы-решения образуются за новым принципом. Суть его заключается в использовании трех векторов-решений предыдущего уравнения для получения расширенного усеченного базиса. Решение системы уравнений (3) при помощи оригинального алгоритма TSS дает 2 решения:

$$s_1 = (0,63,18,25,22),\ s_2 = (63,0,72,2,32).$$

Используя новый предложенный алгоритм, к ним добавляются еще 4 решения:

$$s_3 = (19,1,22,1,10),\ s_4 = (7,7,10,3,6), s_5 = (13,4,16,2,8),\ s_6 = (1,10,4,4,4).$$

4 Эксперименты

В таблице приведены экспериментальные данные позволяющие оценить временные характеристики одной из версий TSS алгоритма, расширенного TSS алгоритма и метода Контежан-Дави. Также приведено количество полученных ответов решенных этими алгоритмами.

СЛОДУ	Метод TSS		Расширенный метод TSS		Метод Контежан-Дави	
	Кол-во реше-ний	Время мс	Кол-во реше-ний	Время мс	Кол-во реше-ний	Время мс
$\begin{cases} L_1(x) = 4x_1 + 2x_2 - 3x_3 - 2x_4 - x_5 = 0 \\ L_2(x) = 2x_1 - x_2 - 4x_3 + x_4 - 5x_5 = 0 \\ L_3(x) = 0x_1 + 2x_2 + 4x_3 + 0x_4 - 9x_5 = 0 \end{cases}$	2	0	6	15	6	11400
$\begin{cases} L_1(x) = x_1 - 3x_2 + 9x_3 - 1x_4 + 81x_5 = 0 \\ L_2(x) = -9x_1 + 27x_2 + 81x_3 - 9x_4 + x_5 = 0 \\ L_3(x) = 1x_1 - 3x_2 + 9x_3 + 1x_4 - 81x_5 = 0 \end{cases}$	2	0	2	1300	2	2600
$\begin{cases} L_1(x) = -1x_1 + 6x_2 - 3x_3 + 2x_4 + 1x_5 = 0 \\ L_2(x) = 1x_1 + 2x_2 - 5x_3 + 3x_4 + 6x_5 = 0 \\ L_3(x) = 7x_1 + 4x_2 - 2x_3 + 7x_4 - 5x_5 = 0 \end{cases}$	2	0	3	16	3	125
$\begin{cases} L_1(x) = -3x_1 - 14x_2 + 1x_3 + 12x_4 - 3x_5 = 0 \\ L_2(x) = 4x_1 - 5x_2 + 16x_3 + 1x_4 - 4x_5 = 0 \\ L_3(x) = -2x_1 - 1x_2 - 5x_3 - 2x_4 + 14x_5 = 0 \end{cases}$	2	0	66	4700	66	26000

Заключение

Предложенный алгоритм генерирует базис множества всех решений СЛОДУ при условии выполнения требований, приведенных в теореме 2. Кроме того, из экспериментальных результатов следует, что временные характеристики предлагаемого алгоритма выглядят лучше по сравнению с экспериментальными данными алгоритма Контенжан-Дэви.

Библиография

[Lloyd J., 1987] Lloyd J., Foundations of Logic Programming (2-d addition). Springer Verlag. - 1987. -276 p.

[Buchberger B, Kollins J, Loos R., 1986] Компьютерная алгебра (Символьные и алгебраические вычисления). Под ред. Бухберга Б., Коллинза Дж., Лооса Р., М.:Мир, 1986, 386с.

[Contenjean E., Devie H., 1994] Contenjean E., Devie H. An efficient algorithm for solving system of Diophantine equations. Inform. Comput., 1994, 113. v.1, PP.143-173.

[Clausen M. Fortenbacher A., 1989]Clausen M. Fortenbacher A. Efficient solution of linear Diophantine equations. Journ. Symbolic Computation, 1989, v.8, PP. 201-216.

[Murata, 1989] T. Murata. Petri Nets: Properties, Analysis and Applications. In Proceedings of the IEEE, 1989, - vol. 77, -N4, -P. 541-580.

[Krivoi S., 1999] Крывый С.Л. О некоторых методах и критериях совместности систем линейных диофантовых уравнений в области натуральных чисел //Кибернетика и системный анализ., 1999, N 4, с. 12-36.

[Krivoi S., 2003] Кривой С.Л. Об алгоритмах решения систем линейных диофантовых констрейнтов в области {0,1}, .ж. Кибернетика и системный анализ. – 2003, - № 5. – С. 58-69.

Информация об авторе

Руслан А. Багрий (Ruslan A. Bagriy) – Хмельницкий национальный университет, Хмельницкий, Украина (National University of Kchmelnitsk, Kchmelnitsk, Ukraine), e-mail: ruslan@beta.tup.km.ua

THE DEVELOPMENT OF PARALLEL RESOLUTION ALGORITHMS USING THE GRAPH REPRESENTATION

Andrey Averin, Vadim Vagin

Abstract. *The parallel resolution procedures based on graph structures method are presented. OR-, AND- and DCDP- parallel inference on connection graph representation is explored and modifications to these algorithms using heuristic estimation are proposed. The principles for designing these heuristic functions are thoroughly discussed. The colored clause graphs resolution principle is presented. The comparison of efficiency (on the Steamroller problem) is carried out and the results are presented. The parallel unification algorithm used in the parallel inference procedure is briefly outlined in the final part of the paper.*

Keywords: *Automated Reasoning, Logical inference*

1. Introduction

The deductive inference procedures are broadly used in variety of fields, such as expert systems, decision support systems, deductive databases and intelligent information systems. Due to high amount of data in practical problems and the exponential growth of the search space, the efficiency of deductive procedures becomes the key factor in the development of deductive inference systems. One of the ways to improve the efficiency is the parallelization of inference procedures. We present a parallel inference procedure based on resolution principle. The connection graph representation is chosen as the basis for designing the parallel resolution procedures. Using the graph representation simplifies the parallelization of the inference process and allows to apply the different parallelization techniques such as OR, AND and DCDP parallelism. We study and implement OR-, AND- and DCDP- parallel resolution procedures, develop useful heuristics which can be used in parallel resolution procedures on connection graphs and make a comparison of the obtained results with the results of algorithms developed by other researchers. Also we describe and implement the clause graphs inference procedure. As the test task, the Schubert's Steamroller problem is examined [1]. The problem of parallelism on the term level is also investigated. The data structure for the term representation and the parallel unification algorithm using this data structure are presented.

2. Connection Graph

The connection graph method was designed by R. Kowalski [2]. A connection graph is a scheme for representing the proper first-order formulas in disjunctive normal form. Each literal is associated with a node in the connection graph. Literals in a clause are combined into a group. If the literals in two clauses form a contrary pair (P and $\neg$P) then there is an edge between the respective nodes of the connection graph.

Example 1.

The initial set of clauses:

1.Q(c)
2. Q(b)
3.R(x) $\vee \neg$Q(y) $\vee \neg$P(x)
4.P(b)
5.$\neg$R(x) $\vee$ S(x,y) $\vee \neg$T(x)
6.T(y) $\vee \neg$B(y)
7.B(a)
8.$\neg$F(y) $\vee \neg$S(y,z) $\vee \neg$B(z)
9.B(x) $\vee \neg$C(x) $\vee \neg$D(y)
10.D(c)
11.F(b)
12.F(c)
13.C(b)

The corresponding connection graph is shown in fig. 1.

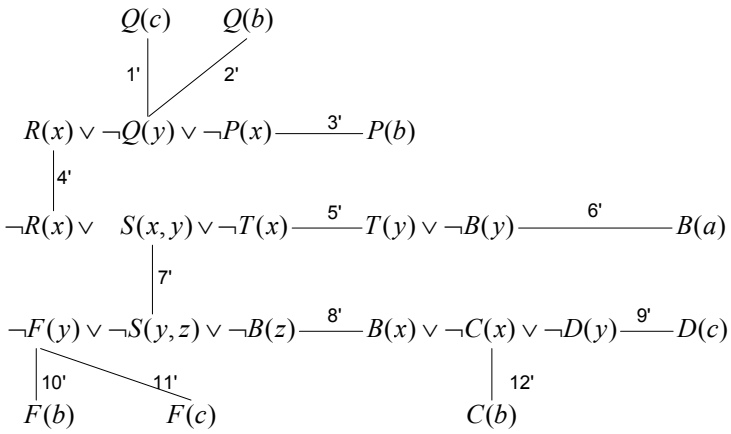


Fig.1

3. Methods of Inference on Connection Graphs

3.1 The Sequential Proof Procedure

To prove the unsatisfiability of a clause set we must generate and resolve the initial connection graph, i.e. derive an empty clause.

The main operation in connection graph refutation is the link resolution, when the resolvent is computed and added to the graph. The corresponding link is deleted and the links of the added resolvent are inserted. A pure clause is a literal group containing a node with no links. Pure clause with all its links can be easily removed from the graph without losing the completeness of the connection graph refutation procedure. Similarly, if we have a tautology clause, it also can be removed from a graph. If a resolvent on some step is a pure clause or a tautology, there is no need to insert this clause into a graph.

The refutation algorithm consists of the following steps:

1. the verification whether there is any clause in a graph or not. If there are no clauses, the algorithm terminates unsuccessfully. If there is the empty clause, then the algorithm is successfully terminated, else go to step 2;
2. if a graph does not contain any link, then the algorithm is unsuccessfully terminated, else go to step 3;
3. a link selection. The link is resolved and the resolvent is generated;
4. if an empty resolvent is obtained, then the algorithm terminates successfully, else the resolvent is inserted into the graph, its links are added, and the algorithm goes to step 2.

The fundamental problem in the connection graph refutation is the choice of suitable links by some criteria at each step of an algorithmic operation. Links are usually selected by using heuristics.

3.2 Parallel Inference on the Kowalski Connection Graph

The Kowalski connection graph can easily be used as the basis for designing parallel resolution algorithms [3]. Since the search space is complete, there is a possibility of using parallel computation strategies for enhancing the inference procedure efficiency. Parallel resolution algorithms differ from the sequential algorithm in step 3 at which a set of links satisfying certain criteria (not a single link as in the sequential procedure) is chosen and parallel resolution of all the links in this set is carried out.

3.2.1. OR-parallel Resolution on a Connection Graph

In case of OR-parallelism, the inference system associates some goal clause with the heads of clauses – possible candidates for resolution. Literals are unified and new clauses are generated. Admissible OR-links sets for the connection graph of Fig.1 are {1', 2'} and {10', 11'}.

3.2.2. DCDP-parallel Resolution on a Connection Graph

One modification of the parallel inference on the Kowalski connection graph is called DCDP parallelism (parallelism for distinct clauses) [4]. The correctness of the DCDP parallel resolution is proved in [5].

Definition 1. Clauses are said to be adjacent if there exist one or several links joining the literals of one clause with the literals of another clause.

Definition 2. A set of links joining pairs of distinct clauses is called a DCDP-link set if the clauses of every pair are not adjacent to any clause of other pairs.

To illustrate these definitions let us study the DCDP-link set for the connection graph of Fig.1. Adjacent pairs of clauses for this set are {(1), (3)}, {(2), (3)}, {(3), (4)}, etc... Thus, one of the DCDP-link sets is {1', 6', 9'}. Other examples of DCDP-link sets are: {2', 6', 12'}, {4', 12'}, {1', 6', 10'}.

3.2.3 AND-parallel Inference

Definition 3. A clause where all its links are resolved in parallel is called a SUN-clause.

Definition 4. Clauses joined with the literals of a SUN-clause are called the satellite clauses.

In AND-parallelism all links of literals of the SUN clause are resolved simultaneously. All resolvents are inserted in the graph along with all inherited links of a satellite clause. A SUN clause with all its links is removed. There is proved the correctness of the AND-parallel resolution [5]. The correct unification of separable variables under AND-parallel resolution is studied in [3].

Let us consider the choice of an AND-link set for the connection graph of fig.1. Admissible AND-link sets are, for example, {5', 4', 7'} (SUN-clause –clause (5)) and {5',6'} (SUN-clause – clause (6)). Detailed description of methods and algorithms can be found in [3,6].

4. Modification of Parallel Inference Procedures

Different heuristics can be used for choosing a link in resolving upon a connection graph. In the parallel resolution we must choose a set of links satisfying certain conditions. Note that the inference procedure becomes unsuitable if links are chosen unsuccessfully. The main principles underlying the design of heuristics are:

1. the number of literals in resolved clauses must be minimal,
2. the number of links in resolved clauses must be minimal,
3. the number of links in a literal for which the clauses are resolved must be minimal,
4. the unifiers of a resolved link must have a substitution of the type {*c*/*x*}, where *c* is a constant or a functional term, and *x* is a variable.

Further we describe the meaning of each principle in more detail.

Principle (1) simplifies a resulting connection graph, because a clause with a small number of literals, usually has a fewer number of links.

Principle (2) also simplifies a resulting connection graph, because the resolution of clauses with a small number of links yields clauses also with a small number of links.

Principle (3) prefers those links, the resolution of which yields "pure" clauses that on removing, can considerably simplify a connection graph.

The same is true for principle (4): as a result of the resolution by links containing a substitution of a variable for a constant, we obtain a clause containing constant terms. Such a clause has, at the first, a small number of links and, at the second, can be effectively used in the resolution. Principles (I)-(4) are taken into consideration in the heuristic function described below.

4.1. The Heuristic Function HI

In the heuristic function *H1* the link estimation is represented as a linear combination of estimations of the objects in the link (unifiers, clauses, and predicate literals in the link).

4.1.1 Computation of the Value of the Heuristic Function

Let *WeightLink* denote the heuristic estimation of a link. Then:

$$WeightLink = k_{clause} \cdot (Clause_{1Heur} + Clause_{2Heur}) + Uni_{Heur} \cdot k_{uni} + k_{pred} \cdot (Pred_{1Heur} + Pred_{2Heur}),$$

where $Clause_{1Heur}$, $Clause_{2Heur}$, Uni_{Heur}, $Pred_{1Heur}$ and $Pred_{2Heur}$ are the heuristic estimations for the first clause, the second clause, the link unifier, the predicate literal in the first clause of a link, and the predicate literal in the second clause of a link, respectively, and k_{uni}, k_{clause} and $k_{pred} \in [1; 100]$ are coefficients. Let us describe these symbols in more detail.

4.1.2 Heuristic Estimation of a Clause

The heuristic estimation must take into account the changes taking place in the characteristics of a connection graph during the inference (for example, changes in the number of links and the number of literals in a clause). Let us examine the heuristic estimation based on principles (1) and (2):

$$Clause_{Heur} = k_1 \cdot ClauseNumberOfLinks + k_2 \cdot ClauseNumberOfClauses,$$

where *ClauseNumberOfLinks* is the number of links in a clause, *ClauseNumberOfClauses* is the number of predicate literals in a clause; k_1, k_2 are arbitrary coefficients, chosen a priori on the basis of graph characteristics such as an average number of literals and links for the clause.

Let for example the average number of literals and links for the clause be 3.2 and 4.8, respectively. Then we can take k_1 = 4.8/3.2 = 1.5 and k_2= 1. In this case, both principles (1) and (2) have the same weight. Characteristics may change their values during the inference. In this case, the initially chosen values of coefficients become "obsolete," and one principle gains a greater weight over the other. To avoid such a situation, the values of coefficients must be changed during the inference. Let *AverageLinkCount* and *AverageClauseLength* denote an average number of links and literals in a clause, respectively.

We can take k_1 = *AverageLinkCount/AverageClauseLength* and k_2=*1*. In this case, principles (1) and (2) both gain the same weight in the course of the whole inference process. The heuristic clause estimation takes the final form:

$$Clause_{Heur}=(AverageLinkCount/AverageClauseLength)\times ClauseNumberOfLinks+ ClauseNumberOfClauses.$$

4.1.3 Heuristic Estimation of the Unifier

The heuristic unifier estimation must be based on principle 4 (the unifier of a resolved link must have a substitution of the type *c/x*, where *c* is a constant or a functional term with constants and *x* is a variable) and must take into account the changes in values of the graph characteristics. The heuristic estimation of the unifier Uni_{Heur} is computed by the formula:

$$Uni_{heur}=AverageLinkCount/(1+NumberOfConstantSubst +NumberOfFuncSubst),$$

where *NumberOfConstantsSubst* is the number of substitutions of the type *{c/x}* in the unifier, *c* is a constant term and *x* is a variable.

NumberOfFuncSubst is the number of substitutions of the type {*f/x*} m the unifier, where *f* is a functional term with constants and *x* is a variable.

4.1.4 Heuristic Estimation of a Predicate Literal

The heuristic estimation of a predicate literal must be based on principle (3) (the number of links in a literal, for which clauses are resolved upon must be minimal). The heuristic function must also take into account the changes in the graph characteristics. The heuristic estimation $Pred_{Heur}$ of a predicate literal is computed by the formula:

$$Pred_{Heur}=PredNumberOfLinks\cdot AverageClauseLength,$$

where *PredNumberOf Links* is the number of links in a predicate literal.

4.1.5 Selection of Coefficients

The coefficients k_{uni}, k_{clause} and k_{pred} are chosen experimentally. We have chosen the following ratio $k_{uni}=k_{pred}=(1/10)\cdot k_{clause}$ for coefficients. In this relationship, the greater weight is attached to principles (1) and (2). As a rule, k_{clause} is taken from the interval [10; 100] so that k_{uni} and k_{pred} lie in the interval [1; 10]. Thus, the heuristic function *H1* takes account of all principles (I)-(4). The weight of principles can be changed. The changes in the graph characteristics are also taken into account. The function *H1* enhances the efficiency of parallel inference algorithms for problems of practical complexity.

5. Deductive Inference on Clause Graphs

The deduction algorithm transforms a clause graph via two special operators – a predicate node elimination operator and a predicate node splitting operator [6,7]. They are applied to a predicate vertex depending on whether the node has multiarcs or not. A predicate node I said to be joined to a clause with multiarcs if the clause contains more than one literal with predicate symbol of the predicate node. For example, the node P in Fig.2 is joined to clause 1,2 and 3 (clause 4 is joined with the predicate node P with an ordinary arc) by the multiarcs.

1. $P(x,y) \rightarrow P(f(x),y)$ $(or \quad \neg P(x,y) \vee P(f(x),y))$
2. $P(u,f(x)) \,\&\, P(v,g(w)) \rightarrow$ $(or \quad \neg P(u,f(x)) \vee \neg P(v,g(w)))$
3. $\rightarrow P(g(x),y) \vee P(a,z)$ $(\text{or } P(g(x),y) \vee P(a,z))$
4. $\rightarrow P(a,x)$ $(or \quad P(a,x))$

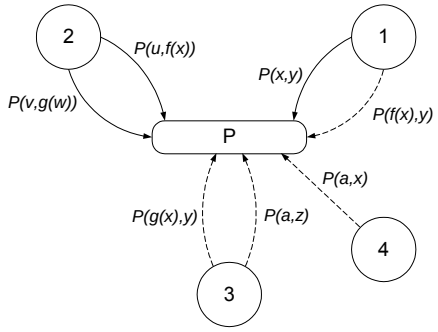


Fig. 2

In this method clauses are expressed in implicative form.

Here 1-4 are the nodes corresponding to the clauses of the logical model. The oval vertex represents the predicate symbol P. Continuous arcs are weighted with conditional literals of clauses, whereas dotted ars are weighted with inference literals of the clause.

An operation similar to colouring of clause graphs is introduced. Clause condition is «network is coloured» by color C1 (the corresponding arcs in figures are shown by a continuous line) and the condition for inference is colored by color C2 (a dotted line in figures). If clause graphs are represented in colored form, then it is easy to search for information and new assertions can be inferred easily and thus the effectiveness of the inference system is enhanced.

The general inference scheme for an empty clause via transformation of clause graphs consists of the following:

1. if the network contains predicate nodes to which the node elimination operator can be applied, then such nodes are removed by this elimination operator;
2. if there are nodes with multiarcs, then the splitting operator is applied to generate nodes without multiarcs. Then the node elimination operator is applied, etc. until a contradiction is obtained in network.

If the node elimination operator consists of a set of usual operations of resolution of specially chosen pairs of clauses, then the splitting operator contains a distinct feature of this algorithm, i.e., a feature that has no analog in other logical inference mechanisms.

The predicate node elimination operator is applied to nodes having no multiarcs. It resolves in all possible ways those clauses that contain contrary pairs of literals. Resolution is implemented by the predicate of the chosen predicate node. Upon completion of all resolutions, parent clauses and the predicate node are removed from the network and the new clauses found via resolution are included in the network. Figures 3a and 3b show a clause graph before and after the application of the elimination operator to the vertex M for the following disjunct set:

1. $Q \,\&\, M \rightarrow$
2. $M \,\&\, H \rightarrow Q$
3. $F \,\&\, M \rightarrow H$
4. $\rightarrow M$
5. $\rightarrow F$

The node M and clauses (1)-(4) were removed and the clauses

6. $Q \rightarrow .$ (1,4)
7. $F \rightarrow .H$ (3,4)
8. $H \rightarrow Q$ (2,4)

were added to the network.

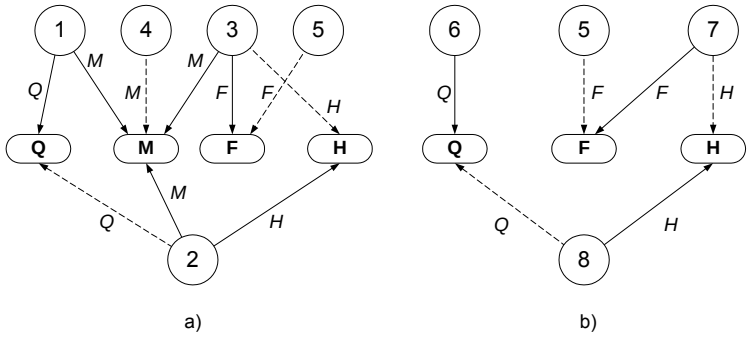


Fig. 3

The predicate splitting operator is introduced in order to remove multiarcs from nodes and thereby create conditions for the application of the elimination operator. The splitting operator generates copies of clauses and a few new vertices having the same predicate symbol as the split vertex, but with new indexes 1,2,3, … . The clause copies and new nodes are interrelated with each other. First, the elimination operator removes those nodes that have higher indexes.

Figure 4 shows the action of a splitting operator for the case of three multiarcs starting from clauses 1,2 and 3 (Fig. 2).

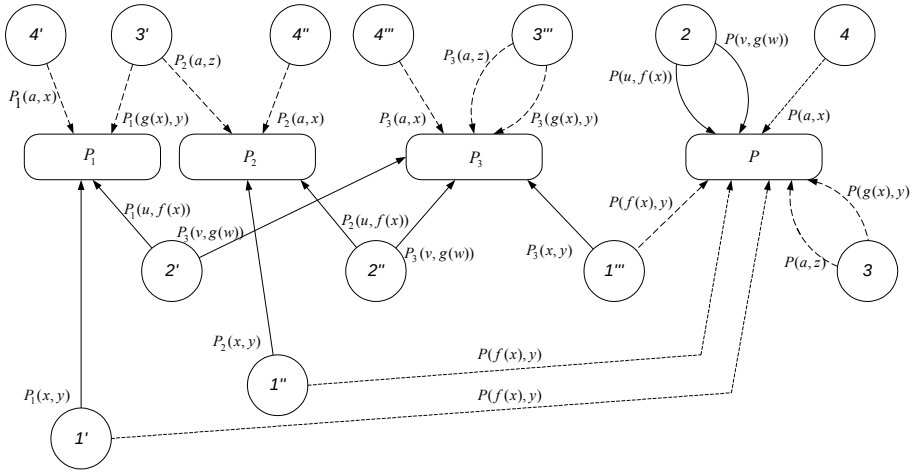


Fig.4

Splitting the three multiarcs (“petals”) of the node P by this operator 1we obtain four nodes P_1, P_2, P_3 and P (their amount is one more than the amount of «petals»), four copies of clause 4, three copies of the «split» clause 1, two copies of the «split» clause 2 with a duplicate at vertex P, and a «split» clause 3 with a duplicate at nodes P and P_3. Splitting occurs in the following order: first the multiarc (1 – P), then the multiarc (2 – P), and finally the multiarc (3 – P) are split. The priority elimination strategies for removing nodes having no multiarcs and nodes with minimal number of arcs are reasonable for forming a sequence of processed nodes in a clause graph. They yield a sequence of eliminations and generate a lesser number of intermediate clauses than the variant in which these strategies are not used. Both these operations are correct i.e. unsatisfiable clauses remain unsatisfiable after the application of these operators[1]. A parallel algorithm for deductive inference on colored clause graphs is described in [7,8].

5. Efficiency

As the test problem, we have researched the "steamroller" problem formulated by L. Schubert in 1978 to test automatic proof systems [1]. This problem requires the generation of an exponential number of intermediate clauses and unifications during the inference process. Below we describe the results obtained by our and other algorithms of deductive inference for the «steamroller» problem.

CG is the inference strategy with a connection graph. SOS is the inference on a connection graph with a goal statement as a support set. TR is the inference within Theory Links - the extension of the standard resolution method. UR is inference with Unit Resolution - a modification of the resolution method. LUR is the inference by Linked Unit Resolution - a modification of the UR resolution method.

According to Figs. 5 and 6, the best results are obtained by the McCune/LUR procedures. Parallel inference procedures have also shown their efficiency for deductive inference on connection graphs.

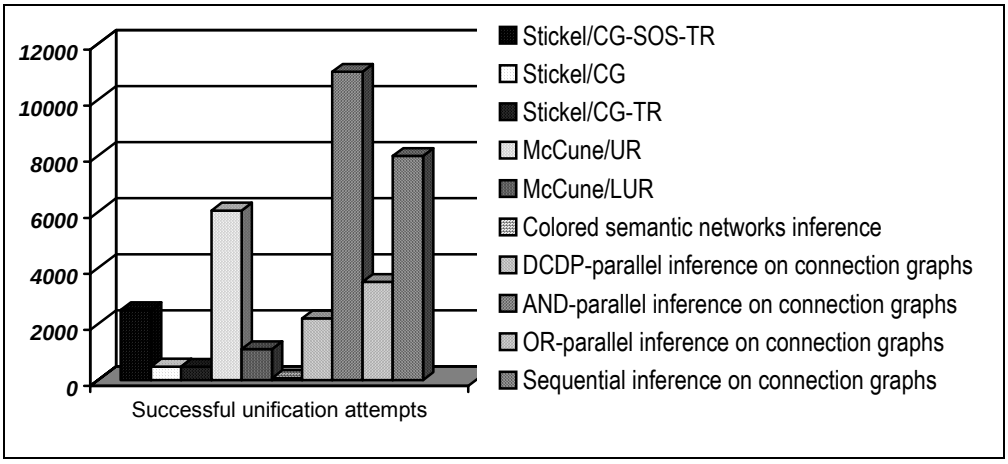


Fig. 5

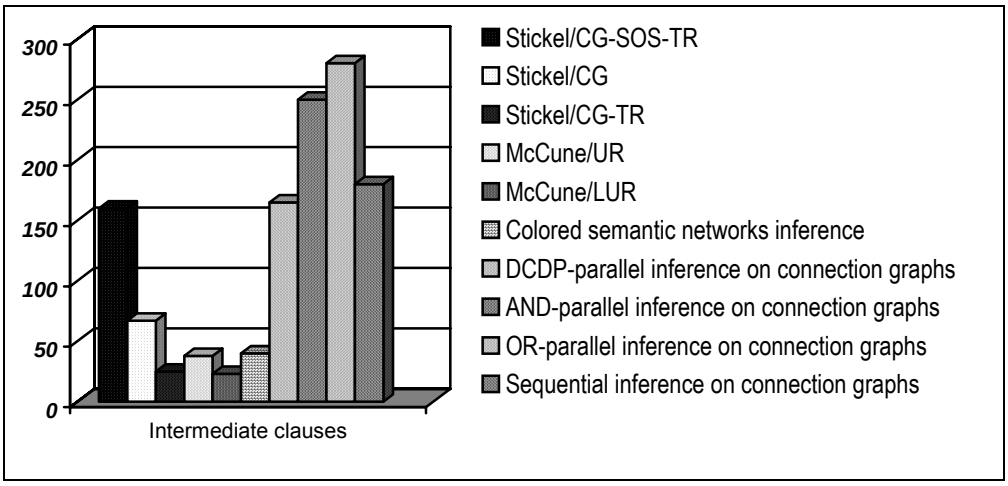


Fig. 6.

6. Parallel Unification in Connection Graph Inference

Due to a high amount of the unification tasks in deductive inference, the efficiency of the implementation of the unification procedure is one of the main factors in designing deductive inference procedures. There is a variety of effective unification procedures, but all of them have their own disadvantages. The main disadvantage is the use of complex, tightly coupled data structures, which are very difficult to parallelize.

We develop rather simple data structure for the term representation based on the notion of path strings. Terms (and clauses) are stored in tables, which are connected with links. Some tables and parts of the tables can be processed in parallel (with having links in mind). As we have no possibility to describe the algorithm and the data structure for term representation in detail, we just briefly outline the main principles used in this approach.

The term is stored as the sequence of strings, where every string presents one symbol in the term. Also such information as the type of a symbol (a variable or a constant symbol), the index number of an argument in a function symbol and the depth of a symbol (i.e. the number of function symbols which this symbol belongs to) is stored. The terms that can be unified are connected with the links of two types.

The first type corresponds to the terms (and strings) of different depth. The second type corresponds to the terms of the same depth. Let us illustrate these notions by the simple example.

Consider the unification task $\{t_1=t_2\}$, where t_1= *f(a,g(h(x)))* and t_2=*f(x,g(y))*. The representation of the term *f(a,g(h(x)))* is *f1.a.1.constant*(1) and *f2.g1.h1.x.3.variable* (2).

The representation of the term *f(x,g(y))* is *f1.x.1.variable*(3) and *f2.g1.x.2.variable*(4) (the index number is shown in brackets). The links are created between the strings (1) and (3) and between the strings (2) and (4). The first link has the second type and the second link has the first type.

The task of link establishing is one of the main tasks in the proposed approach. Two techniques can be used. The simplest one is the special sorting procedure on tables containing strings from the unification task. The main drawback of this approach is the deficit of efficiency caused by the nature of the sorting problem (the complexity of the sorting problem is *nlog(n)*). The second approach is based on automata representation of strings and is similar to discrimination trees. Using automata increases the efficiency of the procedure, but requires higher memory consumption.

The main idea lying in the parallelization of the unification procedure is the use of dependencies graph, where strings connected with arcs cannot be proceeded in parallel. The complexity of the task of determining maximum sets for parallel proceeding is equal to the complexity of the graph coloring task, though heuristics can be used to find non-maximum but satisfactory sets. This unification procedure has been combined with the parallel inference procedure on connection graphs. The structure for term representation and unification are thoroughly described in [9].

7. Conclusions

The Kowalski connection graph and the sequential inference procedure on connection graphs are investigated. The structure of the OR-, AND- and DCDP-parallel inference methods is described. Methods of modifying parallel inference procedures are analyzed and the main principles underlying the design of heuristic link estimations are stated. The heuristic function for link selection is designed. A procedure of inference on colored clause graphs is also described. The developed deductive inference procedures are compared with other procedures on the "steamroller" problem. The procedures of parallel inference on connection graphs and clause graphs are the effective ones for the deductive inference. They can be applied in expert and decision making systems.

Bibliography

[1] M.F. Stickel, Schubert's Steamroller Problem: Formulations and Solutions // *J. Autom. Reasoning,* 1986, vol. 2, pp. 89-100.

[2] R. Kowalski, A Proof Procedure using Connection Graphs // J. ACM, 1975, 22(4), pp. 595-599.

[3] V.N. Vagin and N.O. Salapina, A System of Parallel Inference with the Use of a Connection Graph // *Journal of Computer and System Sciences International,* Vol. 37, No. 5, 1998, pp. 699-708.

[4] G. Hornung, A. Knapp and U. Knapp, A Parallel Connection Graph Proof Procedure // *German Workshop on Artificial Intelligence. Lecture Notes in Computer Science,* Berlin: Springer-Verlag, 1981. pp. 160-167.

[5] R. Loganantharaj, Theoretical and Implemetational Aspects of Parallel Link Resolution in Connection Graphs, *Ph.D. Thesis,* Dept. of Computer Science, Colorado State Univ., 1985.

[6] Vagin V.N. Deduktsiya i obobshcenie v sistemakh prinyatia reshenii (Deduction and Generalization in Decision Making Systems), Moscow: Nauka, 1988.

[7] Vagin V.N., Parallel Inference on Logical Networks, IFIP/WG 12.3 International Workshop on Automated Reasoning, Beijing, China, 1992, pp. 305-310.

[8] A.I. Averin, V.N. Vagin and M.K. Khamidulov, The Investigation of Algorithms of Parallel Inference by the Steamroller Problem // *Journal of Computer and System Sciences International,* Vol. 39, No. 5, 2000, pp. 758-766.

[9] A.I. Averin, V.N. Vagin, Using Parallelism in Deductive Inference, // Journal of Computer and System Sciences International. -2004. - vol. 45, №4. – pp.603-615.

[10] Vagin V.N., Golovina E.Y., Zagoryanzkaya A.A., Fomina M.V.. «Dostoverniy i pravdopodobnyi vivod v intellektualnykh systemakh». (Relaible and Plausible Inference in Intellectual Systems) – Moscow: Fizmatlit. - 2004.

Authors' Information

Andrey Averin – e-mail: averin@rbcmail.ru

Vadim Vagin – e-mail: vagin@apmsun.mpei.ac.ru
Moscow Power Engineering Institute, Krasnokasarmennaya str., 14, Moscow, Russia;

МАГНИТНАЯ ГИДРОДИНАМИКА ЖИДКОСТИ И ДИНАМИКА УПРУГИХ ТЕЛ: МОДЕЛИРОВАНИЕ В СРЕДЕ MATHEMATICA

Ю.Г. Лега, В.В. Мельник, Т.И. Бурцева, А.Н. Папуша

Аннотация: *В статье предложен метод компьютерного моделирования и численного анализа нелинейных взаимодействий в механике сплошных сред, в которых учитываются электромагнитные эффекты при помощи компьютерной алгебры среды Mathematica.*

Введение

В настоящей работе представлены результаты компьютерного моделирования и численного анализа нелинейных взаимодействий в механике сплошных сред, в которых учитываются электромагнитные эффекты. Для этого применяются методы символьной алгебры и компьютерных аналитических преобразований с последующим численным моделированием. Основное внимание уделяется сложным моделям взаимодействия упругих тел с токопроводящей жидкостью, принятой в магнитной гидромеханике.

Также как и в магнитной гидромеханике принимается, что сплошная среда является жидкостью или газом, в котором отсутствует поляризация и намагниченность, но может протекать электрический ток.

В отличие от известных работ [1-3] по математическим моделям механики сплошных сред, в которых подобные вопросы уже обсуждались и в которых приведены некоторые решения взаимодействия упругих тел с жидкостью с учетом электромагнитных эффектов, в настоящей работе все исследование процессов взаимодействия начиная от постановки задачи до ее решения выполнены в среде Matematica методами компьютерной алгебры.

Основное преимущество сформулированного выше подхода в применения компьютерной среды Matematica состоит в том, что современные компьютерные технологии позволяют строить инвариантные символьные модули для вывода и формулировки задач механики сплошных сред, с таким расчетом, что бы в последующем реализовать простой принцип исследования: то что Matematica сформулировала, то Matematica и решила.

Разумеется это глобальная задача не может быть решена в конечном виде для большинства проблем механики сплошных сред современными компьютерными средами, но начальные инвариантные модули для вывода фундаментальных уравнений взаимодействия в электродинамике, механике, физике можно построить уже сейчас. По-видимому, в будущем методами компьютерной алгебры подобные задачи будут решатся в конечном виде и предложенные в настоящем докладе подходы послужат стимулом к созданию подобных прикладных пакетов в среде Matematica для решения различных прикладных проблем нелинейной динамики деформируемых тел.

В первой части работы сформулированы и построены инвариантные модули для задачи взаимодействия упругой цилиндрической оболочки с учетом электромагнитных эффектов нелинейного взаимодействия жидкости и оболочки.

Постановка контактной задачи магнитной гидроупругости цилиндрической оболочки.

Движение жидкости в оболочке, помещенной в электромагнитное поле будет описываться совместной системой уравнений движения, которая включает в себя:

- уравнения движения жидкости,
- уравнения движения оболочки,
- и электромагнитного поля

$$\frac{D}{h}n^2 n^2 w = \frac{n^2 w}{nx^2}\frac{n^2 \phi}{nn^2} + \frac{n^2 w}{ny^2}\frac{n^2 \phi}{nx^2} - 2\frac{n^2 w}{nxnn}\frac{n^2 \phi}{nxnn} + \frac{1}{R}\frac{n^2 \phi}{nx^2} + n_0 h\frac{n^2 w}{nt^2} + \frac{1}{h}(P + P_e);$$

$$\frac{1}{E}n^2 n^2 \phi = \frac{n^2 w}{nxnn} - \frac{n^2 w}{nx^2}\frac{n^2 w}{nn^2} - \frac{1}{R}\frac{n^2 w}{nx^2},$$

$$P = -n_0 \frac{n\phi}{nt};$$

$$P_e = n_e\left[\vec{E} + \frac{1}{c}\left(\frac{nw}{nt}\vec{n} \times \vec{H}\right)\right] + \frac{1}{c}\left(\vec{J}^* \times \vec{H}\right);$$

$$\vec{J}^* = \vec{J} + n_e \frac{nw}{nt}\vec{n}$$

$$\vec{v} = \vec{n}n .$$

где в формулах введены обозначения:

- n_0 - плотность материала оболочки,
- nf - плотность жидкости,
- n - потенциал скоростей жидкости,
- P_e - давление на оболочку от взаимодействия с электромагнитным полем.

Кинематические условия совместного движения жидкости и оболочки имеют вид

$$\frac{nw}{nt} = \frac{n\phi}{nn}.$$

В качестве граничного условия на бесконечности принимается равенство нулю возмущенного движения жидкости.

Начальное условие для принимаются нулевыми

$$n = \frac{n\phi}{nt} = 0.$$

Граничное условие для оболочки имеют вид

$$N_n^m(w) = 0$$

где N_n^m - некоторые дифференциальные операторы на граничных линиях срединной поверхности.

Инвариантные модули для вывода уравнений движения жидкости в декартовых координатах в среде Mathematica.

Уравнения движения в цилиндрических координатах.

Решение первых двух уравнений Максвелла в цилиндрических координатах.

Как известно уравнения Максвелла имеют очевидные решения, которые записываются через скалярный и векторные потенциалы. Задача состоит в том чтобы представить эти решения в виде символьных вычислений в среде Mathematica с тем, чтобы в последующем применять для решения некоторых конкретных задач магнитной электродинамики.

Таким образом электрическое поле можно выразить следующим образом

$$Electric = -Grad\left[n@@Append\left[\vec{r},t\right]\right] - \P_t\, Table\left[A_i @@ Append\left[\vec{r},t\right],\{i,n\}\right]$$

$$\{-A_1^{(0,0,0,1)}(Rr,Ttheta,Zz,t) - n^{(1,0,0,0)}(Rr,Ttheta,Zz,t),$$

$$-A_2^{(0,0,0,1)}(Rr,Ttheta,Zz,t) - \frac{n^{(0,1,0,0)}(Rr,Ttheta,Zz,t)}{Rr},$$

$$-A_3^{(0,0,0,1)}(Rr,Ttheta,Zz,t) - n^{(0,0,1,0)}(Rr,Ttheta,Zz,t)\}$$

Решение первого уравнения Максвелла, которое показывает что вектор магнитного поля есть ротор некоторого вектора.

В данном случае полагаем, что магнитное поле есть ротор вектора - $\vec{A}$.

$$Magnetic = Curl\left[Table\left[A_i \text{ @@ } Append\left[\vec{r}, t\right], \{i, n\}\right]\right]$$

$$\{\frac{1}{Rr}(A_3^{(0,1,0,0)}(Rr, Ttheta, Zz, t) - RrA_2^{(1,0,0,0)}(Rr, Ttheta, Zz, t)),$$
$$A_1^{(0,0,1,0)}(Rr, Ttheta, Zz, t) - A_3^{(1,0,0,0)}(Rr, Ttheta, Zz, t),$$
$$\frac{1}{Rr}(A_2(Rr, Ttheta, Zz, t) - A_1^{(0,1,0,0)}(Rr, Ttheta, Zz, t) + RrA_2^{(1,0,0,0)}(Rr, Ttheta, Zz, t))\}$$

Второе уравнение Максвелла системы записывается в виде

$$Div\left(\vec{H}\right) = 0$$

Решение этого уравнения в среде Mathematica представлено ниже в операторе solutionMagnetic

$$solutionMagnetic =$$
$$\left(Div\left[Magnetic\right] // Expand\right) == 0$$
$$True$$

Как видно из вывода второе уравнение векторным потенциалом $\vec{A}$ тождественно удовлетворяется, а решение выполнено в символьном виде в цилиндрических координатах.

Далее представим символьное решение в потенциалах для первого уравнения Максвелла представленного ранее в системе

$$\vec{n} \times \vec{E} = -\frac{1}{c}\frac{\partial \vec{H}}{\partial t},$$

Решение векторного уравнения через соответствующие потенциалы представлено ниже в кодах среды Mathematica

$$solutionMagneticElectric =$$
$$\left(Div\left[Curl\left[Electic\right] + 1/c\partial_t Magnetic\right] // ExpandAll\right) == 0$$
$$True$$

Как видно из вывода первое уравнение системы, также тождественно удовлетворяется векторным потенциалом $\vec{A}$, а его решение выполнено в символьном виде в цилиндрических координатах.

$$solutionMagnetic$$
$$True$$

Преобразование двух других уравнений Максвелла

Далее преобразуем оставшиеся уравнения Максвелла, с учетом введенных ранее полученных решений через скалярные и векторные потенциалы.

Целью дальнейших преобразований становится выбор такого потенциала для дивергенции $\vec{A}$, чтобы уравнения для векторного и скалярного потенциалов разделились.

$$densityElectric = n_e \text{ @@ } Append\left[\vec{r}, t\right]$$
$$n_e\left(Rr, Ttheta, Zz, t\right)$$

Выбор в произволе выбора дивергенции - $\vec{A}$ приводит к уравнению вида

$$eqSupport = -\left(Div\left[Table\left[A_i \text{ @@ } Append\left[\vec{r}, t\right], \{i, n\}\right]\right] // ExpandAll\right)$$

$$-\frac{A_1(Rr,Ttheta,Zz,t)}{Rr}-A_3^{(0,0,1,0)}(Rr,Ttheta,Zz,t)-$$
$$\frac{A_2^{(0,1,0,0)}(Rr,Ttheta,Zz,t)}{Rr}-A_1^{(1,0,0,0)}(Rr,Ttheta,Zz,t)$$
$$sub=\partial_t eqSupport->1/c\wedge 2\partial_{t,t} n@@Append\left[\vec{r},t\right]$$
$$-\frac{A_1^{(0,0,0,1)}(Rr,Ttheta,Zz,t)}{Rr}-A_3^{(0,0,1,1)}(Rr,Ttheta,Zz,t)-$$
$$\frac{A_2^{(0,1,0,1)}(Rr,Ttheta,Zz,t)}{Rr}-A_1^{(1,0,0,1)}(Rr,Ttheta,Zz,t)\rightarrow$$
$$\frac{n^{(0,0,0,2)}(Rr,Ttheta,Zz,t)}{c^2}$$

Уравнение для скалярного потенциала - представлено ниже.

$$eq4=(((Div\left[Electric\right]//Expand)==$$
$$4PidensityElectric)/.sub)$$
$$\frac{n^{(0,0,0,2)}(Rr,Ttheta,Zz,t)}{c^2}-$$
$$n^{(0,0,2,0)}(Rr,Ttheta,Zz,t)-\frac{n^{(0,2,0,0)}(Rr,Ttheta,Zz,t)}{Rr^2}-$$
$$\frac{n^{(1,0,0,0)}(Rr,Ttheta,Zz,t)}{Rr}-n^{(2,0,0,0)}(Rr,Ttheta,Zz,t)==$$
$$4nn_e(Rr,Ttheta,Zz,t)$$

Очевидно, что для скалярного потенциала имеем волновое неоднородное уравнение, решение которого в символьном виде для пространственного случая Mathematica пока не предоставляет. Вместе с тем, его вывод из инвариантных символьных преобразований проведенных ранее дает основу для дальнейшего исследования при развитии новых компьютерных преобразований.

Решение для скалярного потенциала в случае однородного поля имеет вид

$$eq4=(((Div\left[Electric\right]//Expand)==0)/.sub)$$
$$\frac{n^{(0,0,0,2)}(Rr,Ttheta,Zz,t)}{c^2}-$$
$$n^{(0,0,2,0)}(Rr,Ttheta,Zz,t)-\frac{n^{(0,2,0,0)}(Rr,Ttheta,Zz,t)}{Rr^2}-$$
$$\frac{n^{(1,0,0,0)}(Rr,Ttheta,Zz,t)}{Rr}-n^{(2,0,0,0)}(Rr,Ttheta,Zz,t)==0$$

Составление уравнения для векторного потенциала - $\vec{A}$

$$nobo=Table[\partial_t Grad[\Phi@@Append[\vec{r},t]][[k]]->$$
$$c\wedge 2Grad[Div[Table[A_i@@Append[\vec{r},t],\{i,n\}]]][[k]],$$
$$\{k,n\}]//ExpandAll$$
$$eq3=(c\wedge 2Curl\left[Magnetic\right]-4Pi/cCurrent-\partial_t Electric)/ExpandAll;$$
$$equationVectorPotencial1=$$
$$ReplacePart[eq3[[1]],nobo[[1,2]],7]==0$$

$$-\frac{A_1(Rr,Ttheta,Zz,t)c^2}{Rr^2}-A_1^{(0,0,2,0)}(Rr,Ttheta,Zz,t)c^2-$$
$$\frac{A_1^{(0,2,0,0)}(Rr,Ttheta,Zz,t)c^2}{Rr^2}+\frac{A_1^{(1,0,0,0)}(Rr,Ttheta,Zz,t)c^2}{Rr}+$$
$$2A_3^{(1,0,1,0)}(Rr,Ttheta,Zz,t)c^2+\frac{2A_2^{(1,1,0,0)}(Rr,Ttheta,Zz,t)c^2}{Rr}+$$
$$A_1^{(2,0,0,0)}(Rr,Ttheta,Zz,t)c^2+A_1^{(0,0,0,2)}(Rr,Ttheta,Zz,t)-$$
$$\frac{4\pi J_1(Rr,Zz,t)}{c}-\frac{4\pi\ {}_e v_1(Rr,Zz,t)}{c}==0$$

$$equationVectorPotencial2=$$
$$ReplacePart[eq3[[2]],nobo[[2,2]],7]==0$$
$$\frac{A_2(Rr,Ttheta,Zz,t)c^2}{Rr^2}-A_2^{(0,0,2,0)}(Rr,Ttheta,Zz,t)c^2+$$
$$\frac{2A_3^{(0,1,1,0)}(Rr,Ttheta,Zz,t)c^2}{Rr}+\frac{A_2^{(0,2,0,0)}(Rr,Ttheta,Zz,t)c^2}{Rr^2}-$$
$$\frac{A_2^{(1,0,0,0)}(Rr,Ttheta,Zz,t)c^2}{Rr}+\frac{2A_1^{(1,1,0,0)}(Rr,Ttheta,Zz,t)c^2}{Rr}-$$
$$A_2^{(2,0,0,0)}(Rr,Ttheta,Zz,t)c^2+A_2^{(0,0,0,2)}(Rr,Ttheta,Zz,t)-$$
$$\frac{4\pi J_2(Rr,Zz,t)}{c}-\frac{4\pi\ {}_e v_2(Rr,Zz,t)}{c}==0$$

$$equationVectorPotencial2=$$
$$(1/c\,\hat{}\,2ReplacePart[eq3[[3]],nobo[[3,2]],5]//ExpandAll==0$$
$$-\frac{4\pi J_3(Rr,Zz,t)}{c^3}-\frac{4\pi\ {}_e v_3(Rr,Zz,t)}{c^3}+$$
$$\frac{A_3^{(0,0,0,2)}(Rr,Ttheta,Zz,t)c^2}{c^2}+\frac{2A_1^{(0,0,1,0)}(Rr,Ttheta,Zz,t)}{Rr}+$$
$$A_3^{(0,0,2,0)}(Rr,Ttheta,Zz,t)+\frac{2A_2^{(0,1,1,0)}(Rr,Ttheta,Zz,t)}{Rr}-$$
$$\frac{A_3^{(0,2,0,0)}(Rr,Ttheta,Zz,t)}{Rr^2}-\frac{A_3^{(1,0,0,0)}(Rr,Ttheta,Zz,t)}{Rr}+$$
$$2A_1^{(1,0,1,0)}(Rr,Ttheta,Zz,t)-A_3^{(2,0,0,0)}(Rr,Ttheta,Zz,t)==0$$

Взаимодействия волновых и колебательных движений простейших механических систем и внешних газодинамических и магнитных полей

- Взаимодействие колебания призматического стержня с возмущениями от внешних полей: Уравнения движения и их решение в среде Mathematica
- Свободные колебания призматического стержня в среде без сопротивления
- Совместные колебания призматического стержня и жидкости: Взаимодействие колебаний с гидродинамическими возмущениями поля давления
- Реакция стержня на возмущения магнитного поля
- Решение задачи о распространении волн в цилиндрической системе координат

Динамические уравнения поля и уравнения движения в сферических координатах

Заключение

В настоящей работе представлены результаты символьных вычислений в магнитной гидромеханике, которые решают задачи построения инвариантных векторных операторов математического векторного анализа и задачи построения фундаментальных уравнений движения и состояния среды и поля в любой криволинейной системе координат.

Приведены соответствующие выводы уравнений движения, состояния и т.п., а также обсуждены методы использования компьютерной алгебры в наиболее широко применяемых системах координат (декартова, цилиндрическая, сферическая).

В качестве примеров, приведены решения модельных задач о взаимодействии упругих движений простейших твердых деформируемых тел в гидродинамических и магнитных полях. Для решения модельных задач используются встроенные в среду Mathematica стандартные процедуры, которые показывают направление применения и развития символьных и численных методов в компьютерных технологиях.

Все решения модельных задач можно иллюстрировать графиками распространения гидродинамических и магнитных волн возмущений и волновых реакций упругих тел на эти возмущения. На картинках волновых полей видны процессы взаимодействия, которые могут применяться в учебном процессе и соответствующих курсах.

Библиография

1. Седов Л.И. Механика сплошной среды т.1., М., "Наука", 1973, 536 с.
2. Седов Л.И Механика сплошной среды т.2., М., "Наука", 1973, 584 с.
3. Зоммерфельд А. Механика деформируемых сред. М., Издательство иностранной литературы, 1954, 486 с.
4. Papusha A.N. Mathematica in Non-linearElasticityTheory Proceedingof the Second International Mathematica Symposium Computational Mechanics Publications, Southampton-Boston, Great Britain, 1997, pp. 383-390.

Информация об авторах

Лега Ю.Г. – Черкасский государственный технологический университет, д.т.н., профессор, бульв. Шевченко, 460, Черкассы, Украина

Мельник В.В. – Черкасский государственный технологический университет, к.е.н., доцент кафедры экономической кибернетики, бульв. Шевченко, 460, Черкассы, Украина

Бурцева Т.И. – Черкасский государственный технологический университет, старший преподаватель кафедры экономической кибернетики, бульв. Шевченко, 460, Черкассы, Украина

Папуша А.Н. – д.ф.-м.н., профессор Мурманского государственного технического университета

SOME APPROACHES TO DISTRIBUTED ENCODING OF SEQUENCES

Artem Sokolov, Dmitri Rachkovskij

***Abstract**: We discuss several approaches to similarity preserving coding of symbol sequences and possible connections of their distributed versions to metric embeddings. Interpreting sequence representation methods with embeddings can help develop an approach to their analysis and may lead to discovering useful properties.*

***Keywords**: sequence similarity, metric embeddings, distributed representations, neural networks*

Introduction and Background

In various applications, it is necessary to search for similar sequences of data. Examples include (but are not limited to) gene similarity search in biology, speech recognition, document database or Internet search, comparison of network traffic flows in computer security systems or detecting dangerous deviations from normal behavior of users by observing sequences of their actions.

There are many ways to formalize an intuitive notion of similarity ("looking the same") between strings[1], e.g., by edit distance. Given a set of edit operations, edit distance $L(s,t)$ between string s and t is the minimum number of edit operations needed to transform s into t. This definition benefits from being intuitive clear, simple to understand, and is often motivated by applications (e.g. evolutionary arguments in biology).

The simplest edit distance is the Hamming distance that counts the number of positions where strings differ – or, alternatively, the number of character changes needed to transform one string to another. Being simple (however, very useful in some applications, e.g. error correction), it is not sufficient for many real-world symbol sequences.

A more general definition is the Levenshtein distance [Levenshtein, 1965], where operations are symbol changes, insertions and deletions. Its exact value can be computed using dynamic programming algorithm in $O(n^2)$ time. More flexible definitions extend the set of possible operations with block copies, moves, indels etc.

In applications, sequence length can be very large, especially in Internet and networking applications and/or applications working with streaming data. Estimating their similarity requires fast algorithms, making time consumption of exact algorithms prohibitive. For example, finding Levenshtein distance in quadratic time is not fast enough. Finding minimum sequence of block edit operations is substantially harder – some versions of it were proved to be NP-hard [Lopresti, Tomkins, 1997].

As representations allowing for a simple (element-wise) definition of similarity or distance, consider vector representations of symbolic sequences. Let each element of a vector represent some item - e.g. some substring of the input string. Depending on the chosen sets of substrings, we obtain representations known by different names in literature. If items are all substrings of nearby symbols of length q and the vector contains their occurrence numbers/frequencies, they are known as "q-grams" [Ukkonen, 1992]. If the sequence is a text and items are single words (i.e., q=1), we get a common “bag-of-words” representation often used in Vector Space Models (VSM) for informational retrieval [Salton, 1989]. The latter case (q=1) does not take order of items into account, but this can be regulated by setting q>1.

Such vector representations can be linked with edit distance through an observation that when two strings s and t are within a small edit distance of each other, they share a large number of items. Vector representations of strings can be used for (weakly, see further) approximating edit distances.

A way to solve the problem of fast finding similar sequences is to look for approximate solutions. One research area where such approximate solutions are sought is embedding techniques [Indyk, 2004]. They deal with mapping complex data to some “easier” space, which allows finding or approximating distances faster. “Easy” spaces are usually vectors spaces; particularly interesting are Euclidean and Hamming ones. In those spaces, efficient algorithms may be already available for a specific task (like nearest neighbor search) and/or computing similarity or distances between vectors can be done faster than in the original “complex” space.

The idea behind embeddings is that smaller parts of objects are often sufficient to approximate distances or similarity, provided objects are partitioned in sufficiently random manner. In a number of interesting cases, this happens because of the phenomenon known as concentration of measure. An excellent state-of-the-art review of embeddings of general metrics as well as of special metrics such as edit distances is given in [Indyk, 2004].

Another research area where similar problems are considered is distributed representations that try to capture brain’s way of representing objects [Thorpe, 2003; Arbib, 2003]. In distributed representations data types of various complexity – from elementary to structured ones – are sought to be represented by patterns of activity over pools of "neurons", which can be thought of as "codevectors" (e.g., [Rachkovskij, Kussul, 2001]). It is believed that brain uses similar representations for an efficient recall and comparison of complex internal models of real-world objects.

It was commonly thought that distributed representations are suitable only for "bag-of" representations [Feldman, 1989; Malsburg, 1986], however, the introduction of the so-called binding operations changed the situation [Plate, 2003]. In contrast to the non-distributed representations described above, here similarity of even elementary items can be reflected in the degree of correlation of their codevectors. Another property is the possibility of composing "reduced representations" of complex objects from subsets of codevectors of their parts as well as reconstructing full representations of parts from a reduced representation of

[1] We consider sequences composed of symbols from finite alphabet Σ (e.g., letters, commands, macromolecules), i.e. strings, as opposed to other types of sequences (e.g. speech, movements, behaviors), where components are not so evident.

the whole. Recursive construction of reduced representation results in a codevector representing the whole complex object. Different ways of combining components and reducing their representations give potential for constructing representations that reflect some necessary application-specific notion of similarity. One can use dot products or whatever for measuring similarity of codevectors – which are usually of the same dimensionality even for items of different complexity.

For sequences, goals for embeddings and distributed representations are similar at least in the following: both try to find such a vector representation of sequences that preserves necessary application-specific similarity and provides fast calculation of similarity or difference in the target space. In this paper, we describe some ways of introducing order information into distributed representations, including those of binary sparse type; and their connection to traditional sequence vector representations and embeddings. We show that some non-distributed and distributed sequence-processing methods can be related through random embeddings (particularly, random projections) and can be viewed in a coherent way. We think that connecting sequence representation methods with embedding theory can help develop approach to their analysis and discover useful properties.

Vector Representation of Strings by q-grams

In this section, we mention some of non-distributed representations that can be characterized as bag-of representation approaches and that allow some approximation of edit distance.

Consider a string s and a sliding window of q symbols on it. For each possible window content (q-gram) the number of its occurrences in the string is recorded in a corresponding coordinate of the vector $v_q(s)$ (q-gram vector). In case of q=1 it is just a vector with elements equal to the number of times a respective letter was met in a sequence, equivalent to frequency vector in VSM. As it was noted in [Ukkonen, 1992], each edit operation changes at most q q-grams, so if the edit distance is at most L, then

$$||v_q(s)-v_q(t)|| \le qL \qquad (2)$$

This method discards order of seen q-grams. Nevertheless, this gives a way for approximating edit distance from below $L \ge ||v_q(s)-v_q(t)||/q$ and can be used for filtering. Given a query string for which it is necessary to find the nearest one from a set of strings, too distant strings can be filtered out using Manhattan distance between q-gram vectors. Developed further, the idea of q-grams can be used to define such notions of similarity as "resemblance" and "containment" of strings [Broder, 1997].

Another interesting application of q-grams is solving gapped edit distance problem [Bar-Yossef et al, 2004]. To solve a k vs. m gapped edit distance problem is to be able to answer, given strings s and t, whether the edit distance between them is less than k or it is greater than m. In the algorithm, each string is divided into non-intersecting regions of some length D. Then each q-gram is accompanied with a fingerprint that is equal to the number of region it starts within, constituting a set of pairs (γ, i), where $\gamma \in \Sigma^q$. Pairs are considered equal only if their q-grams are equal and they appear in the same region. So, identical q-grams, but starting far enough from each other, receive different fingerprints and are different. Then Hamming distance is measured between vectors corresponding to the sets of such pairs (γ, i). It turns out that it is possible to solve k vs. $(kn)^{2/3}$ gapped edit distance problem by measuring Hamming distance between characteristic vectors of sets of fingerprinted q-grams. Second step in [Bar-Yossef et al, 2004] is to use dimensionality-reducing technique in Hamming space from [Kushilevitz et al., 1998].

Distributed Sequence Coding

Let us consider some ideas of representing strings with distributed representations. Since symbols are considered non-similar, they are assigned independent random binary codevectors. Here we consider non-hierarchical version of order representation, that are designed for representation of short sequences – to represent long strings it may be necessary to build representations in a hierarchical manner.

Distributed representation of strings by unordered substrings. Let us consider a distributed version of q-gram representation from the previous section. Let us take the allocated random vectors for each of sequence

substrings and sum up (or take disjunction of) them to form a codevector of the sequence. Sequences containing many identical elements would receive close codes. This method takes information about order in the sequence into account only to an extent captured by choosing substrings.

This method is somewhat similar to the Random Indexing (RI) and Random Labeling (RL) methods used in semantic processing of texts, which are versions of vector space methodologies for producing distributed words' representations using co-occurrence data [Kanerva et al., 2000; Karlgren, Sahlgren, 2001]. There a word can be abstractly viewed as a set of contexts it is used in. Each context is a bag of words (either a text where the word was met (RI) or a word's neighborhood (RL)). Each element of a context (either a text or a word) is assigned a random vector (with small number of +1 and -1). Context representation is a sum of those random vectors corresponding to constituent words. Target word representation forms by adding those context vectors each time this word appears in a corpus. As we will see in the following sections, this method can be interpreted straightforwardly with embeddings.

The following methods are trying to merge information about position of an item within a sequence with the representation of the item itself by modifying item's representation in a position-dependent manner. In each of following schemes vectors can be made binary (sums could be substituted by disjunctions), making dot product (a common similarity measure) computation efficient.

Positional binding with permutations. Consider the so-called shift coding which exemplifies an attempt to preserve information about ordering in a string. Vectors are modified by circularly shifting (or otherwise permuting) item's vector by the number of bits that depends on the position of the item. Code of the whole sequence is formed by disjunction of codes of all constituent items.

Consider e.g. strings '*abc*', '*abd*', '*cba*'. Each symbol is represented by a random vector: $v(a)$, $v(b)$, $v(c)$, etc. Then sequence codevectors are formed in this way:

$$v(abc) = (v(a) >> 1) \vee (v(b) >> 2) \vee (v(b) >> 3)$$

$$v(bca) = (v(b) >> 1) \vee (v(c) >> 2) \vee (v(a) >> 3),$$

where $\vee$ is bit disjunction, $X>>y$ means codevector X shifted by y bits.

The intersection between obtained vectors for strings with no identical items in the same positions will be (in expectation) negligible, provided the random vectors are sufficiently sparse and strings of short enough length. Thus, this coding scheme discards information about identical symbols in different positions. However, partial permutation or shift may provide a way to fix that.

Positional binding with codevector. Here codevectors for positions are generated additionally to those for substrings. To code a substring in a particular position, the codevector of the substring and the codevector of its position are bound. In the binary case, binding is done by bit-wise conjunction and is called thinning [Rachkovskij, Kussul, 2001]. Other types of binding are also possible, both for binary codevectors [Rachkovskij, Kussul, 2001; Kanerva, 1996] and codevectors with continuous elements [Plate, 2003].

If we encode substring as continuous codevectors and positions as binary codevectors, binding by element-wise product is possible (see also Gayler's multiplicative binding for real-valued codevectors in [Plate, 2003]), which in this case can be also considered as thinning. Thinning does not remove similarity of identical elements in different positions, as the shift method does. Representations of positions may be correlated for nearby ordinal numbers. Codevectors of symbol-position bindings are then combined by bit-wise disjunction or addition.

Binding an element with its context. Another option is to bind item's vector with vectors of item's from its context. This may give way to build position-independent representation of items, where item's contribution depends only on its context and does not depend directly on the position in a sequence. As edit metrics is usually also position-independent in the sense of counting edit operations independently of the place they were applied; this may help to approximate them. Note an analogy with taking into account contexts by q-grams.

Embeddings and Representational Economy

In this section we discuss some results for embedding vector and sequence distances. Target spaces for embedding are usually vectors spaces like l_p (where $||x||_p=(\sum_{i=1,\ldots,d}x_i^p)^{1/p}$), particularly interesting are Euclidean spaces (p=2), l_1 with Manhattan metrics ($||x||_1=\sum_{i=1,\ldots,d}|x_i|$) or the Hamming space ($\rho(x,y)=\sum_{i=1,\ldots,d}[x_i\neq y_i]$).

The seminal result that we will use is the Johnson-Lindenstrauss reduction lemma [Johnson, Lindenstrauss, 1984]. Let the elements of matrix $R_{d',d}$ be from $N(0,1)$ distribution. Let vectors $v'=Rv$. Then for every $\varepsilon>0$ and any two vectors $v_1,v_2 \in l^d_2$:

$$(1-\varepsilon)||v_1-v_2||_2 \leq ||v'_1-v'_2||_2 \leq (1+\varepsilon)||v_1-v_2||_2 \quad (3)$$

holds with probability $\exp(\Omega(-d'/\varepsilon^2))$. In case of normal distribution of the vectors embedding into normed space is due to 2-stability of the normal distribution. There exist embeddings of norms for l_p using other p-stable distributions for $0<p\leq 2$ (see [Cormode et al, 2002] for a brief overview). Thus it is possible to logarithmically reduce the input dimension while distorting mutual distance not too much. Note that instead of vector elements distributed according to normal distribution we can use either binary {-1,1} or ternary {-1,0,1} elements (with proper distribution) as proven in [Achlioptas, 2001].

Despite the progress in embedding "usual" metrics, embedding Levenshtein distance is a long-standing problem [Indyk, 2001], and a negative result was proved recently that any such algorithm would not have distortion smaller than 3/2 [Andoni et al., 2003]. However, it is possible to embed a relaxed version of edit distance (with block moves) to l_1 with approximately logarithmic distortion $\tilde{O}(log(n))$ by carefully selecting substrings, which add to the characteristic vector of the sequence [Cormode et al, 2000]. This result is particularly interesting because the exact calculation of this distance is NP-hard.

Connection between Distributed and Non-distributed Sequence Processing

In this subsection, we show how to interpret some distributed sequence coding methods with the help of embedding theory. We consider continuous case, so elements of the used vectors are from R and operations are usual summation and multiplication; as well as binary case, where operations are Boolean OR and AND.

Distributed representation of strings by unordered substrings According to it, a codevector $v(s)$ for a string $s=s_1,\ldots,s_n$ is defined as $v(s)=\Sigma_{i=1,\ldots,n}\, r(s_i)$, where $r(s)$ is a random codevector corresponding to an item s. The expression for $v(s)$ can be rewritten as

$$v(s) = \sum_{i=1,\ldots,n} r(s_i) = \sum_{\sigma\in\Sigma}\sum_{i=1,\ldots,n} I[s_i=\sigma] = \sum_{\sigma\in\Sigma} r(\sigma)\, n_s(\sigma), \quad (4)$$

where $n_s(\sigma)$ is the number of times σ occurs in s.

Expression (3) is the same as projecting a bag-of-items vector $n_s^T(\sigma)=(n_1(\sigma),n_2(\sigma),\ldots,n_{|\Sigma|}(\sigma))$ by multiplying it by a random matrix ($d\times|\Sigma|$) with columns $r(\sigma)$. Thus, this coding can be viewed as mapping from (e.g., VSM) space of dimension $|\Sigma|$ to R^d, where d can be made considerably lower then the number of all possible items $|\Sigma|$.

If both spaces are Euclidean, then, applying JL-lemma (1) and taking $r(s)$ from normal distribution (or from certain binary or ternary distributions, see [Achlioptas, 2001]) we obtain, that for a desired distortion $0<\varepsilon<1$, it is possible to reduce dimension, while keeping $||v(s)-v(t)||$ within the range $(1\pm\varepsilon)||v_q(s)-v_q(t)||$ with high probability. This is what is done in RI (however, with sparse codevectors) appealing to "near orthogonality" of random vectors in high dimensions (but, in the same time, dimension can be considerably lower than the number of all contexts). And so using inequality (2), for the two strings within edit distance less then L the Euclidean distance between corresponding codevectors is no more then $(1+\varepsilon)qL$. This provides an upper bound for distance between vectors, and can be used for filtering purposes using representation vectors of low dimension instead of large (however, sparse) q-gram vectors.

Binary sketch There are evidences that taking into account only information about presence of words in texts, it is possible to preserve similarity of texts to an extent sufficient for some applications [Grossman, Frieder, 1998].

Let $k(s,t)$ be the the number of common items and $n(s)$ and $n(t)$ are the total number of items in either s or t. Then the expectation of the inner product of resulting vectors is binomially distributed with parameters $(1+(1-p)^{n(s)}+(1-p)^{n(t)}-(1-p)^{n(t)+n(s)-k(s,t)},d)$. Thus, measuring inner product of such sketches can provide means to

estimate common set of items in the sequences without having to store the whole sequence. However, we may need to add the total number of items into the sketch at the cost $O(\log(n))$.

Connection of thinning coding to random sampling In this subsection show analogy between thinning coding and some of the edit distance approximation approaches. Consider the *i*'s element of output vector $v(s)$:

$$v_i(s) = \sum_{k=1,\ldots,n} c_{ki}\, r_{i\,s[k]} = \sum_{k=1,\ldots,n} c_{ki} \sum_{\sigma \in \Sigma} r_{i\sigma}\, I[s_k=\sigma] = \sum_{\sigma \in \Sigma} \sum_{k=1,\ldots,n} r_{i\sigma}\, I_{\sigma k}\, c_{ki}. \quad (4)$$

Then we can define $V(s)=R \cdot L(s) \cdot C$, where columns of matrix R are random codevectors r_i assigned to the *i*'-th symbol of Σ and rows of C are position codevectors C_i.. Elements of the indicator matrix L are: $I_{\sigma k}=1$ if the *k*'-th substring (or symbol if $q=1$) of string s is σ and $I_{\sigma k}=0$ otherwise. That is, the element of the matrix shows where symbols from the alphabet (symbols or substrings) in the string are situated.

Consider the product $X=L(s) \cdot C$. If matrix C had contained only 1's in each of its cells, then this product would have just given columns of vector representation (e.g., q-gram) for the string. But, as C does contain zeros, substring in not all of the positions is counted into the vector. So, columns of the resulting matrix X would become an "incomplete" vector representations (q-gram representations, if Σ is all q-grams) of the input string s. Columns of matrix C act as binary masks marking positions in s, from where symbols sum up to a particular q-gram vector, e.g., the *j*'s column of C masks s to obtain a vector with the *i*'s element equal to $\sum_{k=1,\ldots,n} I_{\sigma k} c_{ki}$. So, while earlier q-gram representations discarded order information in a sequence, here we introduced it with the help of position vectors.

Above, it was noted that position codevectors (rows of C) can be made so that nearby ones have small Hamming distance of each other and this distance grows as the distance between positions grows. Possible ways of constructing such codevectors from for ordinal numbers are considered in [Rachkovskij et al, 2005] and can also be implemented with a 2-state Markov chain with proper transition probabilities.

Different columns act as independent samplers. We note that similar approach has already led to even sublinear approximation of edit distance in [Batu et al, 2004]. Their approach is similar to ours in that the approximation is also achieved by randomly sampling input string and using the already mentioned observation that strings within small edit distance have many substrings in nearby positions.

Consider now the effect of multiplying X by matrix R. Each *i*-th element of the output vector $v(s)$ is a dot product of the random codevector r_i by the corresponding "thinned" q-gram vector. We already saw such an operation (projection on a random direction) when established analogy between unordered distributed encoding of strings by assigning random codevectors to their substrings (q-grams). The difference is that there each of the q-grams was projected to all random directions, but here different thinned q-gram vectors are projected along different directions. However, if we look closer, we note that because of intersections between columns of C "common parts" of masked (thinned) q-gram vector may undergo projection to different directions. From expression (4):

$$v_i(s) = \sum_{\sigma \in \Sigma} \sum_{k=1,\ldots,n} (c_{ki}\, r_{i\sigma}) I_{\sigma k} = \sum_{k=1,\ldots,n} R(C(k)) \cdot I_k. \quad (4)$$

$R(C(k))$ is matrix R with only those rows left that correspond to those position codevectors that have 1s in position k. So, we see that each vector I_k undergoes projection to a respective random subspace, defined by those C_j that cover position k. The farther are identical symbols (or q-grams) from each other, the less common random directions they have, and so, the more distant are their projections.

Conclusions

We believe that enriching distributed representations with ideas and methods of analysis from embeddings can provide a more formal interpretation of distributed methods usually introduced in an ad-hoc manner. This may help infer the similarity type they approximate or find modifications that will allow them to approximate some, and help obtain bounds on the distortion of the proposed coding schemes, their complexity and limitations.

Here we have described approach to analysis of only few distributed schemes for coding sequences. We hope it could be extended to other schemes for encoding sequences mentioned above, as well as to schemes for distributed encoding of other types of data like numerical [Rachkovskij et al, 2005] or complex relational structures [Rachkovskij, 2004]. Those schemes include binding by context-dependent thinning and hierarchical representations [Rachkovskij, Kussul 2001].

Bibliography

[Achlioptas, 2001] D. Achlioptas, Database-friendly random projections. In Proceedings of PODS-01, pp. 274-281, 2001.

[Andoni et al., 2003] A. Andoni, M. Deza, A. Gupta, P. Indyk, S. Raskhodnikova, Lower Bounds for Embedding of Edit Distance into Normed Spaces. In Proc. of the 14th Symposium on Discrete Algorithms, 2003

[Arbib, 2003] M. Arbib, The Handbook of Brain Theory and Neural Networks. – Cambridge, MA: The MIT Press, 2003

[Bar-Yossef et al, 2004] Z. Bar-Yossef, T.S. Jayram, R. Krauthgamer, R. Kumar: Approximating Edit Distance Efficiently. FOCS, pp. 550-559, 2004

[Batu et al, 2004] T. Batu, F. Ergun, J. Kilian, A. Magen, S. Raskhodnikova, R. Rubinfeld, R. Sami. A sublinear algorithm for weakly approximating edit distance, In Proc. 36th STOC, 2004

[Broder, 1997] A. Z. Broder. On the resemblance and containment of texts. In Proceedings of Compression and Complexity of SEQUENCES, 1997

[Cormode et al, 2000] G. Cormode, M. Paterson, S. C. Sahinalp, U. Vishkin. Communication complexity of text exchange. In Proc. of the 11th ACM-SIAM Annual Symposium on Discrete Algorithms, pp. 197--206, San Francisco, CA, 2000

[Cormode et al, 2002] G. Cormode, P. Indyk, N. Koudas, S. Muthukrishnan. Fast mining of tabular data via approximate distance computations. In Proc. of the International Conference on Data Engineering, pp. 605–616, 2002

[Feldman, 1989] J. Feldman, Neural Representation of Conceptual Knowledge. In L. Nadel, L. Cooper, P. Culicover et al. Neural Connections, mental computation - London, England: The MIT Press, pp. 68-103, 1989

[Grossman, Frieder, 1998] D.A. Grossman, O. Frieder, Information Retrieval: Algorithms and Heuristics, Kluwer, 1998

[Indyk, 2001] P. Indyk, Algorithmic Aspects of Geometric Embeddings, FOCS, 2001

[Indyk, 2004] P. Indyk, Embedded Stringology, talk at the Symposium on Combinatorial Pattern Matching 2004. Available at http://theory.lcs.mit.edu/~indyk/cpm.ps

[Johnson, Lindenstrauss, 1984] W. Johnson, J. Lindenstrauss, Extensions of Lipschitz maps into a Hilbert space, Contemp. Math., 26 , pp. 189-206, 1984

[Kanerva et al, 2000] P. Kanerva, J. Kristofersson, A. Holst. Random indexing of text samples for latent semantic analysis. In Proc. of the 22nd Annual Conference of the Cognitive Science Society, p. 1036. Erlbaum, 2000

[Kushilevitz et al., 1998] E. Kushilevitz, R. Ostrovsky, Y. Rabani. Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces. STOC, pp. 614-623, 1998

[Levenshtein, 1965] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals, Doklady Akademii Nauk SSSR, 163(4):845-848, 1965 (Russian)

[Lopresti, Tomkins, 1997] D. P. Lopresti, A. Tomkins, Block Edit Models for Approximate String Matching. Theoretical Computer Science, 181(1): 159-179, 1997

[Malsburg, 1986] C. von der Malsburg. Am I Thinking Assemblies? In Proc. of the Trieste Meeting on Brain Theory, pp.161-176, 1986

[Plate, 2003] T. Plate, Holographic Reduced Representation: Distributed Representation for Cognitive Structures. - Chicago: Center for the Study of Language and Information, 2003

[Rachkovskij et al, 2005] D.A. Rachkovskij, S.V. Slipchenko, E.M. Kussul , T.N. Baidyk, Sparse binary distributed encoding of scalar values, (to be published) 2005

[Rachkovskij, 2004] D.A. Rachkovskij, Some approaches to analogical mapping with structure sensitive distributed representations. Journal of Experimental and Theoretical Artificial Intelligence,16, №3, pp.125-145, 2004

[Rachkovskij, Kussul, 2001] D.A. Rachkovskij, E.M. Kussul, Binding and Normalization of Binary Sparse Distributed Representations by Context-Dependent Thinning. Neural Computation, 2, №13, pp.411-452, 2001

[Salton, 1989] G. Salton, G. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer, Addison-Wesley, Reading, MA., 1989

[Thorpe, 2003] S. Thorpe, Localized Versus Distributed Representations. In Arbib M. The Handbook of Brain Theory and Neural Networks - Cambridge, MA: MIT Press, pp. 643-646, 2003

[Ukkonen, 1992] E. Ukkonen. Approximate string-matching with q-grams and maximal matches. Theoretical Computer Science, 92:191-211, 1992

Authors' Information

Artem M. Sokolov, Dmitri A. Rachkovskij – International Research and Training Center of Information Technologies and Systems; Pr. Acad. Glushkova, 40, Kiev, 03680, Ukraine; e-mails: sokolov@ukr.net, dar@infrm.kiev.ua

5.2. Modal Logic

REPRESENTING THE CLOSED WORLD ASSUMPTION IN MODAL LOGIC

Frank M. Brown

***Abstract**: The nonmonotonic logic called the Closed World Assumption is shown to be representable in a monotonic Modal Quantificational Logic whose modal laws are stronger than S5. Specifically, it is proven that a set of sentences of First Order Logic is equal to the Closed World Assumption of an initial set of sentences and defaults if and only if the meaning of that set of sentences is logically equivalent to a particular modal functor of the meanings of that initial set of the sentences and those defaults. This result is important because the modal representation allows the use of powerful automatic deduction systems for Modal Logic and because unlike the original Closed World Assumption, it is easily generalized to the case where quantified variables may be shared across the scope of the components of the defaults thus allowing such defaults to produce quantified consequences.*

***Keywords**: Closed World Assumption, Modal Logic, Nonmonotonic Logic.*

1. Introduction

One very simple nonmonotonic logic is the Closed World Assumption [Reiter 1978]. The basic idea of the Closed World Assumption is that there is a set of axioms Γ and some non-logical "inference rules" of the form:

$$\frac{: \neg\chi}{\neg\chi}$$

which suggests that $\neg\chi$ may be inferred whenever it is consistent with Γ. Such "inference rules" are not effective since the determination as to whether χ is derivable depends on whether χ is consistent which is not an effective operation for a First Order Logic. Thus, tentatively applying such inference rules by checking the consistency of χ with the current set of inferences from Γ rather than with the final result produces a χ result which may later have to be retracted if it should later be found to be inconsistent with other inferences (such as those that would be deduced when another axiom were examined). For this reason valid inferences in a nonmonotonic logic such as the Closed World Assumption are essentially carried out not in the original nonmonotonic language, but rather in some (monotonic) metatheory in which that nonmonotonic logic is defined.

This intuition may be explicated by defining the Closed World Assumption in terms of the set theoretic proof theory metalanguage of First Order Logic (i.e. FOL) with the following sentence:

$$\kappa=(\text{cwa } '\Gamma\ '\chi_i)$$

where cwa is defined as:

$(\text{cwa } '\Gamma\ '\chi_i) =_{df} (\text{fol}('\Gamma\cup\{'(\neg\chi_i): ('\chi_i\notin(\text{fol } '\Gamma))\}))$[1,2]

where $'\chi_i$ is the closed sentence of FOL occurring in the ith "inference rule". and $'\Gamma$ is a set of closed sentences of FOL. A closed sentence is a sentence without any free variables. fol is a function which produces the set of theorems derivable in FOL from the set of sentences to which it is applied. The quotations appended to the front of these Greek letters indicate references in the metalanguage to the sentences of the FOL object language. Interpreted doxastically this fixed point equation states:

---

[1] [Reiter 1978] defined cwa as: $(\text{cwa } '\Gamma\ '\chi_i) =_{df} ('\Gamma\cup\{'(\neg\chi_i): ('\chi_i\notin(\text{fol } '\Gamma))\})$. If that definition were used then all the theorems in this paper should have $(\text{cwa } '\Gamma\ '\chi_i)$ replaced by $(\text{fol } (\text{cwa } '\Gamma\ '\chi_i))$.

[2] The Closed World Assumption is often presented less generally by restricting $'\chi_i$ to begin with a particular predicate symbol or one or more such predicate symbols and by requiring that the arguments to those predicates range over all possible variable free terms of particular types [Reiter 1978].

the set of closed sentences which are believed is equal to:
the set of closed sentences derived in FOL from
the union of the initial beliefs consisting of a set of closed sentences: 'Γ
and the set of closed sentences of the form '($\neg\chi_i$)
such that for each i, the closed sentence 'χ_i is not initially believed.

The purpose of this paper is to show that all this metatheoretic machinery including the formalized syntax of FOL, the proof theory of FOL, the axioms of a strong set theory, and the set theoretic equation is not needed and that the essence of the Closed World Assumption is representable as a necessary equivalence in a simple (monotonic) Modal Quantificational Logic. Interpreted as a doxastic logic this necessary equivalence states:

that which is believed is logically equivalent to
the initial beliefs Γ and for each i, if $\neg\chi_i$ is initially believable then $\neg\chi_i$

thereby eliminating all mention of any metatheoretic machinery.

The remainder of this paper proves that this modal representation is equivalent to the Closed World Assumption. Section 2 describes a formalized syntax for a FOL object language. Section 3 describes the part of the proof theory of FOL needed herein (i.e. theorems FOL1-FOL4). Section 4 describes the Intensional Semantics of FOL which includes laws giving the meaning of FOL sentences: M0-M7, theorems giving the meaning of sets of sentences: MS1, MS2, MS3, and laws specifying the relationship of meaning and modality to the proof theory of FOL (i.e. the laws R0, A1, A2, and A3 and the theorems: C1, C2, C3, and C4). The modal version of the Closed World Assumption, called CWA, is defined in section 5 and explicated with theorems MC1-MC6 and SS1-SS2. In section 6, this modal version is shown by theorems CWA1 and CWA2 to be equivalent to the set theoretic fixed point equation for the Closed World Assumption. Finally, in section 7, some consequences are discussed.

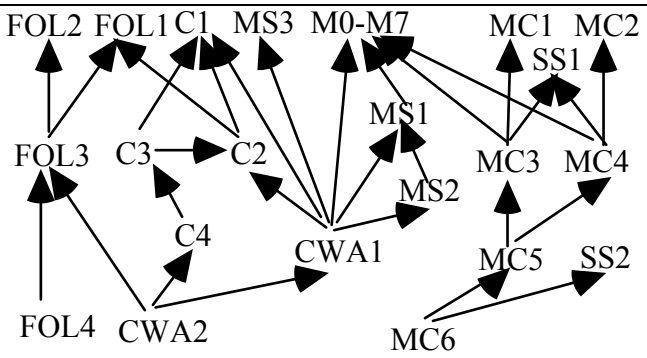


Figure 1: Dependencies among the Theorems

2. Formal Syntax of First Order Logic

We use a First Order Logic (i.e. FOL) defined as the six tuple: ($\rightarrow$, #f, $\forall$, *vars*, *predicates*, *functions*) where $\rightarrow$, #f, and $\forall$ are logical symbols, *vars* is a set of variable symbols, *predicates* is a set of predicate symbols each of which has an implicit arity specifying the number of associated terms, and *functions* is a set of function symbols each of which has an implicit arity specifying the number of associated terms. The sets of logical symbols, variables, predicate symbols, and function symbols are pairwise disjoint. Lower case Roman letters possibly indexed with digits are used as variables. Greek letters possibly indexed with digits or lowercase roman letters are used as syntactic metavariables. γ, γ_1,...γ_n, range over the variables, ξ, ξ_1...ξ_n range over sequences of variables of an appropriate arity, π,π_1...π_n range over the predicate symbols, ϕ, ϕ_1...ϕ_n range over function symbols, δ, δ_1...δ_n, σ range over terms, and α, α_1...α_n, β, β_1...β_n,χ, χ_1...χ_n,Γ,Γ_1,...Γ_n,φ range over sentences. The terms are of the forms γ and (ϕ δ_1...δ_n), and the sentences are of the forms ($\alpha\rightarrow\beta$), #f, ($\forall\gamma$ α), and (π δ_1...δ_n). A nullary predicate π or function ϕ is written as a sentence or a term without parenthesis.

$\varphi\{\pi/\lambda\xi\alpha\}$ represents the replacement of all unmodalized occurrences of π in φ by $\lambda\xi\alpha$ followed by lambda conversion. The primitive symbols are shown in Figure 2 with their intuitive interpretations.

Symbol	Meaning
$\alpha \rightarrow \beta$	if α then β.
#f	falsity
$\forall\gamma\ \alpha$	for all γ, α.

Figure 2: Primitive Symbols of First Order Logic

The defined symbols are listed in Figure 3 with their definitions and intuitive interpretations.

Symbol	Definition	Meaning		Symbol	Definition	Meaning
$\neg\alpha$	$\alpha \rightarrow$ #f	not α		$\alpha\wedge\beta$	$\neg(\alpha \rightarrow \neg\beta)$	α and β
#t	$\neg$ #f	truth		$\alpha\leftrightarrow\beta$	$(\alpha\rightarrow\beta)$ $\wedge(\beta\rightarrow\alpha)$	α if and only if β
$\alpha\vee\beta$	$(\neg\alpha)\rightarrow\beta$	α or β		$\exists\gamma\ \alpha$	$\neg\forall\gamma\neg\alpha$	for some γ , α

Figure 3: Defined Symbols of First Order Logic

The FOL object language expressions are referred in the metalanguage (which also includes a FOL syntax) by inserting a quote sign in front of the object language entity thereby making a structural descriptive name of that entity. Generally, a set of sentences is represented as: $\{'\Gamma_i\}$ which is defined as: $\{'\Gamma_i: \#t\}$ which in turn is defined as: $\{s: \exists i(s='\Gamma_i)\}$ where i ranges over some range of numbers (which may be finite or infinite). With a slight abuse of notation we also write $'\kappa$, $'\Gamma$ to refer to such sets.

3. Proof Theory of First Order Logic

FOL is axiomatized with a recursively enumerable set of theorems as the set of axioms is itself recursively enumerable and its inference rules are recursive. The axioms and inference rules of FOL [Mendelson 1964] are given in Figure 4:

MA1: $\alpha \rightarrow (\beta\rightarrow \alpha)$ MR1: from α and $(\alpha\rightarrow \beta)$ infer β

MA2: $(\alpha\rightarrow (\beta\rightarrow \rho)) \rightarrow ((\alpha\rightarrow \beta)\rightarrow (\alpha\rightarrow \rho))$ MR2: from α infer $(\forall\gamma\ \alpha)$

MA3: $((\neg\alpha)\rightarrow (\neg\beta))\rightarrow (((\neg\alpha)\rightarrow \beta)\rightarrow\alpha)$

MA4: $(\forall\gamma\ \alpha)\rightarrow \beta$ where β is the result of substituting an expression (which is free for the free positions of γ in α) for all the free occurrences of γ in α.

MA5: $(\forall\gamma(\alpha \rightarrow \beta)) \rightarrow (\alpha\rightarrow(\forall\gamma\ \beta))$ where γ does not occur in α.

Figure 4: Inferences Rules and Axioms of FOL

In order to talk about sets of sentences we include in the metatheory set theory symbols This set theory includes the symbols ε, $\notin$, $\supseteq$, $=$, $\cup$ as defined in [Quine 1969].

The derivation operation (i.e. fol) of any FOL obeys the Inclusion (i.e. FOL1), and Idempotence (i.e. FOL2) properties:

FOL1: $(\text{fol } '\kappa)\supseteq'\kappa$ Inclusion

FOL2: $(\text{fol } '\kappa)\supseteq(\text{fol}(\text{fol } '\kappa))$ Idempotence

From these two properties we prove:

FOL3 $(\text{cwa}(\text{fol } '\Gamma))'\chi_i)=(\text{fol}(\text{cwa}(\text{fol } '\Gamma)'\chi_i))$

proof: FOL1 and FOL2 imply that $(\text{fol}(\text{fol } '\kappa))=(\text{fol } '\kappa)$. Since cwa begins with fol this implies: $'k=(\text{fol}(\text{cwa } 'k))$ QED.

FOL4: $('\kappa=(\text{cwa } '\Gamma\ '\chi_i))\rightarrow('\kappa=(\text{fol } '\kappa))$

proof: From the hypothesis and FOL3 $'\kappa=(\text{fol}(\text{cwa } '\Gamma\ '\chi_i))$ is derived. Using the hypothesis to replace $(\text{cwa } '\Gamma\ '\chi_i)$ by $'\kappa$ in this result gives: $('\kappa=(\text{fol } '\kappa))$ QED.

4. Intensional Semantics of FOL

The meaning (i.e. mg) [Brown 1977, Boyer & Moore 1981] or rather disquotation of a sentence of FOL is defined in Figure 5 below[1]. mg is defined in terms of mgs which maps each FOL object language sentence and an association list into a meaning. mgn maps each FOL object language term and an association list into a meaning. An association list is a list of pairs consisting of an object language variable and the meaning to which it is bound.

M0: (mg 'α) =df (mgs '($\forall\gamma 1...\gamma n\ \alpha$)'()) where '$\gamma 1...'\gamma n$ are all the free variables in 'α

M1: (mgs '($\alpha \rightarrow \beta$)a) $\leftrightarrow$ ((mgs 'α a)$\rightarrow$(mgs 'β a))

M2: (mgs '#f a) $\leftrightarrow$ #f

M3: (mgs '($\forall\ \gamma\ \alpha$)a) $\leftrightarrow$ $\forall$x(mgs 'α(cons(cons 'γ x)a))

M4: (mgs '($\pi\ \delta 1...\delta n$)a) $\leftrightarrow$ (π(mgn '$\delta 1$ a)...(mgn 'δn a)) for each predicate symbol 'π.

M5: (mgn '($\phi\ \ \delta 1...\delta n$)a) = ($\phi$(mgn '$\delta 1$ a)...(mgn 'δn a)) for each function symbol 'ϕ.

M6: (mgn 'γ a) = (cdr(assoc 'γ a))

M7: (assoc v L)= (if(eq? v(car(car L)))(car L)(assoc v(cdr L))) where: cons, car, cdr, eq?, and if are as in Scheme.

Figure 5: The Meaning of FOL Sentences

The meaning operator (i.e. mg) is a disquotation operation: (mg 'α)$\leftrightarrow\alpha$

The meaning of a set of sentences is defined in terms of the meanings of the sentences in the set as:

(ms 'κ) =df $\forall$s((sε'κ)$\rightarrow$(mg s)).

MS1: (ms{'α: Γ}) $\leftrightarrow$ $\forall\xi$((Γ{s/'α})$\rightarrow\alpha$) where ξ is the sequence of all the free variables in 'α and where Γ is any sentence of the intensional semantics. proof: (ms{'α:Γ}) Unfolding ms and the set pattern abstraction symbol gives: $\forall$s((sε{s: $\exists\xi$((s='α)$\wedge\Gamma$)})$\rightarrow$(mg s)) where ξ is a sequence of the free variables in 'a. This is equivalent to: $\forall$s(($\exists\xi$((s='α)$\wedge\Gamma$))$\rightarrow$(mg s)) which is: $\forall$s$\forall\xi$ (((s='α)$\wedge\Gamma$)$\rightarrow$(mg s)) which is: $\forall\xi$(Γ{s/'α}$\rightarrow$(mg 'α)). Unfolding mg using M0-M7 then gives: $\forall\xi$((Γ{s/'α})$\rightarrow\alpha$) QED

The meaning of a set is the meaning of all the sentences in the set (i.e. MS2):

MS2: (ms{'Γ_i}) $\leftrightarrow$ $\forall i\forall\xi_i\Gamma_i$ proof: (ms{'Γ_i}) Unfolding the set notation gives: (ms{'Γ_i: #t}). By MS1 this is equivalent to: $\forall_i\forall\xi_i$((#t{s/'α})$\rightarrow\Gamma_i$) which is equivalent to: $\forall i\forall\xi_i\Gamma_i$ QED.

The meaning of the union of two sets of FOL sentences is the conjunction of their meanings (i.e. MS3):

MS3: (ms('$\kappa\cup\Gamma$)) $\leftrightarrow$ ((ms 'κ)$\wedge$(ms 'Γ)) proof: Unfolding ms and union in: (ms('$\kappa\cup\Gamma$)) gives:

$\forall$s((sε{s: (sε'κ)$\vee$(sε'Γ)})$\rightarrow$(mg s)) or rather: $\forall$s(((sε'κ)$\vee$(sε'Γ))$\rightarrow$(mg s)) which is logically equivalent to: ($\forall$s((sε'κ)$\rightarrow$(mg s)))$\wedge$($\forall$s((sε'Γ)$\rightarrow$(mg s))). Folding ms twice then gives:((ms 'κ)$\wedge$(ms 'Γ)) QED.

The meaning operation may be used to develop an Intensional Semantics for a FOL object language by axiomatizing the modal concept of necessity so that it satisfies the theorem:

C1: ('$\alpha\varepsilon$(fol 'κ)) $\leftrightarrow$ ([] ((ms 'κ)$\rightarrow$(mg 'α)))

for every sentence 'α and every set of sentences 'κ of that FOL object language. The necessity symbol is represented by a box: []. C1 states that a sentence of FOL is a FOL-theorem (i.e. fol) of a set of sentences of FOL if and only if the meaning of that set of sentences necessarily implies the meaning of that sentence.

One modal logic which satisfies C1 for FOL is the Z Modal Quantificational Logic described in [Brown 1987; Brown 1989] whose theorems are recursively enumerable. Z has the metatheorem: (<>Γ{$\pi/\lambda\xi\alpha$})$\rightarrow$ (<>Γ) where Γ is a sentence of FOL . Z includes all the laws of S5 Modal Logic [Hughes & Cresswell 1968] whose modal axioms and inference rules are in Figure 6. Therein, κ and Γ are sentences of the intentional semantics.

[1] The laws M0-M7 are analogous to Tarski's definition of truth except that finite association lists are used to bind variables to values rather than infinite sequences. M4 is different since mg is interpreted as being meaning rather than truth.

R0: from κ infer ([] κ)	A2: ([]($\kappa \rightarrow \Gamma$)) $\rightarrow$ (([]κ)$\rightarrow$ ([]Γ))
A1: ([]κ) $\rightarrow \kappa$	A3: ([]κ) $\vee$ ([]$\neg$[]κ)

Figure 6: The Laws of S5 Modal Logic

These S5 modal laws and the laws of FOL given in Figure 6 constitute an S5 Modal Quantificational Logic similar to [Carnap 1946; Carnap 1956], and a FOL version [Parks 1976] of [Bressan 1972] in which the Barcan formula: $(\forall\gamma([]\kappa))\rightarrow([]\forall\gamma\kappa)$ and its converse hold. The defined Modal symbols are in Figure 7 with their definitions and interpretations.

Symbol	Definition	Meaning		Symbol	Definition	Meaning
<>κ	$\neg$ [] $\neg\kappa$	α is logically possible		[κ] Γ	[] ($\kappa\rightarrow\Gamma$)	β entails α
$\kappa\equiv\Gamma$	[] ($\kappa\leftrightarrow\Gamma$)	α is logically equivalent to β		<κ> Γ	<> ($\kappa\wedge\Gamma$)	α and β is logically possible

Figure 7: Defined Symbols of Modal Logic

From the laws of the Intensional Semantics we prove that the meaning of the set of FOL consequences of a set of sentences is the meaning of that set of sentences (C2), the FOL consequences of a set of sentences contain the FOL consequences of another set if and only if the meaning of the first set entails the meaning of the second set (C3), and the sets of FOL consequences of two sets of sentences are equal if and only if the meanings of the two sets are logically equivalent (C4):

C2: (ms(fol 'κ))$\equiv$(ms 'κ) proof: The proof divides into two cases:

(1) [(ms 'κ)](ms(fol 'κ)) Unfolding the second ms gives: [(ms 'κ)]$\forall$s((sε(fol 'κ))$\rightarrow$(mg s))

By the soundness part of C1 this is equivalent to: [(ms 'κ)]$\forall$s(([(ms 'κ)](mg s))$\rightarrow$(mg s))

By the S5 laws this is equivalent to: $\forall$s(([(ms 'κ)](mg s))$\rightarrow$ [(ms 'κ)](mg s)) which is a tautology.

(2) [(ms(fol 'κ))](ms 'κ) Unfolding ms twice gives: [$\forall$s((sε(fol 'κ))$\rightarrow$(mg s))]$\forall$s((sε'κ)$\rightarrow$(mg s))

which is: [$\forall$s((sε(fol 'κ))$\rightarrow$(mg s))]((sε'κ)$\rightarrow$(mg s)) Backchaining on the hypothesis and then dropping it gives: (sε'κ)$\rightarrow$(sε(fol 'κ)). Folding $\supseteq$ gives an instance of FOL1. QED.

C3: (fol 'κ)$\supseteq$(fol 'Γ) $\leftrightarrow$ ([(ms 'κ)](ms 'Γ)) proof: Unfolding $\supseteq$ gives: $\forall$s((sε(fol 'Γ))$\rightarrow$(sε(fol 'κ)))

By C1 twice this is equivalent to: $\forall$s(([(ms 'Γ)](mg s))$\rightarrow$([(ms 'κ)](mg s)))

By the laws of S5 modal logic this is equivalent to: ([(ms 'κ)]$\forall$s(([(ms 'Γ)](mg s))$\rightarrow$(mg s)))

By C1 this is equivalent to: [(ms 'κ)]$\forall$s((sε(fol 'Γ))$\rightarrow$(mg s)). Folding ms then gives: [(ms 'κ)](ms(fol 'Γ))

By C2 this is equivalent to: [(ms 'κ)](ms 'Γ). QED.

C4: ((fol 'κ)=(fol 'Γ)) $\leftrightarrow$ ((ms 'κ)$\equiv$(ms 'Γ)) proof: This is equivalent to (((fol 'κ)$\supseteq$(fol 'Γ))$\wedge$((fol 'Γ)$\supseteq$(fol 'κ))) $\leftrightarrow$ ([(ms 'κ)](ms 'Γ))$\wedge$([(ms 'Γ)](ms 'κ)) which follows by using C3 twice.

5. The Closed World Assumption in Modal Logic

The equation for the Closed World Assumption may be expressed as a necessary equivalence in an S5 Modal Quantificational Logic [Brown 1989] as follows:

$$k\equiv(\text{CWA } \Gamma\ \chi_i)$$

where CWA is defined in Modal Logic as follows: (CWA Γ χ_i) =df $\Gamma\wedge\forall i((<\Gamma>\neg\chi_i)\rightarrow\neg\chi_i)$ where Γ and χ_i are propositions of FOL. Given below are some simple properties of CWA:

MC1: [(CWA Γ χ_i)]Γ

proof: By R0 it suffices to prove: (CWA Γ χ_i)$\rightarrow\Gamma$. Unfolding CWA gives: $(\Gamma\wedge\forall_i((<\Gamma>\neg\chi_i)\rightarrow\neg\chi_i))\rightarrow\Gamma$

which is a tautology. QED.

MC2: (<Γ>$\neg\chi_i$)$\rightarrow$([(CWA Γ χ_i)]$\neg\chi_i$)

proof: Unfolding CWA gives: $(<\Gamma>\neg\chi_i)\rightarrow([\Gamma\wedge\forall_i((<\Gamma>\neg\chi_i)\rightarrow\neg\chi_i)]\neg\chi_i)$. Using the hypotheses on the ith instance gives: $(<\Gamma>\neg\chi_i)\rightarrow([\Gamma\wedge\forall_i((<\Gamma>\neg\chi_i)\rightarrow\neg\chi_i)\wedge\neg\chi_i]\neg\chi_i)$ which is a tautology. QED.

The concept (i.e. ss) of the combined meaning of all the sentences of the FOL object language whose meanings are entailed by a proposition is defined as follows:

$$(ss\ \kappa) =df \forall s(([\kappa](mg\ s))\rightarrow(mg\ s))$$

SS1 shows that a proposition entails the combined meaning of the FOL object language sentences that it entails. SS2 shows that if a proposition is necessarily equivalent to the combined meaning of the FOL object language sentences that it entails, then there exists a set of FOL object language sentences whose meaning is necessarily equivalent to that proposition:

SS1: $[\kappa](ss\ \kappa)$ proof: By R0 it suffices to prove: $\kappa\rightarrow(ss\ \kappa)$. Unfolding ss gives: $\kappa\rightarrow\forall s(([\kappa](mg\ s))\rightarrow(mg\ s))$ which is equivalent to:$\forall s(([\kappa](mg\ s))\rightarrow(\kappa\rightarrow(mg\ s)))$ which is an instance of A1. QED.

SS2: $(\kappa\equiv(ss\ \kappa))\rightarrow \exists s(\kappa\equiv(ms\ s))$ proof: Letting s be $\{s: ([\kappa](mg\ s))\}$ gives: $(\kappa\equiv(ss\ \kappa))\rightarrow (\kappa\equiv(ms\{s: ([\kappa](mg\ s))\}))$. Unfolding ms and lambda conversion gives: $(\kappa\equiv(ss\ \kappa))\leftrightarrow (\kappa\equiv\forall s(([\kappa](mg\ s))\rightarrow(mg\ s)))$. Folding ss gives a tautology. QED.

The theorems MC3 and MC4 are analogous to MC1 and MC2 except that CWA is replaced by the combined meanings of the sentences entailed by CWA.

MC3: $[ss(CWA\ \forall i\Gamma_i\ \chi_i)]\forall i\Gamma_i$

proof: By R0 it suffices to prove: $(ss(CWA\ \forall i\Gamma_i\ \chi_i))\rightarrow\forall i\Gamma_i$ which is equivalent to: $(ss(CWA\ \forall i\Gamma_i\ \chi_i))\rightarrow\Gamma_i$

Unfolding ss gives: $(\forall s(([(CWA\ \forall i\Gamma_i\ \chi_i)](mg\ s))\rightarrow(mg\ s)))\rightarrow\Gamma_i$ which by the meaning laws M0-M8 is equivalent to: $(\forall s(([(CWA\ \forall i\Gamma_i\ \chi_i)](mg\ s))\rightarrow(mg\ s)))\rightarrow(mg\ '\Gamma_i)$. Backchaining on $(mg\ '\Gamma_i)$ with s in the hypothesis assigned to be $'\Gamma_i$ in the conclusion shows that it suffices to prove: $([(CWA\ \forall i\Gamma_i\ \chi_i)](mg\ '\Gamma_i))$ which by M0-M8 is equivalent to: $([(CWA\ \forall i\Gamma_i\ \chi_i)]\Gamma_i)$ which by the laws of S5 Modal Logic is equivalent to: $([(CWA\ \forall i\Gamma_i\ /\chi_i)]\forall i\Gamma_i)$ which is an instance of MC1. QED.

MC4: $(<k>\neg\chi_i)\rightarrow ([ss(CWA\ \Gamma\ \chi_i)]\neg\chi_i)$

proof: Unfolding the last ss gives: $(<k>\neg\chi_i)\rightarrow([\forall s(([(CWA\ \Gamma(mg\ '\chi_i))](mg\ s))\rightarrow(mg\ s))]\neg\chi_i)$

Instantiating s in the hypothesis to $'(\neg\chi_i)$ and then dropping the hypothesis gives:

$(<k>\neg\chi_i)\rightarrow([(([(CWA\ \Gamma(mg\ '\chi_i))](mg\ '(\neg\chi_i)))\rightarrow(mg\ '(\neg\chi_i)))]\neg\chi_i)$. Using the meaning laws M0-M7 gives: $(<k>\neg\chi_i)\rightarrow([(([(CWA\ \Gamma\ \chi_i)]\neg\chi_i)\rightarrow\neg\chi_i)]\neg\chi_i)$. Backchaining on $\neg\chi_i$ shows that it suffices to prove: $(<k>\neg\chi_i)\rightarrow([(CWA\ \Gamma\ \chi_i)]\neg\chi_i)$ which is an instance of theorem MC2. QED.

MC5 and MC6 show that talking about the meanings of sets of FOL sentences in the modal representation of the Closed World Assumption is equivalent to talking about propositions-

MC5: $(ss(CWA(\forall i\Gamma_i)\chi_i)) \leftrightarrow (CWA(\forall i\Gamma_i)\chi_i)$

proof: In view of SS1, it suffices to prove: $[(ss(CWA(\forall i\Gamma_i)\chi_i))](CWA(\forall i\Gamma_i)\chi_i)$

Unfolding the second occurrence of CWA gives: $[(ss(CWA(\forall i\Gamma_i)\chi_i))]((\forall i\Gamma_i)\wedge\forall_i((<(\forall i\Gamma_i)>\neg\chi_i)\rightarrow\neg\chi_i))$

which holds by theorems MC3 and MC4. QED.

MC6: $(\kappa\equiv(CWA(\forall i\Gamma_i)\chi_i))\rightarrow\exists s(\kappa\equiv(ms\ s))$

proof: From the hypothesis and MC5 $\kappa\equiv(ss(CWA(\forall i\Gamma_i)\chi_i))$ is derived. Using the hypothesis to replace $(CWA(\forall i\Gamma_i)\chi_i)$ by κ in this result gives: $\kappa\equiv(ss(CWA\ \kappa))$, By SS2 this implies the conclusion. QED.

6. The Relationship Between The Closed World Assumption and Modal Logic

The relationship between the proof theoretic definition of the Closed World Assumption [Brown 1989] and the modal representation is proven in two steps. First theorem CWA1 shows that the meaning of the set cwa is the proposition CWA and then theorem CWA2 shows that a set of FOL sentences which contains its FOL theorems is equal to cwa of an initial set of axioms and defaults if and only if the meaning (or rather disquotation) of that set of sentences is logically equivalent to CWA of the meaning of that initial set of sentences and those defaults.

CWA1: $(ms(cwa\{'\Gamma_i\}'\chi_i))\equiv(CWA(\forall i\Gamma_i)\chi_i)$

proof: $(ms(cwa\{'\Gamma_i\}'\chi_i))$.

Unfolding the definition of cwa gives: $ms(fol(\{'\Gamma_i\}\cup\{'(\neg\chi_i): ('\chi_i\notin(fol\{'\Gamma_i\}))\}))$

By C2 this is equivalent to: $ms(\{'\Gamma_i\}\cup\{'(\neg\chi_i): ('\chi_i\notin(fol\{'\Gamma_i\}))\})$.

Using C1 gives: $ms(\{'\Gamma_i\}\cup\{'(\neg\chi_i): \neg([(ms\{'\Gamma_i\})](mg\ '\chi_i))\})$.

Using MS3 gives: $(ms\{'\Gamma_i\})\wedge(ms\{'(\neg\chi_i):(\neg([(ms\{'\Gamma_i\})](mg\ '\chi_i)))\})$

Using MS2 twice gives: $(\forall i\Gamma_i)\wedge(ms\{'(\neg\chi_i):(\neg([(\forall i\Gamma_i)](mg\ '\chi_i)))\})$.

By MS1 this is equivalent to: $(\forall i\Gamma_i)\wedge\forall i((\neg([(\forall i\Gamma_i)](mg\ '\chi_i)))\rightarrow(mg'(\neg\chi_i)))$.

Using M0-M7 to gives: $(\forall i\Gamma_i)\wedge\forall i((\neg([(\forall i\Gamma_i)]\chi_i))\rightarrow\neg\chi_i)$ which is equivalent to: $(\forall i\Gamma_i)\wedge\forall i((<(\forall i\Gamma_i)>\neg\chi_i)\rightarrow\neg\chi_i)$.
Folding the definition of CWA gives: $(CWA\ (\forall i\Gamma_i)\chi_i)$ QED.

CWA2: $((fol\ '\kappa)=(cwa\{'\Gamma_i\}'\chi_i))\leftrightarrow((ms\ '\kappa)\equiv(CWA(\forall i\Gamma_i)\chi_i))$

proof: $(fol\ '\kappa)=(cwa\{'\Gamma_i\}'\chi_i)$.

By FOL3 this is equivalent to: $(fol\ '\kappa)=(fol(cwa\{'\Gamma_i\}'\chi_i))$.

By C4 this is equivalent to: $(ms\ '\kappa)\equiv(ms(cwa\{'\Gamma_i\}'\chi_i))$.

By CWA1 this is equivalent to: $(ms\ '\kappa)\equiv(CWA(\forall i\Gamma_i)\chi_i)$ QED.

Theorem CWA2 shows that the set of theorems: $(fol\ '\kappa)$ of a set $'\kappa$ equals cwa if and only if the meaning $(ms\ '\kappa)$ of $'\kappa$: is equivalent to CWA. Furthermore, by FOL4 it cannot be anything else (such as a set not containing all its theorems) and by MC6 there are no other solutions (such as a proposition not representable as a sentence in the FOL object language). Therefore, the Modal representation of the Closed World Assumption (i.e. CWA), faithfully represents the set theoretic description of the Closed World Assumption (i.e. cwa). Finally, we note that $(\forall i\Gamma_i)$ and $(ms\ '\kappa)$ may be generalized to be arbitrary propositions Γ and κ giving the more general modal representation: $\kappa\equiv(CWA\ \Gamma\ \chi_i)$.

7. Conclusion

Theorems CWA2, FOL4, and MC6 prove that The Closed World Assumption can be represented in a monotonic modal quantificational logic which is a slight extension of S5. Since the modal representation:

$$(CWA\ \Gamma\ \chi_i)=df\ \Gamma\wedge\forall i((<\Gamma>\neg\chi_i)\rightarrow\neg\chi_i)$$

is equivalent to the Closed World Assumption and since it an instance of the more general case where free variables may occur in the χ_i sentences:

$$(QCWA\ \Gamma\ \chi_i)\ =df\ \Gamma\wedge\forall i\forall\xi_i((<\Gamma>\neg\chi_i)\rightarrow\neg\chi_i)$$

we now have a logic for inferring consequences from that more general case where universal quantifiers may cross modal scopes. For example,

$$(QCWA\ (P\ 1)\wedge\forall x((P\ x)\rightarrow(P(add1\ x)))\ \ \neg(P\ x))$$

is equivalent to: $\forall x((P\ x)\leftrightarrow([(\pi\ 1)\wedge\forall x((\pi\ x)\rightarrow(\pi(add1\ x)))](\pi\ x)))$

which states that the natural numbers and only the natural numbers bear the property P.

Finally, we note that the techniques used herein to develop a modal representation of the Closed World Assumption and then to generalize it to handle the case where bound variables cross modal scopes, may be applied to other nonmonotonic systems such as those defined by a fixed-point equation [Brown 2003a], thereby allowing those systems to be extended to analogous cases where variables are allowed to cross the scope of nonmonotonic operations. An example of using the Modal Representation of the Closed World Assumption on multiple knowledgebases involving actions and epistemic reasoning is given in [Brown 1986].

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Numbers: MIP-9526532, EIA-9818341, and EIA-9972843.

Bibliography

[Bressan 1972] Bressan, Aldo 1972. *A General Interpreted Modal Calculus,* Yale University Press.

[Boyer&Moore 1981] R. S. Boyer and J Strother Moore, "Metafunctions: proving them correct and using them efficiently as new proof procedures," *The Correctness Problem in Computer Science,* R. S. Boyer and J. Strother Moore, eds., Academic Press, New York, 1981.

[Brown 1977] Brown, F. M., "The Theory of Meaning", Department of Artificial Intelligence Research Report 35, University of Edinburgh, June 1977, University of Edinburgh, *Edinburgh Artificial Intelligence Research Papers 1973-1986*, Scientific Datalink microfische, 1989.

[Brown 1986] Brown, Frank M. 1986. "Reasoning in a Hierarchy of Deontic Defaults", *Proceedings of the Canadian Artificial Intelligence Conference CSCI 86*, Montreal, Canada, Morgan-Kaufmann, Los Altos.

[Brown 1987] Brown, Frank M. 1987. "The Modal Logic Z", In *The Frame Problem in AI"; Proc. of the 1987 AAAI Workshop*, Morgan Kaufmann, Los Altos, CA .

[Brown 1989] Brown, Frank M. 1989. "The Modal Quantificational Logic Z Applied to the Frame Problem", advance paper *First International Workshop on Human & Machine Cognition*, May 1989 Pensacola, Florida. Abbreviated version in *International Journal of Expert Systems Research and Applications, Special Issue: The Frame Problem. Part A.* eds. Kenneth Ford and Patrick Hayes, vol. 3 number 3, pp169-206 JAI Press 1990. Reprinted in *Reasoning Agents in a Dynamic World: The Frame problem*, editors: Kenneth M. Ford, Patrick J. Hayes, JAI Press 1991.

Brown, Frank M., 2003a, "Representing Autoepistemic Logic in Modal Logic", *Information Theories and Applications*, Vol. 10, no. 4, pages 455-462, December 2003.

[Carnap 1946] Carnap, Rudolf 1946. "Modalities and Quantification" *Journal of Symbolic Logic*, vol. 11, no. 2.

[Carnap 1956] Carnap, Rudolf 1956. *Meaning and Necessity: A Study in the Semantics of Modal Logic*, The University of Chicago Press.

[Hughes & Cresswell 1968] Hughes, G. E. and Creswell, M. J., 1968. *An Introduction to Modal Logic*, Methuen & Co. Ltd., London.

[Mendelson 1964] Mendelson, E. 1964. *Introduction to Mathematical Logic*, Van Norstrand, Reinhold Co. New York.

[Parks 1976] Parks, Z. 1976. "An Investigation into Quantified Modal Logic", Studia Logica 35, p109-125.

[Quine 1969] Quine, W.V.O., Set Theory and Its Logic, revised edition, Oxford University Press, London, 1969.

[Reiter 1978] Reiter, R. 1978. "On Closed World Databases", in, *Logic and Databases*, eds. Gallaire and Minker, J., Plenum Press, Ney York.

Author's Information

Frank M. Brown – University of Kansas, Lawrence, Kansas, 66045, e-mail: brown@ku.edu.

REPRESENTING SKEPTICAL LOGICS IN MODAL LOGIC

Frank M. Brown

***Abstract**: Several skeptical nonmonotonic logics are shown to be representable in a monotonic Modal Quantificational Logic whose modal laws are stronger than S5. Specifically, it is proven that under certain conditions a set of sentences of First Order Logic is the intersection of the possibly infinite number of fixed-points of the fixed-point equation of a base nonmonotonic logic with an initial set of axioms and defaults if and only if the meaning of that set of sentences is logically equivalent to the meaning of the set of sentences entailed by the disjunction of the possibly infinite number of solutions to a necessary equivalence formed from a particular modal functor of the meanings of that initial set of sentences and of the sentences in those defaults. This result is important because the modal representation allows the use of powerful automatic deduction systems for Modal Logic and because unlike the set theoretic definition of a skeptical logic, the modal representation is easily generalized to the case quantified variables may be shared across the scope of the components of the defaults thus allowing the such defaults to produce quantified consequences.*

***Keywords**: Skeptical Fixed-point Logics, Modal Logic, Nonmonotonic Logic.*

1. Introduction

In general, a fixed-point equation defining a nonmonotonic logic may have any number of solutions including an infinite number or even none. When an equation has more than one fixed-point, the question arises as to what is actually the case, since a different answer follows for each distinct fixed-point. This question could be answered in the "skeptical" sense of not believing anything that does not hold in all the fixed-points[1] by computing the intersection of all the fixed-points: $\cap\{k: k=(\text{nml } k)\}$ where nml is a set theoretic functor defining the theorems of some nonmonotonic logic. Thus, the set of closed sentences which are skeptically believed is:

the intersection of all sets of closed sentences such that: each such set is equal to nml of itself.

The purpose of this paper is to show that all this metatheoretic machinery including the formalized syntax and the proof theory of First Order Logic (i.e. FOL), the axioms of a strong set theory, the intersection of the fixed-points of the set theoretic fixed-point equation is not needed and that the essence of a Skeptical Logic may be representable in a simple (monotonic) Modal Quantificational Logic as an expression involving a modal functor: NML which is analogous to the set theoretic functor: nml. In this case, that which is skeptically believed is:

the disjunction[2] of all propositions such that each is necessarily equivalent to NML of itself.

thereby eliminating all mention of any metatheoretic machinery.

The remainder of this paper proves that this modal representation is equivalent to the set theoretic description. Section 2 describes a formalized syntax for a FOL object language. Section 3 describes the part of the proof theory of FOL needed herein (i.e. theorems FOL1-FOL3). Section 4 describes the Intensional Semantics of FOL including the meaning operator (i.e. the laws M0-M7) and the relationship of meaning and modality to the proof theory of FOL (i.e. the laws R0, A1, A2 and A3 and the theorems C1, C2, C3, and C4). The modal version of Skepticity, called S, is defined in section 5 and explicated with theorems MSK1-MSK3 and SS1. In section 6, this modal version is shown by theorems SL1 to be equivalent to the set theoretic representation of a Skeptical Logic. Figure 1 outlines the relationship of all these theorems in producing the final theorems SL1 and MSK3. In section 7 these theorems are applied to producing the modal representations of the Skeptical Logics constructed from FOL, the Closed World Assumption [Reiter 1978], Reflective Logic [Brown 1989], Default Logic [Reiter 1980], the recursive definition of Default Logic [Reiter 1980], and Autoepistemic Logic [Moore 1985]. Finally, in section 8, some conclusions are made.

[1] An alternative answer is the "credulous" sense of believing anything that holds in any of the fixed-points.
[2] Just as the word "intersection" in the set theoretic description encompasses an infinite and even a nondenumerable number of sets, our usage of the word "disjunction" encompasses an infinite and even a nondenumerable number of propositions.

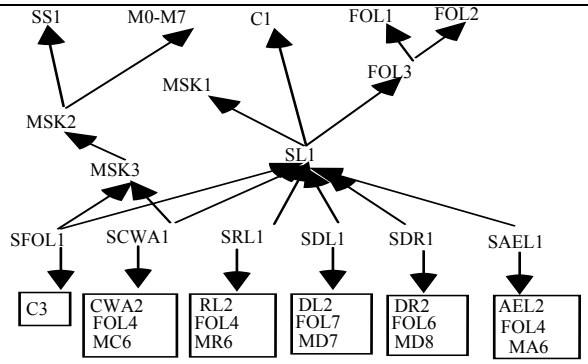


Figure 1: Dependences among the Theorems
(The theorems in the boxes are proven in other papers as noted in section 7.)

2. Formal Syntax of First Order Logic

We use a First Order Logic (i.e. FOL) defined as the six tuple: ($\rightarrow$, #f, $\forall$, *vars*, *predicates*, *functions*) where $\rightarrow$, #f, and $\forall$ are logical symbols, *vars* is a set of variable symbols, *predicates* is a set of predicate symbols each of which has an implicit arity specifying the number of associated terms, and *functions* is a set of function symbols each of which has an implicit arity specifying the number of associated terms. The sets of logical symbols, variables, predicate symbols, and function symbols are pairwise disjoint. Lower case Roman letters possibly indexed with digits are used as variables. Greek letters possibly indexed with digits or lowercase roman letters are used as syntactic metavariables. γ, γ_1,...γ_n, range over the variables, ξ, ξ_1...ξ_n range over sequences of variables of an appropriate arity, π,π_1...π_n range over the predicate symbols, ϕ, ϕ_1...ϕ_n range over function symbols, δ, δ_1...δ_n, σ range over terms, and α, α_1...α_n, β, β_1...β_n,χ, χ_1...χ_n,Γ,Γ_1,...Γ_n,φ range over sentences. The terms are of the forms γ and (ϕ δ_1...δ_n), and the sentences are of the forms ($\alpha\rightarrow\beta$), #f, ($\forall\gamma$ α), and (π δ_1...δ_n). A nullary predicate π or function ϕ is written as a sentence or a term without parenthesis. $\varphi\{\pi/\lambda\xi\alpha\}$ represents the replacement of all unmodalized occurrences of π in φ by $\lambda\xi\alpha$ followed by lambda conversion. The primitive symbols are shown in Figure 2 with their intuitive interpretations.

Symbol	Meaning
$\alpha \rightarrow \beta$	if α then β.
#f	falsity
$\forall\gamma\ \alpha$	for all γ, α.

Figure 2: Primitive Symbols of First Order Logic

The defined symbols are listed in Figure 3 with their definitions and intuitive interpretations.

Symbol	Definition	Meaning		Symbol	Definition	Meaning
$\neg\alpha$	$\alpha \rightarrow$ #f	not α		$\alpha\wedge\beta$	$\neg(\alpha \rightarrow \neg\beta)$	α and β
#t	$\neg$ #f	truth		$\alpha\leftrightarrow\beta$	$(\alpha\rightarrow\beta) \wedge (\beta\rightarrow\alpha)$	α if and only if β
$\alpha\vee\beta$	$(\neg\alpha)\rightarrow\beta$	α or β		$\exists\gamma\ \alpha$	$\neg\forall\gamma\ \neg\alpha$	for some γ , α

Figure 3: Defined Symbols of First Order Logic

The FOL object language expressions are referred in the metalanguage (which also includes a FOL syntax) by inserting a quote sign in front of the object language entity thereby making a structural descriptive name of that entity. Generally, a set of sentences is represented as: $\{'\Gamma_i\}$ which is defined as: $\{'\Gamma_i$: #t$\}$ which in turn is defined as: $\{s: \exists i(s='\Gamma_i)\}$ where i ranges over some range of numbers (which may be finite or infinite). With a slight abuse of notation we also write $'\kappa$, $'\Gamma$ to refer to such sets.

3. Proof Theory of First Order Logic

The axioms and inference rules of FOL [Mendelson 1964] are given in Figure 4:

MA1: $\alpha \rightarrow (\beta \rightarrow \alpha)$ MR1: from α and $(\alpha \rightarrow \beta)$ infer β

MA2: $(\alpha \rightarrow (\beta \rightarrow \rho)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \rho))$ MR2: from α infer $(\forall \gamma\ \alpha)$

MA3: $((\neg \alpha) \rightarrow (\neg \beta)) \rightarrow (((\neg \alpha) \rightarrow \beta) \rightarrow \alpha)$

MA4: $(\forall \gamma\ \alpha) \rightarrow \beta$ where β is the result of substituting an expression (which is free for the free positions of γ in α) for all the free occurrences of γ in α.

MA5: $(\forall \gamma(\alpha \rightarrow \beta)) \rightarrow (\alpha \rightarrow (\forall \gamma\ \beta))$ where γ does not occur in α.

Figure 4: Inferences Rules and Axioms of FOL

In order to talk about sets of sentences we include in the metatheory set theory symbols ε, $\notin$, $\supseteq$, $=$, $\cap$ as defined in [Quine 1969]. The derivation operation (i.e. fol) of FOL obeys the Inclusion and Idempotence properties:

FOL1: (fol 'κ)$\supseteq$'κ Inclusion

FOL2: (fol 'κ)$\supseteq$(fol(fol 'κ)) Idempotence

From these two properties we prove:

FOL3: $\forall p((p=(\text{fol } p)) \rightarrow \alpha) \leftrightarrow \forall p(\alpha\{p/(\text{fol } p)\})$ and $\exists p((p=(\text{fol } p)) \wedge \alpha) \leftrightarrow \exists p(\alpha\{p/(\text{fol } p)\})$

proof: The universal quantifier version follows from the existential quantifier version by running negation through both sides of the bi-implication. The existential version is proven as follows. There are two cases:

(1) $((p=(\text{fol } p)) \wedge \alpha) \rightarrow \exists\ p(\alpha\{p/(\text{fol } p)\})$. The existentially quantified p is replaced by p giving: $((p=(\text{fol } p)) \wedge \alpha) \rightarrow (\alpha\{p/(\text{fol } p)\})$. The equation in the hypothesis is used to replace p in α by (fol p) giving the conclusion.

(2) $(\alpha\{p/(\text{fol } p)\}) \rightarrow \exists p((p=(\text{fol } p)) \wedge \alpha)$. Letting p in the conclusion be (fol p) gives: $(\alpha\{p/(\text{fol } p)\}) \rightarrow (((\text{fol } p)=(\text{fol}(\text{fol } p))) \wedge (\alpha\{p/(\text{fol } p)\}))$ which holds by FOL1 and FOL2.

4. Intensional Semantics of FOL

The meaning (i.e. mg) [Brown 1977] or rather disquotation of a sentence of FOL is defined in Figure 5 below. mg is defined in terms of mgs which maps each FOL object language sentence and an association list into a meaning. mgn maps each FOL object language term and an association list into a meaning. An association list is a list of pairs consisting of an object language variable and the meaning to which it is bound.

M0: (mg 'α) =df (mgs '$(\forall \gamma 1 \ldots \gamma n\ \alpha)$'()) where '$\gamma 1 \ldots$'$\gamma n$ are all the free variables in 'α

M1: (mgs '$(\alpha \rightarrow \beta)$a) $\leftrightarrow$ ((mgs 'α a)$\rightarrow$(mgs 'β a))

M2: (mgs '#f a) $\leftrightarrow$ #f

M3: (mgs '$(\forall\ \gamma\ \alpha)$a) $\leftrightarrow \forall$x(mgs 'α(cons(cons 'γ x)a))

M4: (mgs '$(\pi\ \delta_1 \ldots \delta_n)$a) $\leftrightarrow$ (π(mgn 'δ_1 a)...(mgn 'δ_n a)) for each predicate symbol 'π.

M5: (mgn '$(\phi\ \delta_1 \ldots \delta_n)$a) = ($\phi$(mgn '$\delta_1$ a)...(mgn 'δ_n a)) for each function symbol 'ϕ.

M6: (mgn 'γ a) = (cdr(assoc 'γ a))

M7: (assoc v L)= (if(eq? v(car(car L)))(car L)(assoc v(cdr L))) where: cons, car, cdr, eq?, and if are as in Scheme.

Figure 5: The Meaning of FOL Sentences

The meaning operator (i.e. mg) is a disquotation operation: (mg 'α)$\leftrightarrow\alpha$. It may be used as an Intensional Semantics for a FOL object language by axiomatizing the modal concept of necessity to satisfy the theorem:

C1: ('$\alpha\varepsilon$(fol 'κ)) $\leftrightarrow$ ([] ((ms 'κ)$\rightarrow$(mg 'α)))

for every sentence 'α and every set of sentences 'κ of that FOL object language. The necessity symbol is represented by a box: []. C1 states that a sentence of FOL is a FOL-theorem (i.e. fol) of a set of sentences of FOL if and only if the meaning of that set of sentences necessarily implies the meaning of that sentence.

One modal logic which satisfies C1 for FOL is the Z Modal Quantificational Logic described in [Brown 1987; Brown 1989] whose theorems are recursively enumerable. Z has the metatheorem: $(<>\Gamma\{\pi/\lambda\xi\alpha\})\rightarrow(<>\Gamma)$ where Γ is a sentence of FOL . Z includes all the laws of S5 Modal Logic [Hughes & Cresswell 1968] whose modal axioms and inference rules are in Figure 6. Therein, κ and Γ are sentences of the intentional semantics.

R0: from κ infer $([]\kappa)$	A2: $([](\kappa\rightarrow\Gamma))\rightarrow(([]\kappa)\rightarrow([]\Gamma))$
A1: $([]\kappa)\rightarrow\kappa$	A3: $([]\kappa)\vee([]\neg[]\kappa)$

Figure 6: The Laws of S5 Modal Logic

These S5 modal laws and the laws of FOL given in Figure 6 constitute an S5 Modal Quantificational Logic similar to [Carnap 1946; Carnap 1956], and a FOL version [Parks 1976] of [Bressan 1972] in which the Barcan formula: $(\forall\gamma([]\kappa))\rightarrow([]\forall\gamma\kappa)$ and its converse hold. The defined Modal symbols are given in Figure 7.

Symbol	Definition	Meaning		Symbol	Definition	Meaning
$<>\kappa$	$\neg[]\neg\kappa$	α is logically possible		$[\kappa]\Gamma$	$[](\kappa\rightarrow\Gamma)$	β entails α
$\kappa\equiv\Gamma$	$[](\kappa\leftrightarrow\Gamma)$	α is logically equivalent to β		$<\kappa>\Gamma$	$<>(\kappa\wedge\Gamma)$	α and β is logically possible

Figure 7: Defined Symbols of Modal Logic

5. Skepticity Represented in Modal Logic

The set theoretic intersection defining a Skeptical Logic in terms of a base nonmonotonic logic called nml:

$$\cap\{k: k=(\text{nml } k)\}$$

may be expressed in an S5 Modal Quantificational Logic supplemented with propositional quantifiers [Fine 1970; Bressan 1972] which obey the normal laws of Second Order Logic (i.e. laws analogous to MR2, MA4, and MA5 given in Figure 4 where γ is now a propositional variable), as follows:

$$\exists k(k\wedge(k\equiv(\text{NML } k)))$$

where NML is the modal functor corresponding to nml. The idiom $\exists k(k\wedge(k\equiv(\varphi\ k)))$ is intuitively read as a nominal as the (possibly infinite) disjunction of all propositions such that κ_i. When the necessary equivalence has a finite number of solutions: $\kappa_1,...,\kappa_\eta$ then the idiom is equivalent to: $\kappa_1\vee...\vee\kappa_\eta$, but there is in general no requirement that the necessary equivalence holds for only a finite or even only a denumerable number of solutions.

Some theorems are now proven which show that reasoning about meanings of sets of sentences is equivalent to reasoning about propositions in general. MSK1 shows (in the cases of interest) that quantifying over the meanings of sets of FOL sentences is equivalent to quantifying propositions in general:

MSK1 if $(\kappa\equiv(\text{NML }\kappa))\rightarrow\exists s(\kappa\equiv(\text{ms } s))$ then $(\exists k((\text{ms } k)\wedge((\text{ms } k)\equiv(\text{NML}(\text{ms } k)))))\equiv(\exists k(k\wedge(k\equiv(\text{NML } k))))$

proof: By R0 it suffices to prove: $(\exists k((\text{ms } k)\wedge((\text{ms } k)\equiv(\text{NML}(\text{ms } k)))))\leftrightarrow\exists k(k\wedge(k\equiv(\text{NML } k)))$. There are two cases: (1) $\exists k((\text{ms } k)\wedge((\text{ms } k)\equiv(\text{NML}(\text{ms } k))))\rightarrow\exists k(k\wedge(k\equiv(\text{NML } k)))$. By FOL this is equivalent to:

$((\text{ms } k)\wedge((\text{ms } k)\equiv(\text{NML}(\text{ms } k))))\rightarrow\exists k(k\wedge(k\equiv(\text{NML } k)))$. Letting k in the conclusion be (ms k) gives a tautology.

(2) $\exists k(k\wedge(k\equiv(\text{NML } k)))\rightarrow\exists k((\text{ms } k)\wedge((\text{ms } k)\equiv(\text{NML}(\text{ms } k)))))$. By FOL this is :$(k\wedge(k\equiv(\text{NML } k)))$

$\rightarrow(\exists k((\text{ms } k)\wedge((\text{ms } k)\equiv(\text{NML}(\text{ms } k)))))$. From $(k\equiv(\text{NML } k))$ and the hypothesis to the theorem we infer $\exists s(\kappa\equiv(\text{ms } s))$ and then $\kappa\equiv(\text{ms } s)$. Using this to replace k by (ms s) gives: $((\text{ms } s)\wedge((\text{ms } s)\equiv(\text{NML}(\text{ms } s))))\rightarrow(\exists k((\text{ms } k)\wedge((\text{ms } k)\equiv(\text{NML}(\text{ms } k)))))$. Letting k in the conclusion be s then gives a tautology. QED.

The concept (i.e. ss) of the combined meaning of all the sentences of the FOL object language whose meanings are entailed by a proposition is defined as follows: $(\text{ss }\kappa)=_{df}\forall s(([\kappa](\text{mg } s))\rightarrow(\text{mg } s))$. SS1 shows that a proposition entails the combined meaning of the FOL object language sentences that it entails.

SS1: $[\kappa](\text{ss }\kappa)$ proof: By R0 it suffices to prove: $\kappa\rightarrow(\text{ss }\kappa)$. Unfolding ss gives: $\kappa\rightarrow\forall s(([\kappa](\text{mg } s))\rightarrow(\text{mg } s))$ which is equivalent to:$\forall s(([\kappa](\text{mg } s))\rightarrow(\kappa\rightarrow(\text{mg } s)))$ which is an instance of A1. QED.

One might think that: $(\text{ss}(\exists k(k\wedge(k\equiv(\text{NML } k))))\leftrightarrow(\exists k(k\wedge(k\equiv(\text{NML } k)))$, but in general $\text{ss}(\exists k(k\wedge(k\equiv(\text{NML } k)))$ does not imply: $\exists k(k\wedge(k\equiv(\text{NML } k))$ because if there were an infinite number of solutions to: $k\equiv(\text{NML } k))$ none of which

was entailed by the others, then that disjunction of all the solutions would be infinite and therefore might not be representable as a sentence of FOL. However, if there are only a finite number of solutions then this relation will hold as is proven by MSK2 below:

MSK2: If k does not occur in any 'Γi then: $(ss(\exists k(k\wedge(\vee_{i=1,n}(k\equiv(ms\ '\Gamma i))))))\equiv(\vee_{i=1,n}(ms\ '\Gamma_i))$

proof: By SS1 it suffices to prove: $[(ss(\exists k(k\wedge(\vee_{i=1,n}(k\equiv(ms\ '\Gamma_i))))))](\vee_{i=1,n}(ms\ '\Gamma_i))$

By the laws of FOL this is equivalent to: $[(ss(\vee_{i=1,n}(\exists k(k\wedge(k\equiv(ms\ '\Gamma_i))))))](\vee_{i=1,n}(ms\ '\Gamma_i))$

Since k does not occur free in Γ_i by the laws of S5 this is equivalent to: $[(ss(\vee_{i=1,n}(ms\ '\Gamma_i)))](\vee_{i=1,n}(ms\ '\Gamma_i))$. The last ms is unfolded giving: $[ss(\vee_{i=1,n}(ms\ '\Gamma_i))](\vee_{i=1,n}(\forall s((s\varepsilon'\Gamma_i)\rightarrow(mg\ s))))$. By C1 this is equivalent to: $[ss(\vee_{i=1,n}(ms\ '\Gamma_i))](\vee_{i=1,n}(\forall s(([(ms\ '\Gamma_i)](mg\ s))\rightarrow(mg\ s))))$. Renaming and pulling out the universal quantifiers gives: $[ss(\vee_{i=1,n}(ms\ \Gamma_i))]\forall s_1...\forall s_n(\vee_{i=1,n}(([(ms\ '\Gamma_i)](mg\ s_i))\rightarrow(mg\ s_i)))$. This is equivalent to:

$[ss(\vee_{i=1,n}(ms\ \Gamma_i))]\forall s_1...\forall s_n((\wedge_{i=1,n}([(ms\ '\Gamma_i)](mg\ s_i)))\rightarrow(\vee_{i=1,n}(mg\ s_i)))$ By the mg laws this is equivalent to: $[ss(\vee_{i=1,n}(ms\ \Gamma_i))]\forall s_1...\forall s_n((\wedge_{i=1,n}([(ms\ '\Gamma_i)](mg\ s_i)))\rightarrow(mg\ `(\vee_{i=1,n}\ ,s_i)))$. Unfolding ss gives:

$[(\forall s(([(\vee_{i=1,n}(ms\ \Gamma_i))](mg\ s))\rightarrow(mg\ s))]\ \forall s_1...\forall s_n((\wedge_{i=1,n}([(ms\ \Gamma_i)](mg\ s_i)))\rightarrow(mg'(\vee_{i=1,n}\ ,s_i)))$

Backchaining letting $s:=(\vee_{i=1,n}\ ,s_i)$ shows that it suffices to prove:

$(\forall s_1...\forall s_n((\wedge_{i=1,n}([(ms\ \Gamma_i)](mg\ s_i)))\rightarrow([(\vee_{i=1,n}(ms\ \Gamma_i))](mg\ `(\vee_{i=1,n}\ ,s_i))))$.

In view of the hypothesis it suffices to prove: $([(\vee_{i=1,n}(mg\ s_i))](mg\ `(\vee_{i=1,n}\ ,s_i)))$.

By the mg laws this is equivalent to: $([(\vee_{i=1,n}(mg\ s_i))](\vee_{i=1,n}(mgs_i)))$ which is a tautology. QED.

MSK3 is the instance of MSK2 with only one solution:

MSK3 $(ss(\exists k(k\wedge(k\equiv(ms\ '\Gamma)))))\equiv(ms\ '\Gamma)$. and $(ss(ms\ '\Gamma))\equiv(ms\ '\Gamma)$ proof: Let n=1 in theorem MSK2.

6. The Relationship between Skeptical Logics and Modal Logic

The relationship between the proof theoretic definition of Skepticity and the modal representation is proven with theorem SL1. Theorem SL1 shows that the meaning of the intersection of all the fixed-points of the set theoretic functor nml is the meaning of the sentences entailed by the "disjunction" of the solutions of the Modal functor NML provided that hypotheses H1, H2, and H3 (listed therein) hold.

SL1: Let nml be a set theoretic functor and let NML be a modal functor. If the following three hypotheses hold:

H1: $('\kappa=(nml\ '\kappa))\rightarrow('\kappa=(fol\ '\kappa))$

H2: $((fol\ '\kappa)=(nml(fol\ '\kappa)))\leftrightarrow((ms\ '\kappa)\equiv(NML(ms\ '\kappa)))$

H3: $(\kappa\equiv(NML\ \kappa))\rightarrow\exists s(\kappa\equiv(ms\ s))$

then: $(ms(\cap\{k:\ k=(nml\ k)\}))\equiv(ss(\exists k(k\wedge k\equiv(NML\ k))))$

proof: $ms(\cap\{k:\ k=(nml\ k)\})$. By hypothesis H1 this is equivalent to: $ms(\cap\{k:\ (k=(fol\ k))\wedge(k=(nml\ k))\})$

Unfolding the definition of intersection gives: $ms\{s:\forall k((k\varepsilon\{k:\ (k=(fol\ k))\wedge(k=(nml\ k))\})\rightarrow(s\varepsilon k))\}$ which is equivalent to: $ms\{s:\forall'k(((k=(fol\ k))\wedge(k=(nml\ k)))\rightarrow(s\varepsilon k))\}$. By FOL3 this is equivalent to: $ms\{s:\forall'k(((fol\ k)=(nml(fol\ k))))\rightarrow(s\varepsilon(fol\ k)))\}$. By the Hypothesis H2, this is equivalent to: $ms\{s:\forall k(((ms\ k)\equiv(NML(ms\ k)))\rightarrow(s\varepsilon(fol\ k))))\}$

By C1 this is equivalent to: $ms\{s:\forall k(((ms\ k)\equiv(NML(ms\ k)))\rightarrow([ms\ k](mg\ s)))\}$. By the S5 laws pushing $\forall k$ to lowest scope gives: $ms\{s:\ ([\exists k((ms\ k)\wedge((ms\ k)\equiv(NML(ms\ k)))))](mg\ s))\}$

Hypothesis H3 is equivalent to the hypothesis of theorem MSK1 therefore by the conclusion of MSK1 the above expression is equivalent to: $ms\{s:\ ([\exists k(k\wedge(k\equiv(NML\ k)))](mg\ s))\}$. Unfolding ms and lambda conversion gives: $\forall s(([\exists k(k\wedge(k\equiv(NML\ k)))](mg\ s))\rightarrow(mg\ s))$. Folding ss gives $ss(\exists k(k\wedge(k\equiv(NML\ k))))$ QED.

Theorem SL1 shows that the modal representation of Skepticity using ss is logically equivalent to the set theoretic representation. $ss(\exists k(k\wedge(k\equiv(NML\ k))))$ may be generalized to: $(\exists k(k\wedge(k\equiv(NML\ k))))$ since the latter entails the same FOL object language sentences.

7. Example Relationships between Skeptical Logics and Modal Logics

Figure 8 gives the set theoretic representations of six Skeptical Logics in terms of their base logics. The six base logics are FOL: (i.e. fol), The Closed World Assumption (i.e. cwa) [Reiter 1978], Reflective Logic (i.e. rl) [Brown 1989], Default Logic (i.e. dl) [Reiter 1980], the "recursive" definition of Default Logic [Reiter 1980], and Autoepistemic Logic (i.e. ael) [Moore 1986]. The first two Skeptical Logics are defined with degenerate equations of the form k=(nml) where nml does not contain k, and thus are identical to their base Logics. These exemplify the theorems in section 6 relating set theoretic descriptions of Skeptical Logics to their corresponding modal descriptions. The last four Skeptical Logics involve the more interesting nondegenerate cases.

Skeptical Logic	Skeptical Logic constructed from the Base Logic
sfol	$\cap\{k: k=(\text{fol } '\Gamma)\}$ or $(\text{fol } '\Gamma)$
scwa	$\cap\{k: k=(\text{cwa } '\Gamma\ '\chi_i)\}$ or $(\text{cwa } '\Gamma\ '\chi_i)$where: $(\text{cwa}'\Gamma'\chi_i)=\text{df fol}('\Gamma\cup\{'\chi_i:'(\neg\chi_i)\notin(\text{fol } '\Gamma)\})$
srl	$\cap\{k: k=(\text{rl k } '\Gamma\ '\alpha_i:'\beta_{ij}/'\chi_i)\}$where: $(\text{rl } '\kappa\ '\Gamma\ '\alpha_i:'\beta_{ij}/'\chi_i)=\text{df fol}('\Gamma\cup\{'\chi_i:('\alpha_i\varepsilon'\kappa)\wedge\wedge_{j=1,mi}'(\neg\beta_{ij})\notin'\kappa\})$
sdl	$\cap\{k: k=(\text{dl k } '\Gamma\ '\alpha_i:'\beta_{ij}/'\chi_i)$ where: $(\text{dl } '\kappa\ '\Gamma\ '\alpha_i:'\beta_{ij}/'\chi_i)=\text{df}\cap\{p:(p\supseteq(\text{fol } p))\wedge(p\supseteq\Gamma)\wedge\forall_i((('\alpha_i\varepsilon p)\wedge\wedge_{j=1,mi}'(\neg\beta_{ij})\notin'\kappa)\rightarrow('\chi_i\varepsilon p))\}$
sdr	$\cap\{k: '\kappa=(\text{dr } '\kappa\ '\Gamma\ '\alpha_i:'\beta_{ij}/'\chi_i)\}$ where: $(\text{dr } '\kappa\ '\Gamma\ '\alpha_i:'\beta_{ij}/'\chi_i)=\text{df}\cup_{t=1,\omega}(\text{r t } '\kappa\ '\Gamma\ '\alpha_i:'\beta_{ij}/'\chi_i)$ $(\text{r } 0\ '\kappa\ '\Gamma\ '\alpha_i:'\beta_{ij}/'\chi_i)=\text{df (fol } '\Gamma)$ $(\text{r t+1 } 'k)=\text{df (fol}((\text{r t } '\kappa\ '\Gamma\ '\alpha_i:'\beta_{ij}/'\chi_i)\cup\{'\chi_i:('\alpha_i\varepsilon(\text{r t } '\kappa\ '\Gamma\ '\alpha_i:'\beta_{ij}/'\chi_i))\wedge\wedge_{j=1,mi}'(\neg\beta_{ij})\notin'\kappa)\}))$
sael	$\cap\{k: k=(\text{ael k } \Gamma)\}$ where: $(\text{ael } '\kappa\ '\Gamma)=\text{df (fol}('\Gamma\cup\{'(L'\chi_i):'\chi_i\varepsilon'\kappa\}\cup\{'(\neg(L'\chi_i)):'\chi_i\notin'\kappa\}))$

Figure 8: Set Theoretic Representations of Skeptical Logics

Figure 9 gives the Modal Logic representations of six Skeptical Logics in terms of their base logic. The six base logics are the modal representations of FOL (i.e. the identity operator), The Closed World Assumption (i.e. CWA) [Brown 2005], Reflective Logic (i.e. RL) [Brown 2003a], Default Logic (i.e. DL) [Brown 2003b], "recursive" definition of Default Logic [Brown 2004], and Autoepistemic Logic (i.e. AEL) [Moore 2003c].

Skeptical Logic	Skeptical Logic constructed from the Base Logic
SFOL	$\exists k(k\wedge(k\equiv(\text{FOL } \Gamma)$ or Γ where: $(\text{FOL } \Gamma)=\text{df } \Gamma$
SCWA	$\exists k(k\wedge(k\equiv(\text{CWA } \Gamma\ \chi_i)))$ or $(\text{CWA } \Gamma\ \chi_i)$ where: $(\text{CWA } \Gamma\ \chi_i)=\text{df } \Gamma\wedge\forall_i((<\Gamma>\chi_i)\rightarrow\chi_i)$
SRL	$\exists k(k\wedge(k\equiv(\text{RL } \kappa\ \Gamma\ \alpha_i:\beta_{ij}/\chi_i)$ where:$(\text{RL k } \Gamma\ \alpha_i:\beta_{ij}/\chi_i)=\text{df } \Gamma\wedge\forall_i((([\kappa]\alpha_i)\wedge\wedge_{j=1,mi}(<\kappa>\beta_{ij}))\rightarrow\chi_i)$
SDL	$\exists k(k\wedge(k\equiv(\text{DL k } \Gamma\ \alpha_i:\beta_{ij}/\chi_i))$ where: $(\text{DL } \kappa\ \Gamma\ \alpha_i:\beta_{ij}/\chi_i)=\text{df } \exists p(p\wedge([p]\Gamma)\wedge\forall_i((([p]\alpha_i)\wedge\wedge_{j=1,mi}<\kappa>\beta_{ij})\rightarrow[p]\chi_i))$
SDR	$\kappa\equiv(\text{DR } \kappa\ \Gamma\ \alpha_i:\beta_{ij}/\chi_i)$ where DR is defined as: $(\text{DR } \kappa\ \Gamma\ \alpha_i:\beta_{ij}/\chi_i)=\text{df }\forall t(\text{R t k } \Gamma\ \alpha_i:\beta_{ij}/\chi_i)$ $(\text{R } 0\ \text{k } \Gamma\ \alpha_i:\beta_{ij}/\chi_i)=\text{df } \Gamma$ $(\text{R t+1 k } \Gamma\ \alpha_i:\beta_{ij}/\chi_i)=\text{df (R t k } \Gamma\ \alpha_i:\beta_{ij}/\chi_i)\wedge\forall_i((([(\text{R t k } \Gamma\ \alpha_i:\beta_{ij}/\chi_i)]\alpha_i)\wedge\wedge_{j=1,mi}(<\kappa>\beta_{ij}))\rightarrow\chi_i)$
SAEL	$\exists k(k\wedge(k\equiv(\text{AEL k } \Gamma))$ where: $(\text{AEL } \kappa\ \Gamma)=\text{df } \Gamma\wedge\forall i((\text{L } '\chi_i)\leftrightarrow([\kappa]\chi_i))$

Figure 9: Modal Representations of Skeptical Logics

The skeptical logics defined in set theory are now related to the corresponding skeptical logics defined in Modal Logic. The following theorems show that the two representation coincide

SFOL1: $(\text{ms}(\cap\{k: k=(\text{fol } '\Gamma)\}))\equiv(\text{ss}(\exists k(k\wedge(k\equiv(\text{FOL(ms } '\Gamma)))))$

proof: Letting nml be fol and NML be the identity operator makes the three hypotheses to theorem SL3 true:

(1) $('\kappa=(\text{fol } '\Gamma))\rightarrow('\kappa=(\text{fol } '\kappa))$

(2) $((\text{fol } '\kappa)=(\text{fol(fol } '\Gamma)))\leftrightarrow((\text{ms } '\kappa)\equiv(\text{FOL(ms } '\Gamma)))$

(3) $(\kappa\equiv(\text{FOL(ms } '\Gamma)))\rightarrow\exists s(\kappa\equiv(\text{ms s}))$

FOL is the identity functor. (1) is a tautology. Letting s be 'Γ makes (3) a tautology. The left hand of (2) is equivalent to: ((fol 'κ)=(fol 'Γ)). By C4 proven in [Brown 2005] this holds. The conclusion of SL1 with the variable nml instantiated to fol and the variable NML instantiated to FOL must therefore be true. QED. Since k does not occur in (FOL(ms 'Γ)), (k≡(FOL(ms 'Γ)) has only one solution and theorem MSK3 allows SFOL1 to be rewritten as: (ms(fol 'Γ))≡(FOL(ms 'Γ)).

SCWA1: (ms($\cap${k: k=(cwa 'Γ 'χ_i)}))≡(ss($\exists$k(k$\wedge$(k≡(CWA(ms 'Γ)χ_i)))))

proof: Letting nml be cwa [Reiter 1978] and NML be CWA makes the three hypotheses of theorem SL3 true. These three hypotheses were proven as theorems FOL4, CWA2, and MC6 in [Brown 2005]:

(1) FOL4: ('κ=(cwa 'Γ 'χ_i))$\rightarrow$('κ=(fol 'κ))

(2) CWA2: ((fol 'κ)=(cwa(fol 'Γ)'χ_i)) $\leftrightarrow$((ms 'κ)≡(CWA(ms 'Γ)χ_i))

(3) MC6: (κ≡(CWA(ms 'Γ)(mg 'χ_i)))$\rightarrow\exists$s(κ≡(ms s))

The conclusion of SL1 with the variable nml instantiated to cwa and the variable NML instantiated to CWA must therefore be true. QED. Since k does not occur in (CWA(ms 'Γ)χ_i)), (k≡(CWA(ms 'Γ)χ_i)) has only one solution and theorem MSK3 allows SCWA1 to be rewritten as: (ms(cwa 'Γ 'χ_i))≡(CWA(ms '$\Gamma\chi_i$)).

SRL1: (ms($\cap${'k: 'k=(rl 'k 'Γ 'α_i:'β_{ij}/'χ_i)}))≡(ss($\exists$k(k$\wedge$k≡(RL k(ms 'Γ)α_i:β_{ij}/'χ_i))))

proof: Letting nml be rl [Brown 1989] and NML be RL makes the three hypotheses of theorem SL3 true. These three hypotheses were proven as theorems FOL4, RL2, and MR6 in [Brown 2003a]:

(1) FOL4: ('κ=(rl 'κ 'Γ 'α_i:'β_{ij}/'χ_i))$\rightarrow$('κ=(fol 'κ))

(2) RL2: (ms(rl(fol 'κ)'Γ 'α_i:'β_{ij}/'χ_i))≡(RL(ms 'κ)(ms 'Γ)α_i:β_{ij}/χ_i))

(3) MR6: (κ≡(RL κ(ms 'Γ)α_i:β_{ij}/(mg 'χ_i)))$\rightarrow\exists$s(κ≡(ms s))

The conclusion of SL1 with nml instantiated to rl and NML instantiated to RL must therefore be true. QED.

SDL1: (ms($\cap${'k: 'k=(dl 'k 'Γ 'α_i:'β_{ij}/'χ_i)}))≡(ss($\exists$k(k$\wedge$k≡(DL k(ms 'Γ)α_i:β_{ij}/χ_i))))

proof: Letting nml be dl and NML be the DL [Reiter 1980] makes the three hypotheses of theorem SL3 true. These three hypotheses were proven as theorems FOL6, DL2, and MD7 in [Brown 2003b]:

(1) FOL6: ('k=(dl 'κ 'Γ 'α_i:'β_{ij}/'χ_i))$\rightarrow$('κ=(fol 'κ))

(2) DL2: ((fol 'κ)=(dl(fol 'κ)'Γ 'α_i:'β_{ij}/'χ_i)) $\leftrightarrow$((ms 'κ)≡(DL(ms 'κ)(ms 'Γ)α_i:β_{ij}/χ_i)))

(3) MD7: (κ≡(DL κ(ms 'Γ)α_i:β_{ij}/(mg 'χ_i)))$\rightarrow\exists$s(κ≡(ms s))

The conclusion of SL1 with nml instantiated to dl and NML instantiated to DL must therefore be true. QED.

SDR1:(ms($\cap${'k: 'k=(dl 'k 'Γ 'α_i:'β_{ij}/'χ_i)}))≡(ss($\exists$k(k$\wedge$k≡(DL k(ms 'Γ)α_i:β_{ij}/χ_i))))

proof: Letting nml be dl [Reiter 1980] and NML be the DL makes the three hypotheses of theorem SL3 true. These three hypotheses were proven as theorems FOL6, DL2, and MD7 in [Brown 2004]:

(1) FOL6: ('k=(dr 'κ 'Γ 'α_i:'β_{ij}/'χ_i))$\rightarrow$('κ=(fol 'κ))

(2) DR2: ((fol 'κ)=(dr(fol 'κ)'Γ 'α_i:'β_{ij}/'χ_i)) $\leftrightarrow$((ms 'κ)≡(DL(ms 'κ)(ms 'Γ)α_i:β_{ij}/χ_i)))

(3) MD8: (κ≡(DR κ(ms 'Γ)α_i:β_{ij}/(mg 'χ_i)))$\rightarrow\exists$s(κ≡(ms s))

The conclusion of SL1 with nml instantiated to dr and NML instantiated to DR must therefore be true. QED.

SAEL1: (ms($\cap${k: k=(ael k 'Γ)}))≡(ss($\exists$k(k$\wedge$(k≡(AEL k(ms 'Γ))))))

proof: Letting nml be ael [Moore 1986] and NML be AEL makes the three hypotheses of theorem SL true. These three hypotheses were proven as theorems FOL4, AEL2, and MA6 in [Brown 2003c]:

(1) FOL4: ('κ=(ael 'κ'Γ))$\rightarrow$('κ=(fol 'κ))

(2) AEL2: ((fol 'κ)=(ael(fol 'κ)'Γ)) $\leftrightarrow$((ms 'κ)≡(AEL(ms 'κ)(ms 'Γ)))

(3) MA6: (κ≡(AEL κ(ms 'Γ)))$\rightarrow\exists$s(κ≡(ms s))

The conclusion of SL1 with nml instantiated to ael and NML instantiated to AEL must therefore be true. QED.

8. Conclusion

Theorem SL1 proves and section 7 exemplifies that many skeptical nonmonotonic logics can all be represented in a single modal quantificational logic which is a slight extension of S5.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Numbers: MIP-9526532, EIA-9818341, and EIA-9972843.

Bibliography

[Bressan 1972] Bressan, Aldo 1972. *A General Interpreted Modal Calculus,* Yale University Press.

[Brown 1977] Brown, F. M., "The Theory of Meaning", Dept. of Artificial Intelligence Research Report 35, Univ. of Edinburgh, June 1977, *Edinburgh Artificial Intelligence Research Papers 1973-1986*, Scientific Datalink microfische, 1989.

[Brown 1986] Brown, Frank M. 1986. "Reasoning in a Hierarchy of Deontic Defaults", *Proceedings of the Canadian Artificial Intelligence Conference CSCI 86*, Montreal, Canada, Morgan-Kaufmann, Los Altos.

[Brown 1987] Brown, Frank M. 1987. "The Modal Logic Z", In *The Frame Problem in AI"; Proc. of the 1987 AAAI Workshop*, Morgan Kaufmann, Los Altos, CA .

[Brown 1989] Brown, Frank M. 1989. "The Modal Quantificational Logic Z Applied to the Frame Problem", advanced paper *First International Workshop on Human & Machine Cognition*, May 1989 Pensacola, Florida. Abreviated version published in *International Journal of Expert Systems Research and Applications, Special Issue: The Frame Problem. Part A.* eds. Kenneth Ford and Patrick Hayes, vol. 3 number 3, pp169-206 JAI Press 1990. Reprinted in *Reasoning Agents in a Dynamic World: The Frame problem*, editors: Kenneth M. Ford, Patrick J. Hayes, JAI Press 1991.

[Brown 2003a] Frank M. Brown "Representing Reflective Logic in Modal Logic", *Information Theories and Applications*, Vol. 10, no. 4, pages: 431-438, December 2003.

[Brown 2003b] Frank M. Brown, "Representing Default Logic in Modal Logic", *Information Theories and Applications*, Vol. 10, no. 4, pages 439-446, December, 2003.

[Brown 2003c]Frank M. Brown, "Representing Autoepistemic Logic in Modal Logic", *Information Theories and Applications*, Vol. 10, no. 4, pages 455-462, December 2003.

[Brown 2004] Brown, Frank M. 2004, "Representing "Recursive" Default Logic in Modal Logic", *Information Theories and Applications*, Vol. 11, no. 4, pages: 431-438, 2004.

[Brown 2005] Brown, Frank M. 2004, "Representing the Closed World Assumption in Modal Logic", University of Kansas Artificial Intelligence Lab Report 2005-1. Submitted to this Conference.

[Carnap 1946] Carnap, Rudolf 1946. "Modalities and Quantification" *Journal of Symbolic Logic*, vol. 11, number 2.

[Carnap 1956] Carnap, Rudolf 1956. *Meaning and Necessity: A Study in the Semantics of Modal Logic*, The University of Chicago Press.

[Fine 1970] Fine, K. 1970. "Propositional Quantifiers in Modal Logic" *Theoria* 36, p336--346.

[Hughes & Cresswell 1968] Hughes, G. E. and Creswell, M. J., 1968, *An Introduction to Modal Logic*, Methuen & Co. Ltd., London.

[Mendelson 1964] Mendelson, E. 1964. *Introduction to Mathematical Logic*, Van Norstrand, Reinhold Co., New York.

[Moore 1985] Moore, R. C. 1985. "Semantical Considerations on Nonmonotonic Logic" *Artificial Intelligence*, 25.

[Parks 1976] Parks, Z. 1976. "An Investigation into Quantified Modal Logic", Studia Logica 35, p109-125.

[Quine 1969] Quine, W.V.O., Set Theory and Its Logic, revised edition, Oxford University Press, London, 1969.

[Reiter 1978] Reiter, R. 1978. "On Closed World Databases", in, *Logic and Databases*, eds. Gallaire and Minker, J., Plenum Press, Ney York.

[Reiter 1980] Reiter, R. 1980. "A Logic for Default Reasoning", *Artificial Intelligence*, 13.

Author's Information

Frank M. Brown – University of Kansas, Lawrence, Kansas, 66045, e-mail: brown@ku.edu.

AUTOMATIC FIXED-POINT DEDUCTION SYSTEMS FOR FIVE DIFFERENT PROPOSITIONAL NONMONOTONIC LOGICS

Frank M. Brown

***Abstract**: The commonality and differences among five different nonmonotonic logics is described by implementing an automatic fixed-point equation solver for their propositional logic versions with finite stream based algorithms involving maps, filters, and accumulators. The result of organizing nonmonotonic computations in this fashion is to make apparent in an elementary way, that different nonmonotonic systems embody many of the same basic ideas and in fact differ by often only a few filters or accumulators. The nonmonotonic systems investigated are the Closed World Assumption, the kernel of Autoepistemic Logic, Frame Logic, Default Logic, and Parallel Circumscription. Scheme code which defines all the fixed-points for all these systems for all propositional problems is given.*

***Keywords**: Automatic Deduction Systems, Fixed-point, Nonmonotonic Logic.*

1. Introduction

Growing out of the need for logics and automatic deduction systems for common sense reasoning phenomena such as default reasoning, reasoning about an agent's knowledge or lack thereof, and reasoning about the consequences of robotic actions, researchers have developed a number of logical systems generally called nonmonotonic logics. The various systems at first appeared to be very different partly because of the different formalisms involved and partly because the interrelationships were not at first understood or were misunderstood. However, there are now a number of known important relationships between the different nonmonotonic systems. These results are often metatheorems which state that certain sets of nonlogical axioms and inference rules in one nonmonotonic system have fixed-point solutions that are related to the fixed-point solutions produced by certain other sets of nonlogical axioms and inference rules in some other nonmonotonic system. Herein we take a different approach to comparing nonmonotonic systems. Our approach is to begin by studying the propositional structure of nonmonotonic systems while ignoring the quantificational structure. Later, we hope to extend this research by restoring widening ranges of quantificational structure to the propositional nonmonotonic systems studied herein. Because herein we limit ourselves to propositional nonmonotonic systems everything is finite and we will be able to present the different nonmonotonic systems as Lambda Calculus functions. These functions, define the propositional versions of these nonmonotonic systems by specifying all the fixed-points for any input sets of nonlogical axioms and inference rules. By writing these functions in the fashion of (finite) stream functions involving generators, maps, filters, and accumulators, we are then able to extract out the differences among different nonmonotonic systems as some minor changes in a few of these Lambda Calculus functions thereby providing an interesting elementary way to compare them. Alternatively a programmer (as opposed to a theoretical logician) may view these functions as being Scheme programs.

Section 2 presents the call to a Propositional Logic Decision Procedure. Section 3 presents the generic functions for testing whether defaults hold. These functions implement the basic ideas of testing defaults and generating potential fixed-points which are common to all the nonmonotonic systems discussed herein. Since we are limiting this study to propositional logic the tableaux algorithm in Section 2 is decidable returning true or false. For this reason we don't need to represent the nonmonotonic default structure itself in a formal logic such as in the modal logic style of [Brown 1986] and [Brown 1989], nor in the tableaux style of [Bonatti et.al 2002] and [Olivetti 1992]. These results in a significant simplification of the automatic deduction systems involved, allowing each to be described in as a few small functions thereby making these systems available to the many AI scientists who understand LISP like languages but are not familiar with modal logic nor settheoretic fixed-point description of nonmonotonic logics. The remaining sections give Propositional Logic versions of the Closed World Assumption (Section 4), the kernel of Autoepistemic Logic (Section 5), Frame Logic (Section 6), Default Logic (Section 7), and Parallel Circumscription (Section 8). Some conclusions are made in Section 9.

2. An Automatic Theorem Prover for Propositional Logic

We assume the existence of a decision procedure for Propositional Logic represented as a Scheme function called `prove`. This function is called with two arguments as `(prove h x)` where `h` is a list of hypotheses and `c` is a theorem to be proven. Each hypothesis and the theorem are sentences of the propositional logic obtained by replacing the variables p, q, p1,...,pn in one of the forms: #t, #f, (and p1,...,pn), (or p1,...,pn), (not p), (if p q), (iff p q) by other sentences. The elementary sentences are distinct symbols such as P or list structures not beginning with a logical symbol such as (Loves John Mary).

3. Nonmonotonic Default Inference Rules

Nonmonotonic inferences in Propositional Logic may be specified by nonlogical inference rules called "defaults" of the form:

$$\frac{\alpha_1,...,\alpha_m : \beta_1,...,\beta_n}{\chi}$$

where $\alpha_1,...,\alpha_m$, $\beta_1,...,\beta_n$, and χ are sentences. Such a nonlogical inference rule is interpreted to mean that if each α_j holds in a given theory A_j and if each β_i is consistent with respect to a given theory B_i then χ may be inferred. If we suppose that all the A_j theories are identical, then the inference rule may be rewritten as:

$$\frac{\alpha : \beta_1,...,\beta_n}{\chi}$$

where α is (and $\alpha_1...\alpha_m$) since α holds in A_i if and only if each α_i holds in that theory. The case where there are no α_j sentences may then be represented by letting α be #t. However, if we suppose that the B_i theories are identical, the $\beta_1,...,\beta_n$ sentences cannot likewise be replaced by one large β since each β_i sentence may be consistent with a theory without (and $\beta_1,...,\beta_n$) itself being consistent with the theory, as for example, in the theory `(not(and p q))` where β_1 is `p` and β_2 is `q`. Thus, a nonlogical inference rule or default will be represented as a three element list of the form: $(\alpha(\beta_1...\beta_n)\chi)$

Assuming that all A_j are identical and that all the B_i are identical, a Scheme function to determine whether a nonlogical inference rule is applicable is given in Figure 2. `test` takes as arguments: A_j, B_i, and a default.

```
(define(test a b d)
       (define(pv x) (prove a x))
       (define(pos x) (not(prove b(list 'not x))))
       (and(pv(car d)) (all pos(cadr d))))
```

Fig. 1. Procedure to determine whether a default is applicable.

`test` assumes that the theories `a` and `b` are known before one tests the default `d`. However, if either of the theories on which the α and $\beta_1,...,\beta_n$ sentences of each default are tested itself includes the χ sentences of those defaults which fired, then we need to already know which χ sentences fired in order to test the α and $\beta_1,...,\beta_n$ sentences. This circularity may be avoided by picking a subset of the set of defaults to be the fired subset and then testing that each default in the chosen set is applicable and that each default not in the chosen subset is not applicable. By successively choosing as the set of fired defaults each subset of the initial set of defaults we get all potential default subsets for generating the fixed-points.

A Scheme function called `testall?` which determines whether a subset `s` of the set of defaults `Ds` is a fired subset of defaults is given in Figure 3. This function calls `test+` which checks to see that every default which should fire (i.e. those in `s`) does and `test-` which checks to see that every default which is not supposed to fire (i.e. those not in `s`), does not. It also calls the function `setdifference` which computes the defaults which are not fired. Also given in Figure 3 are the `fp` and `subsets` functions. The `fp` function produces a potential fixed-point which is a list of all the χ sentences of the fired defaults appended to the set of axioms `As`. The `subsets` function generates the list of all subsets of the set of defaults.

```
(define(testall? a b s Ds)
       (and(test+ a b s)(test- a b(set-difference Ds s))))
(define(test+ a b s)(all(lambda(d)(test a b d))s))
(define(test- a b s)(all(lambda(d)(not(test a b d)))s))
(define(set-difference Ds s) (filter(lambda(x)(not(memq x s)))Ds))
(define(fp As s)(append(map caddr s)As))
(define(subsets L)(accumulate
       (lambda(x m)(append(map(lambda(s)(cons x s))m)m))'(())L))
```

Fig. 2. Procedures to test whether a subset of defaults is a firing set of defaults.

4. The Closed World Assumption

The Closed World Assumption [Reiter 1978] (which is related to Completion systems [Clark 1978]) is a rule which allows the inference of a sentence of the form $\neg(\pi\ \delta_1...\delta_m)$ where π in an m-ary predicate and $\delta_1...\delta_m$ are variable free terms, whenever that sentence is consistent with a given theory:

$$\frac{: \neg(\pi\ \delta_1...\delta_m)}{\neg(\pi\ \delta_1...\delta_m)}$$

Since no variables occur in $(\pi\ \delta_1...\delta_m)$ we may think of it as being a propositional constant χ.

$$\frac{: \chi}{\chi}$$

We interpret this structure to mean that there is a set of axioms As such that if χ is consistent with As then χ. We generalize the closed world assumption so that there may be an α sentence, any number of β_i sentences which may be different from χ:

$$\frac{\alpha: \beta_1,...,\beta_n}{\chi}$$

and interpret this structure to mean that there is a set of axioms As such that if α follows from As and each β_i is consistent with As then χ. A Scheme function to compute the result of applying such defaults to an initial set of axioms is given in Figure 4. This function works by generating all the subsets of the set of defaults, filtering them by removing those subsets which are not fired subsets, and then by constructing the resulting fixed-points.

```
(define(cwa-fixedpoints As Ds)
       (map(lambda(s)(fp As s))
           (filter(lambda(s)(testall? As As s Ds))
                  (subsets Ds))))
```

Fig. 3. Fixed-point Deducer for The Closed World Assumption

Since the χ sentences of the defaults are not part of As which is being used to test the defaults, there is only one set of firing defaults. The resulting theory is the result of appending the χ sentences of the firing defaults to As. Thus, there is a single cwa fixed-point which may be computed by the more efficient function given in Figure 5 which tests each default to see if it fires and then constructs the fixed-point from As and all the firing defaults.

```
(define(cwa-fixedpoint As Ds)
       (fp As(filter(lambda(d)(test As As d))Ds)))
```

Fig. 4. Efficient Fixed-point Deducer for the Closed World Assumption

Example 1: This example has one axiom: A and one default:

$$\frac{:B}{B}$$

The set of axioms is represented as a list of axioms `'(A)` and the set of defaults is represented as a list of defaults `'((#t(B)B))`. Since there is no α part of this default, it is represented as `#t`. To compute the fixed-point we simply apply `cwa-fixedpoint` to the set of axioms and the set of defaults:

```
(cwa-fixedpoint '(A) '((#t(B)B))  => (A B)
```

Example 2: Here is an example illustrating the problem `cwa` has in dealing with two contradictory defaults:

$$\frac{:A}{A} \qquad \frac{:\neg A}{\neg A}$$

```
(cwa-fixedpoint '()'((#t(A)A)(#t((not A))(not A))=>(A(not A))
```

which is a fixed-point from which `#f` may be derived by the prove function.

5. The Kernel of Autoepistemic Logic

Autoepistemic Logic [Moore 1985] for a Propositional Logic includes the syntax of Propositional Logic supplemented with a unary operation L which has the properties of an S4.5 modal logic. These modal properties allow each sentence of this Propositional Autoepistemic Logic to be rewritten [Konolige 1987] as a finite set of sentences of the form: $((L\alpha)\wedge(\neg L\neg\beta_1)\wedge...\wedge(\neg L\neg\beta_n))\rightarrow\chi$ where the $(L\alpha)$ expression is optional in any sentence and n may be 0. Such a sentence can then be thought of as being the inference rule:

$$\frac{\alpha: \beta_1,...,\beta_n}{\chi}$$

where α follows from the given theory k and each β_i is consistent with that theory k. The given theory k will always be a fixed-point. The fixed-points can be obtained by enumerating all the subsets of the set of defaults, filtering out those subsets which are not fired subsets and conjoining all the χ sentences of the firing defaults to the initial axioms. A Scheme function to do this, called `aek-fixedpoints`, is given in Figure 6. [Bouix 1998] gives an analogous function written in Logistica [Brown 2003b]. This procedure is an encoding of the splitting technique on the modal logic representation of nonmonotonic reasoning described in [Brown 1986].

```
(define(aek-fixedpoints As Ds)
       (map(lambda(s)(fp As s))
           (filter(lambda(s)(testall?(fp As s)(fp As s)s Ds))
                  (subsets Ds))))
```

Fig. 6. Fixed-point Deducer for the Kernel of Autoepistemic Logic

Since the χ sentences of the defaults are part of the set being used to test the defaults there is not necessarily a single fixed-point as in the case of cwa. For example, the empty set of axioms and the default A:/¬A has no fixed-points. Likewise the empty set of axioms and the two defaults: :A/A and :¬A/¬A has two fixed-points, namely one consisting of A and the other consisting of ¬A. In general, if there are n defaults then there will be 2^n potential fixed-points produced by `subsets`. Since each call to `testall` involves a finite number of calls to the propositional logic automatic theorem prover which itself is essentially an $O(2^n)$ process, the resulting complexity is essentially: $O(2^{2n})$. This contrasts favourably with [Antoniou 1997] which describes a procedure that generates 2^m subsets where m is the total number of α and β sentences in all the defaults. Thus [Antoniou 1997] is an $O(2^{n+m})$ process where m is greater than or equal to n and is usually much larger. ([Antoniou 1997] does not however claim to be presenting an efficient algorithm). Another approach is presented in [Eiter, Klotz, Tompits, & Woltran 2002] which is based on a quadratic encoding of an Autoepistemic Theory into a propositional logic with propositional quantifiers. [Brown 2003a] compares that approach with the approach used herein. Since quantified propositional logic is at least as difficult as propositional logic, the quadratic representation leads to a proof procedure which is essentially: $O(2^{n*n})$. Both the systems of [Antoniou 1997] and [Eiter, Klotz, Tompits, &

Woltran 2002] produce additional sentences of the form: L'χ_i whenever χ_i is a theorem of the fixed-point and $\neg$L'χ_i whenever χ_i is not a theorem of (all) the fixed-points. This may be obtained in our system by adding the schema: L'$\chi_i \leftrightarrow [k]\chi_i$ to each kernel fixed-point k. However since L does not occur in the kernel fixed-point this addition constitutes a conservative extension of the kernel fixed-point and is therefore irrelevant. The null set of axioms with the n defaults: $\{A_i:/A_i\}_{i=1,n}$ has 2^n fixed-points – one corresponding to each subset of the set of defaults. Since, subsets in Figure 6 produces 2^n subsets, and each subset needs to be checked with `prove`, it is difficult to imagine a more asymptotically efficient algorithm that produces all the fixed-points than the one given hereabove, although producing fixed points by splitting on the modal subformulas and symbolically simplifying at each step, as was used in the nonmonotonic automatic deduction systems described in: [Brown & Araya 1991] and [Leasure 1993] would often allow solutions to be determined without enumerating all the fixed-points.

Aek is a powerful logic capable of representing, as defaults rules, action logics involving both precondition/result pairs of actions and the necessary frame laws:

$$\frac{\text{preconditions-action(t):}}{\text{results-action (t+1)}} \qquad \frac{\alpha(t):\ \chi(t+1)}{\chi(t+1)}$$

t is a number representing the time (represented as an additional argument to each predicate) when the action is applied. The first default states that the results of an action applied at time t holds at time t+1 if the preconditions held at the previous moment of time t. The second law is the frame law which states that a property α which holds at time t causes a property χ to hold at time t+1 if it is consistent for χ to do so. In the simpler cases (usually discussed in the literature) α is identical to χ.

6. Frame Logic

Frame Logic [Brown 1987] represents in a direct manner action logics involving both the precondition/result pairs of actions and the necessary frame laws. A Propositional Frame Logic involves a set of axioms and default laws representing precondition/action pairs and the necessary frame laws of the following forms:

$$\frac{\text{preconditions-action:}}{\text{results-action}} \qquad \frac{\alpha:\ \chi}{\chi}$$

Unlike Autoepistemic Logic (aek) no numeric subscripts representing time are needed because the sentences before the colon in any default rule follow from a given theory representing what holds at the preceding moment in time. Let OLD be the theory which specifies what holds in the previous moment of time. An action law then says that if the preconditions held in OLD then the results hold in the fixed-point (which represents what now holds). Likewise, the frame laws say that a property α which holds in OLD causes a property χ to hold in the fixed-point if χ is consistent for it to do so. Again in the simpler cases (usually discussed in the literature) α is identical to χ. [Brown 1989] discusses more sophisticated cases dealing with Newtonian Mechanics.

We generalize Frame Logic defaults to:

$$\frac{\alpha:\ \beta_1,\ldots,\beta_n}{\chi}$$

where α holds in the old theory and each βi is consistent with a resulting fixed-point which includes the χ sentences of the firing defaults. The fixed-points can be obtained by enumerating all the subsets of the set of defaults, filtering out those subsets which are not fired subsets and conjoining all the χ sentences of the firing defaults to the initial axioms. A Scheme function to do this, called `frame-fixedpoints`, is in Figure 7.

```
(define(frame-fixedpoints old As Ds)
       (map(lambda(s)(fp As s))
           (filter(lambda(s)(testall? old(fp As s)s Ds))
                  (subsets Ds))))
```

Fig. 7. Fixed-point Deducer for Frame Logic

7. Default Logic

Default Logic [Reiter 1980] for a Propositional Logic is essentially a set of axioms As and a finite set of default rules of the form:

$$\frac{\alpha : \beta_1,\ldots,\beta_n}{\chi}$$

where α follows from a theory k and each β_i is consistent with that theory k. The theory k must be constructible by adding to the axioms the χ sentences of those defaults whose α sentences are already deducible from the axioms and previously deduced χ sentences. This requirement does not allow the α sentence of a default to be proven by using its own χ sentence. This supported nature of Default Logic, perhaps, more closely represents defaults such as those used in taxonomies and other cases than do logics such as Autoepistemic logic. Other authors prefer other Default Logics such as Justified Default Logic or Constrained Default Logic. [Antoniou 1997] discusses the merits of different alternatives.

The fixed-points of a system of defaults of Default Logic is obtained by enumerating all the subsets of the set of defaults, filtering out those subsets which are not fired subsets, filtering out those subsets which are not supported, and then conjoining all the χ sentences of each set of firing defaults to the initial axioms. A Scheme function to derive all the fixed-points, called `dl-fixedpoints`, is in Figure 8.

```
(define(dl-fixedpoints As Ds)
       (map(lambda(s)(fp As s))
           (filter(lambda(s)(supported? As s))
                  (filter(lambda(s)(testall?(fp As s)(fp As s)s Ds))
                         (subsets Ds)))))
(define(supported? rn s)
      (define fs(filter(lambda(d)(prove rn(car d)))s))
      (if(null? fs)(null? s)
         (supported?(append(map caddr fs)rn)(set-difference s fs))))
```

Fig. 8. Fixed-point Deducer for Default Logic

Since Default logic as defined herein differs from Autoepistemic kernels by a single filter (i.e. `supported?`) it is obvious that the fixed-points of Default Logic are a subset of the fixed-points of Autoepistemic Kernels as Konolige suggested and eventually proved. (see [Konolige 1987]). This algorithm has many variations. For example, `testall?` can be replaced by:

```
(and(test+ '(#f)(fp As s)s Ds))
    (test-(fp As s)(fp As s)(set-difference Ds s)))
```

making the entailment check in test+ trivially true. This check is not needed since it is implied by the `supported?` filter. The Default Logic algorithms given in both [Schwind 1990] and [Antoniou 1997] use this fact. Another variation is that the order of filters may be swapped. Whereas [Schwind 1990] apply the above filter first followed by the `supported?` filter, [Antoniou 1997] does the reverse. [Antoniou 1997] gives an algorithm (where no default may have more than one β expression) taking this approach but which also combines the `supported?` test into the subset generation procedure. Instead of using the subset generator in Figure 8, which produces 2^n candidates, [Antoniou 1997] generates n! permutations (but suggests that these redundancies may be eliminated). His algorithm has the advantage of not producing candidate fixed-points for defaults whose α expressions do not hold in a fixed-point, and likewise for β expressions which are not consistent with any fixed-point with the expense of redundant tests on the βs.

8. Circumscription

A nonmonotonic system such as the Closed World Assumption produces precisely one fixed-point. In such a case one may determine that a conjecture follows from that fixed point by simply applying the prove function to the fixed-point and the conjecture. However, most nonmonotonic systems do not always produce precisely one fixed-point. In such a case the question arises as to what fixed-point should be used to derive further

consequences. We could choose arbitrarily but a more conservative approach is to require that the theorem hold in all fixed-points. Since (fp_i=>t) & ... & (fp_n=>t) is equivalent to (fp_1 or ... or fp_n) => t we need to prove t from the disjunction of all the fixed-points. From this perspective the problem is to compute this disjunction. It turns out that there is a nonmonotonic system, namely Circumscription [McCarthy 1980, McCarthy 1986, and Lifschitz 1985] that produces just this disjunction [Konolige 1989, Brown 1989] when all the defaults are of the form:

$$\frac{:\chi}{\chi}$$

Specifically, Circumscription for a Propositional Logic may be thought of as being a rule which infers sentences where the form $\neg(\pi\ \delta_1...\delta_m)$ where π in an m-ary predicate and $\delta_1...\delta_m$ are variable free terms, whenever that sentence is consistent with a given theory:

$$\frac{:\neg(\pi\ \delta_1...\delta_m)}{\neg(\pi\ \delta_1...\delta_m)}$$

Such a predicate is said to be Circumscribed. Since no variables occur in $(\pi\ \delta_1...\delta_m)$ we think of it as being a propositional constant. In addition to Circumscribed Predicates, there may be Variable Predicates and Fixed Predicates. Variable Predicates involve no rules but Fixed Predicates involve two contradictory rules of the form:

$$\frac{:\neg(\pi\ \delta_1...\delta_m)}{\neg(\pi\ \delta_1...\delta_m)} \qquad \frac{:(\pi\ \delta_1...\delta_m)}{(\pi\ \delta_1...\delta_m)}$$

These default structures may be interpreted as an Autoepistemic Kernel structure, a Frame Logic structure, or as a Default Logic structure since all three interpretations are the same for the defaults used in Circumscription.

We generalize the defaults of circumscription so that there may be an α sentence, any number of β_i sentences, and so that last occurrences of χ may be any sentence:

$$\frac{\alpha: \beta_1,...,\beta_n}{\chi}$$

A Scheme function for Circumscription by interpreting its defaults as Autoepistemic Kernel defaults is given in Figure 9. This function accumulates the fixed-points together by simply returning the disjunction of the conjunction of all the sentences in each Autoepistemic Kernel fixed-point (see [Brown 1989, Konolige 1989]).

```
(define(circumscription As Ds)
      (define(or-fps s) (cons 'or(map(lambda(k)(cons 'and k))s)))
      (or-fps(aek-fixedpoints As Ds)))
```

Fig. 12. Theory Constructor for Parallel Circumscription with circumscribed, fixed, and variable predicates.

9. Conclusion

The nonmonotonic systems discussed herein are summarized in Table 2 in terms of the filters and accumulators that are used. The main choice is whether testall or `test+` is used and in either case what theories are used to test the α sentences and β sentences of the defaults. There are also the issues of whether the `supported?` filter is used and whether the `or-fps` accumulator is used. Table 2 suggests the existence other nonmonotonic systems with different combinations of filters and accumulators.

Table 1. The differences among the different Nonmonotonic Systems. As is the initial set of axioms, k is the fixed-point, and kβ is the conjunction of the axioms in k and each of the β sentences of all firing defaults.

nonmonotonic system	filters	accumulator
Closed World Assumption	`λs(testall? As As s)`	
Autoepistemic Kernel	`λs(testall? k k s)`	
Frame Logic	`λs(testall? old k s)`	
Default Logic	`λs(testall? k k s),` `λs(supported? As s)`	
Circumscription	`λs(testall? k k s)`	`or-fps`

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Numbers: MIP-9526532, EIA-9818341, and EIA-9972843.

Bibliography

[Antoniou 1997] Antoniou, Grigoris 1997. *NonMonotonic Reasoning*, MIT Press.

[Bonatti, et.al 2002], Nicola, Bonatti, Piero Andrea & Olivetti, Nicola, "Sequent Calculi for Propositional Nonmonotonic Logics", ACM Transactions on Computational Logic, Vol. 3, No. 2, pages 226-278, April 2002.

[Bouix 1998] Bouix, Sylvain, "Solving Autoepistemic Equivalences for Propositional logic", May 7, 1998. personal notes.

[Brown 1986] Brown, F. M., "A Commonsense Theory of Nonmonotonic Reasoning", *Proceedings of the 8th International Conference on Automated Deduction, CADE8,* Lecture Notes in Computer Science, vol. 230, Oxford, England, July 1986, Springer Verlag, New York.

[Brown 1987] Brown, Frank M. 1987. "The Modal Logic Z", In *The Frame Problem in AI"; Proc. of the 1987 AAAI Workshop*, Morgan Kaufmann, Los Altos, CA.

[Brown 1989] Brown, Frank M. 1989. "The Modal Quantificational Logic Z Applied to the Frame Problem", advanced paper *First International Workshop on Human & Machine Cognition*, May 1989 Pensacola, Florida. Abbreviated version published in *International Journal of Expert Systems Research and Applications, Special Issue: The Frame Problem. Part A.* eds. Kenneth Ford and Patrick Hayes, vol. 3 number 3, pp169-206 JAI Press 1990. Reprinted in *Reasoning Agents in a Dynamic World: The Frame problem*, editors: Kenneth M. Ford, Patrick J. Hayes, JAI Press 1991.

[Brown & Araya 1991] Brown, Frank M.& Carlos Araya, "A Deductive System for Theories of Nonmonotonic Reasoning," *Transactions of the Eighth Army Conference on Applied Mathematics and Computing, 1991*.

[Brown 2003a] Frank M. Brown "Decision Procedures for the Propositional Cases of 2nd Order Logic and Z Modal Logic Representations of a 1st Order L-Predicate Nonmonotonic Logic", *Automated Reasoning with Analytic Tableaux and Related Method: Tableaux 2003*, September 2003, Rome, *Lecture Notes in Artificial Intelligence 2796*, ISBN 3-540-40787-1, Springer-Verlag, Berlin, 2003.

[Brown 2003b] Frank M. Brown "Logistica 2.0: A Technology for Implementing Automatic Deduction Systems", *Automated Reasoning with Analytic Tableaux and Related Method: Tableaux 2003*, September 2003, Rome, *Lecture Notes in Artificial Intelligence 2796*, ISBN 3-540-40787-1, Springer-Verlag, Berlin, 2003.

[Clark 1978] Clark, Keith 1978. "Negation as Failure", in, *Logic and Databases*, eds. Gallaire and Minker, J., Plenum Press, New York, pages 293-322.

[Eiter, Klotz, Tompits, & Woltran 2002] Eiter, Thomas & Klotz, Volker & Tompits, Hans and Woltran, Stefan, 2002, "Modal Nonmonotonic Logics Revisited: Efficient Encodings for the Basic Reasoning Tasks", *Automated Reasoning with Analytic Tableaux and Related Method: Tableaux 2002, LNAI 2381, Springer Verlag*.

[Konolige 1987] Konolige, Kurt 1987, "On the Relation between Default Theories and Autoepistemic Logic", IJCAI87.

[Konolige 1989] Konolige, Kurt 1989. "On the Relation between Autoepistemic Logic and Circumscription Preliminary Report", IJCAI89.

[Leasure 1993] Leasure David, "A Logistica Deduction System for Solving NonMonotonic Reasoning Problems Using the Modal Logic Z" Ph.D dissertation, University of Kansas, 1993.

[Lifschitz, 1985] Lifschitz, V. 1985. "Computing Circumscription" *IJCAI 9*,pages 121-127.

[McCarthy 1980] McCarthy, J. "Circumscription -- A Form of Nonmonotonic Reasoning", *Artificial Intelligence*, vol. 13. 1980.

[McCarthy 1986] McCarthy, J. 1986. "Applications of Circumscription to Formalizing Common-Sense Reasoning", *Artificial Intelligence*, vol. 28.

[Moore 1985] Moore, R. C. 1985. "Semantical Considerations on Nonmonotonic Logic" *Artificial Intelligence*, 25.

[Olivetti 1992] Olivetti, Nicola 1992, "Tableaux and Sequent Calculus for Minimal Entailment", Journal of Automated Reasoning 9: 99-139.

[Reiter 1978] Reiter, R. 1978. "On Closed World Databases", in, *Logic and Databases*, eds. Gallaire and Minker, J., Plenum Press, New York.

[Reiter 1980] Reiter, R. 1980. "A Logic for Default Reasoning" *Artificial Intelligence*, 13.

[Schwind 1990] Schwind, Camilla 1990. "A Tableaux-Based Theorem Prover for a Decidable Subset of Default Logic", 10th International Conference on Automated Deduction, Kaiserslautern. Springer Verlag. Lecture Notes in AI vol. 449.

Author's Information

Frank M. Brown – University of Kansas, Lawrence, Kansas, 66045, e-mail: brown@ku.edu.

NONMONOTONIC SYSTEMS BASED ON SMALLEST AND MINIMAL WORLDS REPRESENTED IN WORLD LOGIC, MODAL LOGIC, AND SECOND ORDER LOGIC

Frank M. Brown

***Abstract**: A monotonic representation of a nonmonotonic logic makes it possible for an automatic theorem prover for that monotonic logic to be used to automatically deduce consequences for the nonmonotonic logic. Multiple monotonic representations, allow different automatic deduction approaches to be developed. Herein, we discuss two different nonmonotonic concepts, namely simplest worlds and minimal worlds, and show how each may be represented in three different monotonic logics. These monotonic logics are World Logic, Modal Logic, and Second Order Logic. These representations are more general than those previously described and are related back to less general previous work. In all these representations quantifiers obey the normal laws of both classical logic and S5 Modal Logic and may quantify variables across the scope of the nonmonotonic structures.*

***Keywords**: Smallest Worlds, Minimal Worlds, Nonmonotonic Logic.*

1. Introduction

From the standpoint of the Z Priorian Modal Second Order Logic, we might ask when a sentence is entailed by worlds with certain properties. Two such types of worlds of interest to nonmonotonic reasoning are the nonmonotonic concepts of the Smallest World and the Minimal Worlds. Using the laws of the Z Priorian Modal Second Order Logic, whose notation and axiomatization is given in [Brown 2005], we prove that both Simplest Worlds and Minimal Worlds can be represented in two other ways, namely in a Z MODAL Logic and in a Second Order Logic. This is important because the three equivalent representations of these two nonmonotonic concepts, allow different automatic theorem proving methods to be developed. In the succeeding sections we define Simplest and Minimal Worlds and show that they are representable in all three sub-languages. Smallest Worlds are discussed in Section 2 and Minimal Worlds are discussed in Section 3. Finally, in Section 4 we draw some conclusions.

2. Smallest Worlds

Smallestworlds is the "infinite disjunction" of all the Smallest worlds. A world I is Smallest iff every other world entailing Γ entails more instances of α. These definitions are given below:

(Smallestworlds Γ α_i) =d $\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Smallest I Γ α_i))

(Smallest I Γ α_i) =df $\forall$J(((world J)$\wedge$([J]Γ))$\rightarrow$(I$\leq_\alpha$ J))

(I $\leq_\alpha$ J) =df $\wedge_{i=1,n}\forall\xi_i$(([I]$\alpha_i$)$\rightarrow$([J]$\alpha_i$))

The above definitions may be taken as the Z World Logic representation. We now prove the existence of two other equivalent representations of Smallestworlds. We prove that Smallestworlds are equivalent to the Quantified Closed World Assumption which is represented in Z Modal Logic and that it is also equivalent to Completion, which is representable in SOL. The Quantified Closed World Assumption (i.e. QCWA) is written in Z Modal SOL as:[1]

(QCWA Γ α_i) =df $\exists$k(k$\wedge$(k$\equiv\Gamma$)$\wedge\wedge_{i=1,n}\forall\xi_i$((<k>$\neg\alpha_i$)$\rightarrow\neg\alpha_i$))

where ξ_i is the sequence of free variables in α_i. Intuitively, QCWA asserts the negation of each α_i which is not entailed by Γ. A common subcase of QCWA is where each α_i has the form (π_i ξ_i).

[1] QCWA is equivalent to: $\Gamma\wedge\wedge_{i=1,n}\forall\xi_i$((<$\Gamma$>$\neg\alpha_i$)$\rightarrow\neg\alpha_i$). However, this is not necessarily a linear representation because Γ occurs n times. If there are no free variables in any α_i this sentence may be expressed in FOL metatheory as:
(cwa 'Γ 'α_i)=df ('$\Gamma\cup${'$\neg\alpha_i$: ('$\alpha_i\notin$(fol-theorems 'Γ)})) where fol-theorems produces the set of First Order Logic consequences. Thus, if the α_i constitute the set of all sentences beginning with a predicate and followed by a sequence of variable free terms we get the Closed World Assumption (i.e. cwa) as described in [Reiter 1978].

Example: (QCWA ((N 0)$\wedge\forall$x((N x)$\rightarrow$(N(+1 x)))) (N x))
is equivalent to: $\forall$x((N x)$\leftrightarrow$([Γ 0)$\wedge\forall$x((π x)$\rightarrow$(π(+1 x)))](π x)))

We first reformulate Smallest as an entailment:
QCWA1:(Smallest I Γ α_i)$\leftrightarrow$([Γ]$\wedge_{i=1,n}\forall\xi_i$(([I]$\alpha_i$)$\rightarrow\alpha_i$))
proof: (Smallest I Γ α_i). Unfolding the definitions gives: $\forall$J(((world J)$\wedge$([J]Γ))$\rightarrow$(I $\leq_\alpha$ J))
and then: $\forall$J(((world J)$\wedge$([J]Γ))$\rightarrow\wedge_{i=1,n}\forall\xi_i$(([I]$\alpha_i$)$\rightarrow$([J]$\alpha_i$)))
Since J is a world, we pull out the J entailments giving: $\forall$J((world J)$\rightarrow$[J]($\Gamma\rightarrow\wedge_{i=1,n}\forall\xi_i$(([I]$\alpha_i$)$\rightarrow\alpha_i$)))
By N1 this is equivalent to: []($\Gamma\rightarrow\wedge_{i=1,n}\forall\xi_i$(([I]$\alpha_i$)$\rightarrow\alpha_i$)) which is equivalent to: [Γ]$\wedge_{i=1,n}\forall\xi_i$(([I]$\alpha_i$)$\rightarrow\alpha_i$) QED.

We now show that the (Smallest Γ α_i) is logically equivalent to the Quantified Closed World Assumption.
QCWA2: (Smallestworlds Γ α_i) $\equiv$ (QCWA Γ α_i)
proof: (Smallestworlds Γ α_i). Unfolding the definitions gives: $\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Smallest I Γ α_i))
Using theorem QCWA1 gives: $\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$[Γ]$\wedge_{i=1,n}\forall\xi_i$(([I]$\alpha_i$)$\rightarrow\alpha_i$))
which is equivalent to: $\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge\ \wedge_{i=1,n}\forall\xi_i$(([I]$\alpha_i$)$\rightarrow$([$\Gamma$]$\alpha_i$)))
Since J is a world [I] may be pulled out and joined giving: $\exists$I(I$\wedge$(world I)$\wedge$([I]($\Gamma\wedge\ \wedge_{i=1,n}\forall\xi_i$($\alpha_i\rightarrow$([$\Gamma$]$\alpha_i$))))).
From theorem D1 we get: $\Gamma\wedge\ \wedge_{i=1,n}\forall\xi_i$($\alpha_i\rightarrow$([$\Gamma$]$\alpha_i$)) which may be rewritten as: $\Gamma\wedge\wedge_{i=1,n}\forall\xi_i$((<$\Gamma$>$\neg\alpha_i$)$\rightarrow$($\neg\alpha_i$)).
Pulling out all occurrences of Γ this may be rewritten as: $\exists$k(k$\wedge$(k$\equiv\Gamma$)$\wedge\wedge_{i=1,n}\forall\xi_i$((<k>$\neg\alpha_i$)$\rightarrow\neg\alpha_i$)) which is just (QCWA Γ α_i). QED.

Completion is a representation of Smallest Worlds in Second Order Logic. Completion in SOL is defined as:
(Completion Γ α_i) =df ($\Gamma\wedge\forall P_1\ldots P_m$(($\Gamma${$\pi_j$/$P_j$})$\rightarrow\wedge_{i=1,n}\forall\xi_i$($\alpha_i\rightarrow$($\alpha_i${$\pi_j$/$P_j$}))))
where $\pi_i\ldots\pi_m$ are <u>all</u> the unmodalized predicate symbols in Γ and α_i.
Example: (Completion ((N 0)$\wedge\forall$x((N x)$\rightarrow$(N(+1 x)))) (N x)) is equivalent to:
$\forall$x((N x)$\leftrightarrow$($\forall$P(((P 0)$\wedge\forall$x((P x)$\rightarrow$(P(+1 x))))$\rightarrow$(P x)))

CL1: (Smallestworlds Γ α_i)$\equiv$(Completion Γ α_i)
proof: (Smallestworlds Γ α_i). Unfolding Smallestworlds gives: $\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Smallest I Γ α_i))
By QCWA1 this is: $\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$([Γ]$\wedge_{i=1,n}\forall\xi_i$(([I]$\alpha_i$)$\rightarrow\alpha_i$)))
Since I is a world this is equivalent to: $\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$[I]([Γ]$\wedge_{i=1,n}\forall\xi_i$(([I]$\alpha_i$)$\rightarrow\alpha_i$)))
By T1 this is equivalent to: $\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$[I]$\forall P_1\ldots P_m$($\Gamma\rightarrow\wedge_{i=1,n}\forall\xi_i$(([I]$\alpha_i$)$\rightarrow\alpha_i$)){$\pi_j$/$P_j$})
where $\pi_i\ldots\pi_m$ are all the unmodalized predicates in Γ or any α_i. Pushing the substitutions in (to the unmodalized subformulas) gives the equivalent sentence:
$\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$[I]$\forall P_1\ldots P_m$((Γ{π_j/P_j})$\rightarrow\wedge_{i=1,n}\forall\xi_i$(([I]$\alpha_i$)$\rightarrow$($\alpha_i${$\pi_j$/$P_j$}))))
Since I is a world, [I] may be pushed in with nestings [I][I] absorbed to [I] and the then pulled out giving:
$\exists$I(I$\wedge$(world I)$\wedge$([I]($\Gamma\wedge\ \forall P_1\ldots P_m$(($\Gamma${$\pi_j$/$P_j$})$\rightarrow\wedge_{i=1,n}\forall\xi_i$($\alpha_i\rightarrow$($\alpha_i${$\pi_j$/$P_j$}))))))
By D1 this is equivalent to: $\Gamma\wedge\forall P_1\ldots P_m$(($\Gamma${$\pi_j$/$P_j$})$\rightarrow\wedge_{i=1,n}\forall\xi_i$($\alpha_i\rightarrow$($\alpha_i${$\pi_j$/$P_j$}))) QED

(SOL) Parallel Predicate Completion is the instance of Completion where each α_i is a sentence beginning with a distinct predicate: (π_i ξ_i). In this case n$\leq$m. It may be written as:
$\Gamma\wedge\ \forall P_1\ldots P_M$(($\Gamma${$\pi_i$/$P_i$})$\rightarrow\wedge_{i=1,n}\forall\xi_i$(($\pi_i$ ξ_i)$\rightarrow$(P_i ξ_i))) where $\pi_i\ldots\pi_m$ are <u>all</u> the unmodalized predicate symbols in Γ. The Completed Predicates are $\pi_1\ldots\pi_n$ and the Varying Predicates are $\pi_{n+1}\ldots\pi_m$. Rewriting the Varying Predicates with different variable and metavariable symbols and indices gives:
$\Gamma\wedge\ \forall P_1\ldots P_n\forall Z_1\ldots Z_m$(($\Gamma${$\pi_i$/$P_i$}{$\rho_j$/$Z_j$})$\rightarrow\wedge_{i=1,n}\forall\xi_i$(($\pi_i$ ξ_i) $\rightarrow$(P_i ξ_i)))
where $\pi_1\ldots\pi_n$ and $\rho_1\ldots\rho_m$ are <u>all</u> of the unmodalized predicate symbols in Γ.[1] 2.

[1]A procedure for completing predicates [Clark 1978] for a set of Horn Clauses Γ involving separable predicates is as follows: Γ is put into the form ($\chi\wedge\wedge_{i=1,n}\forall\xi_i$($\alpha_i\rightarrow$($\pi_i$ ξ_i))) where $\pi_1\ldots\pi_n$ do not occur in χ nor in any α_i. The completion is to replace each $\rightarrow$ by $\leftrightarrow$. In this form the definition of predicate completion in the text becomes:
($\chi\wedge\wedge_{i=1,n}\forall\xi_i$($\alpha_i\rightarrow$($\pi_i$ ξ_i))) $\wedge\ \forall P_1\ldots P_m$(($\chi\wedge\wedge_{i=1,n}\forall\xi_i$($\alpha_i\rightarrow$($P_i$ ξ_i))) $\rightarrow\wedge_{i=1,n}\forall\xi_i$(($\pi_i$ ξ_i) $\rightarrow$(P_i ξ_i)))

3 Minimal Worlds

Minworlds is the "infinite disjunction" of all the minimal worlds which entail Γ. A world I is minimal iff every world which entails Γ and is less than or equal to I for all α_i, is such that I is less than or equal to it for all α_i. A world is less than or equal to another world for α_i iff α_i is entailed by the second world whenever it is entailed by the first. These definitions[1] are given below:

(Minworlds Γ α_i) =df $\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Minimal I Γ α_i))

(Minimal I Γ α_i) =df $\forall$J(((world J)$\wedge$([J]Γ)$\wedge$(J $\leq\alpha$ I))$\rightarrow$(I $\leq\alpha$ J))

(J $\leq\alpha$ I) =df $\wedge_{i=1,n}\forall\xi_i$(([J]$\alpha_i$)$\rightarrow$([I]$\alpha_i$))

Every Smallest World is a Minimal World. Even though Minimal Worlds are not necessarily Smallest worlds, if the "disjunction" of the smallest worlds is possible then there is only one and it will be the Minimal World. These facts are proven below[2]:

SM1: Smallest worlds entail minimal worlds: [(Smallestworlds Γ α_i)](Minworlds Γ α_i)

proof: Unfolding the definitions of Smallestworlds and Minworlds gives:

[($\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Smallest I Γ α_i)))] ($\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Minimal I Γ α_i)))

which by letting I:=I simplifies to just: [(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Smallest I Γ α_i))] (Minimal I Γ α_i)

Generalizing gives: (Smallest k Γ α_i)$\rightarrow$(Minimal k Γ α_i). Unfolding Smallest and Minimal gives:

($\forall$J(((world J)$\wedge$([J]Γ))$\rightarrow$(I $\leq\alpha$ J)))$\rightarrow\forall$J(((world J)$\wedge$([J]Γ)$\wedge$(J $\leq\alpha$ I))$\rightarrow$(I $\leq\alpha$ J))

which holds by classical logic. QED

SM2: A Reduction case of Minworlds to Smallestworlds.

(<>(Smallestworlds Γ α_i))$\rightarrow$(Minworlds Γ α_i)$\equiv$(Smallestworlds Γ α_i)

proof: By SM1 it suffices to prove: (<>(Smallestworlds Γ α_i))$\rightarrow$ [(Minworlds Γ α_i)](Smallestworlds Γ α_i)

Unfolding Smallestworlds and Minworlds gives:

(<>$\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Smallest I Γ α_i)))$\rightarrow$

[$\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Minimal I Γ α_i)) $\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Smallest I Γ α_i))

The hypothesis simplifies as follows: first: <>$\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Smallest I Γ α_i)), then

$\exists$I <> (I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Smallest I Γ α_i)), then $\exists$I ((<>I)$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Smallest I Γ α_i)), and then:

$\exists$W (world W)$\wedge$([W]Γ)$\wedge$(Smallest W Γ α_i)) and the conclusion simplifies as follows:

[$\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Minimal I Γ α_i))]$\exists$I(I$\wedge$(world I)$\wedge$([I]Γ)$\wedge$(Smallest I Γ α_i))

$\forall$U[(U$\wedge$(world U)$\wedge$([U]Γ)$\wedge$(Minimal U Γ α_i))]$\exists$V(V$\wedge$(world V)$\wedge$([V]Γ)$\wedge$(Smallest V Γ α_i)). In Z, this is equivalent to: $\forall$U((world U)$\wedge$([U]Γ)$\wedge$(Minimal U Γ α_i))$\rightarrow\exists$V([U]V]$\wedge$(world V)$\wedge$([V]Γ)$\wedge$(Smallest V Γ α_i))

Since a world entails a world only if they are synonymous we get:

$\forall$U((world U)$\wedge$([U]Γ)$\wedge$(Minimal U Γ α_i))$\rightarrow$ $\exists$V(U$\equiv$V)$\wedge$(world V)$\wedge$([V]Γ)$\wedge$(Smallest V Γ α_i))

Eliminating the $\exists$V gives: $\forall$U ((world U)$\wedge$([U]Γ)$\wedge$(Minimal U Γ α_i))$\rightarrow$((world U)$\wedge$([U]Γ)$\wedge$(Smallest U Γ α_i))

which simplifies to: $\forall$U(((world U)$\wedge$([U]Γ)$\wedge$(Minimal U Γ α_i))$\rightarrow$(Smallest U Γ α_i))

Putting the simplified hypothesis and conclusion together gives:

Letting P_i be $\lambda\xi_i\alpha_i$ allows us to infer: ($\chi\wedge\wedge_{i=1,n}\forall\xi_i(\alpha_i\rightarrow(\pi_i\ \xi_i)))\wedge\wedge_{i=1,n}\forall\xi_i((\pi_i\ \xi_i)\rightarrow\alpha_i)$
which is equivalent to: $\chi\wedge\wedge_{i=1,n}\forall\xi_i((\pi_i\ \xi_i)\leftrightarrow\alpha_i)$.

[1] A long-winded definition of Minimal is: (Minimal I Γ α_i β_j) =df ($\forall$J((world J)$\wedge$([J]Γ)$\wedge$(J $\leq\alpha$ I)$\wedge$(J =β I))$\rightarrow$(I $\leq\alpha$ J)), (J $\leq\alpha$ I) =df ($\wedge_{i=1,n}\forall\xi_i$(([J]$\alpha_i$)$\rightarrow$([I]$\alpha_i$))), (J =$\beta$ I) =df (J $\leq\beta$ I)$\wedge$(I $\leq\beta$ J). Since (I $\leq\beta$ J)) is equivalent to (J $\leq\neg\beta$ I) this definition of minimal is equivalent to the one used in the text provided that the α_i used therein is (α_i,β_j,$\neg\beta_j$) for the α_i and β_j used here.

[2] The instance of SM2 where each α_i is of the form (π_i ξ_i), where π_1,...,π_n are all the predicates in Γ, and where the domain is finite is analogous to the result in [Lifschitz 1985] relating Simplest Models (actually cwa) to Minimal Models.

$(\exists W((\text{world } W)\wedge([W]\Gamma)\wedge(\text{Smallest } W\ \Gamma\ \alpha_i)))\rightarrow \forall U(((\text{world } U)\wedge([U]\Gamma)\wedge(\text{Minimal } U\ \Gamma\ \alpha_i))\rightarrow(\text{Smallest } U\ \Gamma\ \alpha_i))$

which follows from:

$((\text{world } W)\wedge([W]\Gamma)\wedge(\text{Smallest } W\ \Gamma\ \alpha_i)\wedge(\text{world } U)\wedge([U]\Gamma)\wedge(\text{Minimal } U\ \Gamma\ \alpha_i))\rightarrow(\text{Smallest } U\ \Gamma\ \alpha_i)))$

Unfolding Smallest and Minimal gives:

$(\text{world } W)\wedge([W]\Gamma)\ \wedge\forall J(((\text{world } J)\wedge([J]\Gamma))\rightarrow(W\leq\alpha\ J)))\wedge$

$(\text{world } U)\wedge([U]\Gamma)\ \wedge(\forall J((\text{world } J)\wedge([J]\Gamma)\wedge(J\leq\alpha\ U))\rightarrow(U\leq\alpha\ J)))\rightarrow\forall J(((\text{world } J)\wedge([J]\Gamma))\rightarrow(U\leq\alpha\ J))))$

Noting that all variables are worlds, which entail Γ let us pretend these are sorted quantifiers of that ilk. Writing the above as sorted quantifiers gives: $(\forall J(W\leq\alpha\ J)\ \wedge\forall J((J\leq\alpha\ U)\rightarrow(U\leq\alpha\ J)))\rightarrow\forall J(U\leq\alpha\ J)$

which is implied by: $(\forall J(W\leq\alpha\ J)\ \wedge\forall J((J\leq\alpha\ U)\rightarrow(U\leq\alpha\ J)))\rightarrow(U\leq\alpha\ J)$

Letting the first J quantifier have the instances: U and J, and letting the second J quantifier have the instance W gives: $(W\leq\alpha\ U)\wedge(W\leq\alpha\ J)\ \wedge((W\leq\alpha\ U)\rightarrow(U\leq\alpha\ W))\rightarrow(U\leq\alpha\ J)$

which simplifies to: $(W\leq\alpha\ U)\wedge(W\leq\alpha\ J)\ \wedge(U\leq\alpha\ W))\ \rightarrow(U\leq\alpha\ J)$

Since $\leq\alpha$ is transitive the second and third hypotheses imply the conclusion. QED

T2 suggests the importance of Minimal Worlds as opposed to Smallest Worlds lies in the case where the Smallestworlds is not possible. A simple example of this case is in dealing with disjunctive information when the negation of the components of the disjunction is defaults:

$(\text{Smallestworlds } ((P\ a)\vee(P\ b))\ \ (P\ x))\equiv \#f$

$(\text{Minworlds } ((P\ a)\vee(P\ b))\ \ (P\ x))\equiv\ (\forall x((P\ x)\leftrightarrow(x=a)))\vee(\forall x((P\ x)\leftrightarrow(x=b)))$

The definition of Minimal Worlds constitutes the World Logic Representation. We now prove the existence of two other equivalent representations of Minimal Worlds. We prove that Minimal Worlds are equivalent to the "infinite disjunction" of a necessary equivalence which is represented in Z Modal Logic and that it is also equivalent to Circumscription which is representable in SOL. Quantified Possibility (i.e., $(Qpos\ \kappa\ \Gamma\ \alpha_i)$) with a sentence κ representing the resulting knowledgebase, a sentence Γ representing the conjunction of the initial nonlogical axioms, and a conjunction of defaults which imply α_i if α_i is possible with κ, is defined as follows:

$(Qpos\ \kappa\ \Gamma\ \alpha_i)\ =df\ \Gamma\wedge\ \wedge_{i=1,n}\forall\xi_i(<\kappa>\neg\alpha_i\rightarrow\neg\alpha_i)$

From this definition we form the necessary equivalence representing the solutions as follow[1],[2]:

$$\kappa\equiv(Qpos\ \kappa\ \Gamma\ \alpha_i)$$

A necessary equivalence may have zero or more solutions including an infinite number of solutions. When a necessary equivalence has more than one solution there the question as to which solution is to be used arises . One way of avoiding this question is to adopt the skeptical approach of only accepting what is common to all the solutions. This is achieved with the "$\exists k(k\wedge$" idiom as in:

$$\exists k(k\wedge\ (k\equiv(Qpos\ k\ \Gamma\ \alpha_i)))$$

This may be read as the (possibly infinite) disjunction of all the solutions k to the necessary equivalence. When there are a finite number of solutions $\beta_1,\ldots,\beta_n$ to such a necessary equivalence represented as: $(k\equiv\beta_1)\vee\ldots\vee(k\equiv\beta_n)$, the sentence $\exists k(k\wedge(k\equiv(Qpos\ k\ \Gamma\ \alpha_i)))$ is equivalent to $\exists k(k\wedge((k\equiv\beta_1)\vee\ldots\vee(k\equiv\beta_n)))$ which by the distribution properties of $\wedge$ and $\vee$ and by the fact that $\exists$ associates through $\vee$ gives the equivalent expression:

$(\exists k(k\wedge(k\equiv\beta_1)))\vee\ldots\vee(\exists k(k\wedge(k\equiv\beta_n)))$.

[1] Necessary Equivalences in Modal Logics were first discussed in [Brown 1986].

[2] This particular type of necessary equivalence is a generalization of fixed-point representations where quantifiers are allowed to cross the scope of defaults of both Default Logic [Reiter 1980] consisting only of defaults of the form: $\#t{:}\neg\alpha/\neg\alpha$ and of the Kernel of Autoepistemic Logic where the modal sentences are of the form: $(\neg L\alpha)\rightarrow\neg\alpha$. (see [Konolige 1987, Konolige 1987b, Brown 1987, Brown 1989, Antoniou 1997]. Unlike a number of other generalizations of these logics to the case where where free variables may occur in α [Konolige 1989], this representation in World Logic obeys all the normal laws of FOL, SOL, and S5 Modal Logic.

Since $\exists k(k\wedge(k\equiv\beta_i))$ is logically equivalent to just β_i, the result will then be equivalent to the disjunction of solutions: $\beta_1\vee\ldots\vee\beta_n$. However, if there are an infinite number of solutions the existential quantifier is not eliminatable in this manner.

SK1: $(\text{Minimal I }\Gamma\ \alpha_i)\leftrightarrow([\Gamma\wedge\wedge_{i=1,n}\forall\xi_i(\alpha_i\rightarrow([I]\alpha_i))]\ \wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow\alpha_i))$

proof: $(\text{Minimal I }\Gamma\ \alpha_i)$. Unfolding the definitions gives: $\forall J(((\text{world J})\wedge([J]\Gamma)\wedge(J\leq_\alpha I))\rightarrow(I\leq_\alpha J))$

and then: $\forall J(((\text{world J})\wedge([J]\Gamma)\wedge(\wedge_{i=1,n}\forall\xi_i(([J]\alpha_i)\rightarrow([I]\alpha_i))))\rightarrow\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow([J]\alpha_i)))$

Since J is a world we can pull out the J entailments giving:

$\forall J((\text{world J})\rightarrow[J]((\Gamma\wedge\wedge_{i=1,n}\forall\xi_i(\alpha_i\rightarrow([I]\alpha_i))\rightarrow\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow\alpha_i)))$

By Prior's Law this is equivalent to: $[](((\Gamma\wedge\wedge_{i=1,n}\forall\xi_i(\alpha_i\rightarrow([I]\alpha_i))\rightarrow\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow\alpha_i))$

which is equivalent to: $[\Gamma\wedge\wedge_{i=1,n}\forall\xi_i(\alpha_i\rightarrow([I]\alpha_i))]\ \wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow\alpha_i)$ QED

We now show that the $(\text{Minworlds }\Gamma\ \alpha_i)$ is logically equivalent to the "infinite disjunction" of all solutions to the fixed point equivalence:

SK2: $(\text{Minworlds }\Gamma\ \alpha_i)\equiv\exists k(k\wedge(k\equiv(\text{Qpos k }\Gamma\ \alpha_i)))$

proof: $(\text{Minworlds }\Gamma\ \alpha_i)$. Unfolding the definition of Minworlds gives: $\exists I(I\wedge(\text{world I})\wedge([I]\Gamma)\wedge(\text{Minimal I }\Gamma\ \alpha_i))$

which by SK1 is just: $\exists I(I\wedge(\text{world I})\wedge([I]\Gamma)\wedge([\Gamma\wedge\wedge_{i=1,n}\forall\xi_i(\alpha_i\rightarrow([I]\alpha_i))]\ \wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow\alpha_i)))$

This is equivalent to: $\exists I(I\wedge(\text{world I})\wedge([I]\Gamma)\wedge([\Gamma\wedge\wedge_{i=1,n}\forall\xi_i((\langle I\rangle\neg\alpha_i)\rightarrow\neg\alpha_i))]\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow\alpha_i)))$

Since I is a world it entails: $\wedge_{i=1,n}\forall\xi_i((\langle I\rangle\neg\alpha_i)\rightarrow\neg\alpha_i))$and the above may be rewritten as:

$\exists I(I\wedge(\text{world I})\wedge([I](\Gamma\wedge\wedge_{i=1,n}\forall\xi_i((\langle I\rangle\neg\alpha_i)\rightarrow\neg\alpha_i)))\wedge([\Gamma\wedge\wedge_{i=1,n}\forall\xi_i((\langle I\rangle\neg\alpha_i)\rightarrow\neg\alpha_i)]\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow\alpha_i)))$

Using the definition of Qpos this becomes: $\exists I(I\wedge(\text{world I})\wedge([I](\text{Qpos I }\Gamma\ \alpha_i))\wedge([(\text{Qpos I }\Gamma\ \alpha_i)]\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow\alpha_i)))$

which we rewrite so as to combine the two Qpos subexpressions:

$(\exists I(I\wedge(\text{world I})\wedge\exists k((k\equiv(\text{Qpos I }\Gamma\ \alpha_i))\wedge([I]k)\ \wedge([k]\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow\alpha_i)))$

and then by S5 modal Logic as: $\exists k\exists I(I\wedge(\text{world I})\wedge([I]k)\wedge(k\equiv(\text{Qpos I }\Gamma\ \alpha_i))\wedge(\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow([k]\alpha_i)))$

Since $([I]k)$ implies that $\wedge_{i=1,n}\forall\xi_i(([k]\alpha_i)\rightarrow([I]\alpha_i))$ the above is equivalent to:

$\exists k\exists I(I\wedge(\text{world I})\wedge([I]k)\wedge(k\equiv(\text{Qpos I }\Gamma\ \alpha_i))\ \wedge(\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\leftrightarrow([k]\alpha_i)))$

Since $\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\leftrightarrow([k]\alpha_i)))$ is equivalent to $\wedge_{i=1,n}\forall\xi_i((\langle I\rangle\neg\alpha_i)\leftrightarrow(\langle k\rangle\neg\alpha_i))$, it allows I in Qpos to be replaced by k giving: $\exists k\exists I(I\wedge(\text{world I})\wedge([I]k)\wedge(k\equiv(\text{Qpos k }\Gamma\ \alpha_i))\ \wedge(\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\leftrightarrow([k]\alpha_i)))$

Again, since $([I]k)$ implies $\wedge_{i=1,n}\forall\xi_i(([k]\alpha_i)\rightarrow([I]\alpha_i)))$ this is equivalent to:

$\exists k\exists I(I\wedge(\text{world I})\wedge([I]k)\wedge(k\equiv(\text{Qpos I }\Gamma\ \alpha_i))\wedge(\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow([k]\alpha_i)))$

which is equivalent to: $\exists k\exists I(I\wedge(\text{world I})\wedge([I]k)\wedge(k\equiv(\text{Qpos k }\Gamma\ \alpha_i))\wedge(\wedge_{i=1,n}\forall\xi_i((\langle k\rangle\neg\alpha_i)\rightarrow\neg([I]\alpha_i)))$

Since $(k\equiv(\text{QPos k }\Gamma\ \alpha_i))$ implies $\wedge_{i=1,n}\forall\xi_i((\langle k\rangle\neg\alpha_i)\rightarrow([k]\neg\alpha_i))$, $[I]k$ implies $\wedge_{i=1,n}\forall\xi_i(([k]\neg\alpha_i)\rightarrow([I]\neg\alpha_i))$, and (world I) implies $\wedge_{i=1,n}\forall\xi_i([I]\neg\alpha_i)\rightarrow\neg([I]\alpha_i))$, from these three hypotheses we infer: $\wedge_{i=1,n}\forall\xi_i((\langle k\rangle\neg\alpha_i)\rightarrow\neg([I]\alpha_i))$.

Thus the formula above is equivalent to: $\exists k\exists I(I\wedge(\text{world I})\wedge([I]k)\wedge(k\equiv(\text{Qpos k }\Gamma\ \alpha_i)))$

By D1 this simplifies to just: $\exists k(k\wedge(k\equiv(\text{Qpos k }\Gamma\ \alpha_i)))$ QED

Circumscription in Second Order Logic is defined as:

$(\text{Circ }\Gamma\ \alpha_i)=\text{df }(\Gamma\wedge\forall P_1\ldots P_m(((\Gamma\{\pi_u/P_j\})\wedge\wedge_{i=1,n}\forall\xi_i((\alpha_i\{\pi_j/P_j\})\rightarrow\alpha_i\ )))\rightarrow\wedge_{i=1,n}\forall\xi_i(\alpha_i\rightarrow(\alpha_i\{\pi_j/P_j\})))$

where $\pi_i\ldots\pi_m$ are <u>all</u> the unmodalized predicate symbols in Γ and α_i.

CIRC1: $(\text{Minworlds }\Gamma\ \alpha_i)\equiv(\text{Circ }\Gamma\ \alpha_i)$

proof: $(\text{Minworlds }\Gamma\ \alpha_i)$. Unfolding Minworlds gives: $\exists I(I\wedge(\text{world I})\wedge([I]\Gamma)\wedge(\text{Minimal I }\Gamma\ \alpha_i))$

By SK1 this is: $\exists I(I\wedge(\text{world I})\wedge([I]\Gamma)\wedge[\Gamma\wedge\wedge_{i=1,n}\forall\xi_i(\alpha_i\rightarrow([I]\alpha_i))]\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow\alpha_i))$

Since I is a world this is equivalent to: $\exists I(I\wedge(\text{world I})\wedge([I]\Gamma)\wedge[I][\Gamma\wedge\wedge_{i=1,n}\forall\xi_i(\alpha_i\rightarrow([I]\alpha_i))]\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow\alpha_i))$

By T1 this is equivalent to:

$\exists I(I\wedge(\text{world I})\wedge([I]\Gamma)\wedge[I]\forall P_1\ldots P_m\ (((\Gamma\wedge\wedge_{i=1,n}\forall\xi_i(\alpha_i\rightarrow([I]\alpha_i)))\rightarrow\wedge_{i=1,n}\forall\xi_i(([I]\alpha_i)\rightarrow\alpha_i))\{\pi_j/P_j\}))$

where $\pi_i \ldots \pi_m$ are all the unmodalized predicates in Γ or any α_i. Pushing the substitutions into the unmodalized subformulas gives the equivalent sentence:

$\exists I(I \wedge (\text{world } I) \wedge ([I]\Gamma) \wedge [I]\forall P_1 \ldots P_m\ (((\Gamma\{\pi_j/P_j\}) \wedge \wedge_{i=1,n}\forall\xi_i((\alpha_i\{\pi_j/P_j\}) \rightarrow ([I]\alpha_i))) \rightarrow \wedge_{i=1,n}\forall\xi(([I]\alpha_i) \rightarrow (\alpha_i\{\pi_j/P_j\}))))$

Since I is a world, [I] may be pushed in with nestings [I][I] absorbed to [I] and the then pulled out giving:

$\exists I(I \wedge (\text{world } I) \wedge ([I](\Gamma \wedge \forall P_1 \ldots P_m\ (((\Gamma\{\pi_j/P_j\}) \wedge \wedge_{i=1,n}\forall\xi_i((\alpha_i\{\pi_j/P_j\}) \rightarrow \alpha_i)) \rightarrow \wedge_{i=1,n}\forall\xi_i(\alpha_i \rightarrow (\alpha_i\{\pi_j/P_j\})))))$

By D1 this is equivalent to: $\Gamma \wedge \forall P_1 \ldots P_m(((\Gamma\{\pi_j/P_j\}) \wedge \wedge_{i=1,n}\forall\xi_i((\alpha_i\{\pi_j/P_j\}) \rightarrow \alpha_i)) \rightarrow \wedge_{i=1,n}\forall\xi_i(\alpha_i \rightarrow (\alpha_i\{\pi_j/P_j\})))$ QED

Circumscription with explicit fixed predicates:

Our definition of Circumscription requires that $\pi_i \ldots \pi_m$ are all of the unmodalized predicate symbols in Γ and any α_i. This requirement is not necessary since any Circumscription not obeying that requirement (which we will call Circ*) is equivalent to a Circumscription which does:

$(\text{Circ}^*\ \Gamma\ \alpha_i) =_{df} \Gamma \wedge \forall P_1 \ldots P_m(((\Gamma\{\pi_u/P_j\}) \wedge \wedge_{i=1,n}\forall\xi_i((\alpha_i\{\pi_j/P_j\}) \rightarrow \alpha_i)) \rightarrow \wedge_{i=1,n}\forall\xi_i(\alpha_i \rightarrow (\alpha_i\{\pi_j/P_j\})))$

where $\pi_i \ldots \pi_m$ are some the unmodalized predicate symbols in Γ or any α_i.

CIRC2: $(\text{Circ}\ \Gamma\ (\alpha_i, (\rho_j\ \xi_j), \neg(\rho_j\ \xi_j))) \equiv (\text{Circ}^*\ \Gamma\ \alpha_i)$ where the ρ_j predicates are the ones missing from the π_i list.

proof: $(\text{Circ}\ \Gamma\ (\alpha_i, (\rho_j\ \xi_j), \neg(\rho_j\ \xi_j)))$. Unfolding Circ letting Q be the variables for the ρ predicates from gives:

$\forall P_1 \ldots P_m \forall Q_1 \ldots Q_l(((\Gamma\{\pi_j/P_j\}\{\rho_j/Q_j\})$

$\wedge\wedge_{i=1,n}\forall\xi_i((\alpha_i\{\pi_j/P_j\}\{\rho_j/Q_j\}) \rightarrow \alpha_i) \wedge\wedge_{j=1,l}\forall\zeta_i(((\rho_j\ \zeta_i)\{\rho_j/Q_j\}) \rightarrow (\rho_j\ \zeta_j)) \wedge\wedge_{j=1,l}\forall\zeta_i(((\neg\rho_j\ \zeta_j)\{\rho_j/Q_j\}) \rightarrow (\neg\rho_j\ \zeta_j)))$

$\rightarrow(\wedge_{i=1,n}\forall\xi_j(\alpha_i \rightarrow (\alpha_i\{\pi_j/P_j\}\{\rho_j/Q_j\} \wedge\wedge_{j=1,l}\forall\zeta_j((\rho_j\ \zeta_j) \rightarrow ((\rho_j\ \zeta_j)\{\rho_j/Q_j\})) \wedge\wedge_{j=1,l}\forall\zeta_j((\neg\rho_j\ \zeta_j) \rightarrow ((\neg\rho_j\ \zeta_j)\{\rho_j/Q_j\}))))$

which is equivalent to:

$\forall P_1 \ldots P_m \forall Q_1 \ldots Q_l(((\Gamma\{\pi_j/P_j\}\{\rho_j/Q_j\}) \wedge\wedge_{i=1,n}\forall\xi_i((\alpha_i\{\pi_j/P_j\}\{\rho_j/Q_j\}) \rightarrow \alpha_i) \wedge\wedge_{j=1,l}\forall\zeta_i((Q_j\ \zeta_j) \leftrightarrow (\rho_j\ \zeta_j)))$
$\rightarrow(\wedge_{i=1,n}\forall\xi_j(\alpha_i \rightarrow (\alpha_i\{\pi_j/P_j\}\{\rho_j/Q_j\})) \wedge\wedge_{j=1,l}\forall\zeta_j((\rho_j\ \zeta_j) \leftrightarrow (Q_j\ \zeta_j)))$

The $\wedge_{i=1,l}\forall\zeta_i((Q_j\ \zeta_j) \leftrightarrow (\rho_j\ \zeta_j))$ hypothesis allows each Q_j elsewhere to be replaced by ρ_j giving:

$\forall P_1 \ldots P_m \forall Q_1 \ldots Q_l(((\Gamma\{\pi_j/P_j\}\{\rho_j/\rho_j\}) \wedge\wedge_{i=1,n}\forall\xi_i((\alpha_i\{\pi_j/P_j\}\{\rho_j/\rho_j\}) \rightarrow \alpha_i) \wedge\wedge_{j=1,l}\forall\zeta_i((Q_j\ \zeta_j) \leftrightarrow (\rho_j\ \zeta_j)))$
$\rightarrow(\wedge_{i=1,n}\forall\xi_j(\alpha_i \rightarrow (\alpha_i\{\pi_j/P_j\}\{\rho_j/\rho_j\})) \wedge\wedge_{j=1,l}\forall\zeta_j((\rho_j\ \zeta_j) \leftrightarrow (\rho_j\ \zeta_j)))$

This simplifies to:

$\forall P_1 \ldots P_m \forall Q_1 \ldots Q_l(((\Gamma\{\pi_j/P_j\}) \wedge\wedge_{i=1,n}\forall\xi_i((\alpha_i\{\pi_j/P_j\}) \rightarrow \alpha_i) \wedge\wedge_{j=1,l}\forall\zeta_i((Q_j\ \zeta_j) \leftrightarrow (\rho_j\ \zeta_j))) \rightarrow \wedge_{i=1,n}\forall\xi_j(\alpha_i \rightarrow (\alpha_i\{\pi_j/P_j\}))$

Pushing the Q quantifiers to lowest scope gives:

$\forall P_1 \ldots P_m(((\Gamma\{\pi_j/P_j\}) \wedge\wedge_{i=1,n}\forall\xi_i((\alpha_i\{\pi_j/P_j\}) \rightarrow \alpha_i) \wedge\wedge_{j=1,l}\exists Q_j\forall\zeta_i((Q_j\ \zeta_j) \leftrightarrow (\rho_j\ \zeta_j))) \rightarrow \wedge_{i=1,n}\forall\xi_j(\alpha_i \rightarrow (\alpha_i\{\pi_j/P_j\}))$

Letting each Q_j be ρ_j then gives: $\forall P_1 \ldots P_m((\Gamma\{\pi_j/P_j\}) \wedge\wedge_{i=1,n}\forall\xi_i((\alpha_i\{\pi_j/P_j\}) \rightarrow \alpha_i)) \rightarrow \wedge_{i=1,n}\forall\xi_j(\alpha_i \rightarrow (\alpha_i\{\pi_j/P_j\}))$

Folding Circ* then gives: $(\text{Circ}^*\ \Gamma\ \alpha_i)$. QED

(SOL Single) Formulae Circumscription [McCarthy 1986] is the instance of Circ* where n=1. Dropping the superfluous i subscript, it may be expressed as:

$\Gamma \wedge \forall P_1 \ldots P_m(((\Gamma\{\pi_j/P_j\}) \wedge \forall\xi((\alpha\{\pi_j/P_j\}) \rightarrow \alpha)) \rightarrow \forall\xi(\alpha \rightarrow (\alpha\{\pi_j/P_j\})))$

where $\pi_1 \ldots \pi_m$ are some of the unmodalized predicate symbols in Γ or α.

(SOL) Parallel (Predicate) Circumscription [Lifschitz 1985] is the instance of Circ* where each α_i is a sentence beginning with a distinct predicate: $(\pi_i\ \xi_j)$ In this case n≤m. This may be written as:

$\Gamma \wedge \forall P_1 \ldots P_m(((\Gamma\{\pi_i/P_i\}) \wedge\wedge_{i=1,n}\forall\xi_i((P_i\ \xi_i) \rightarrow (\pi_i\ \xi_i))) \rightarrow \wedge_{i=1,n}\forall\xi_i((\pi_i\ \xi_i) \rightarrow (P_i\ \xi_i)))$

where $\pi_1 \ldots \pi_m$ are some of the unmodalized predicate symbols in Γ. The Circumscribed Predicates are $\pi_1 \ldots \pi_n$ and the Varying Predicates are $\pi_{n+1} \ldots \pi_m$. Rewriting the Varying Predicate with different variable and metavariable symbols and indices gives the usual formulation:

$\Gamma \wedge \forall P_1 \ldots P_n \forall Z_1 \ldots Z_m\ (((\Gamma\{\pi_i/P_i\}\{\rho_j/Z_j\}) \wedge\wedge_{i=1,n}\forall\xi_i((P_i\ \xi_i) \rightarrow (\pi_i\ \xi_i))) \rightarrow \wedge_{i=1,n}\forall\xi_i((\pi_i\ \xi_i) \rightarrow (P_i\ \xi_i)))$

where $\pi_1 \ldots \pi_n$ and $\rho_1 \ldots \rho_m$ are some of the unmodalized predicate symbols in Γ.

(FOL) Parallel (Predicate) Circumscription without any varying predicates: Ruling out any varying predicates and replacing the second order variables in the above sentence with schematic metavariables, we may infer the FOL axiom scheme suggested in [McCarthy 1980]: $\Gamma \wedge (((\Gamma\{\pi_i/\lambda\xi_i\beta_i\}) \wedge\wedge_{i=1,n}\forall\xi_i(\beta_i \rightarrow (\pi_i\ \xi_i))) \rightarrow \wedge_{i=1,n}\forall\xi_i((\pi_i\ \xi_i) \rightarrow \beta_i))$

where $\pi_1 \ldots \pi_m$ are some of the unmodalized predicate symbols in Γ.

4. Conclusion

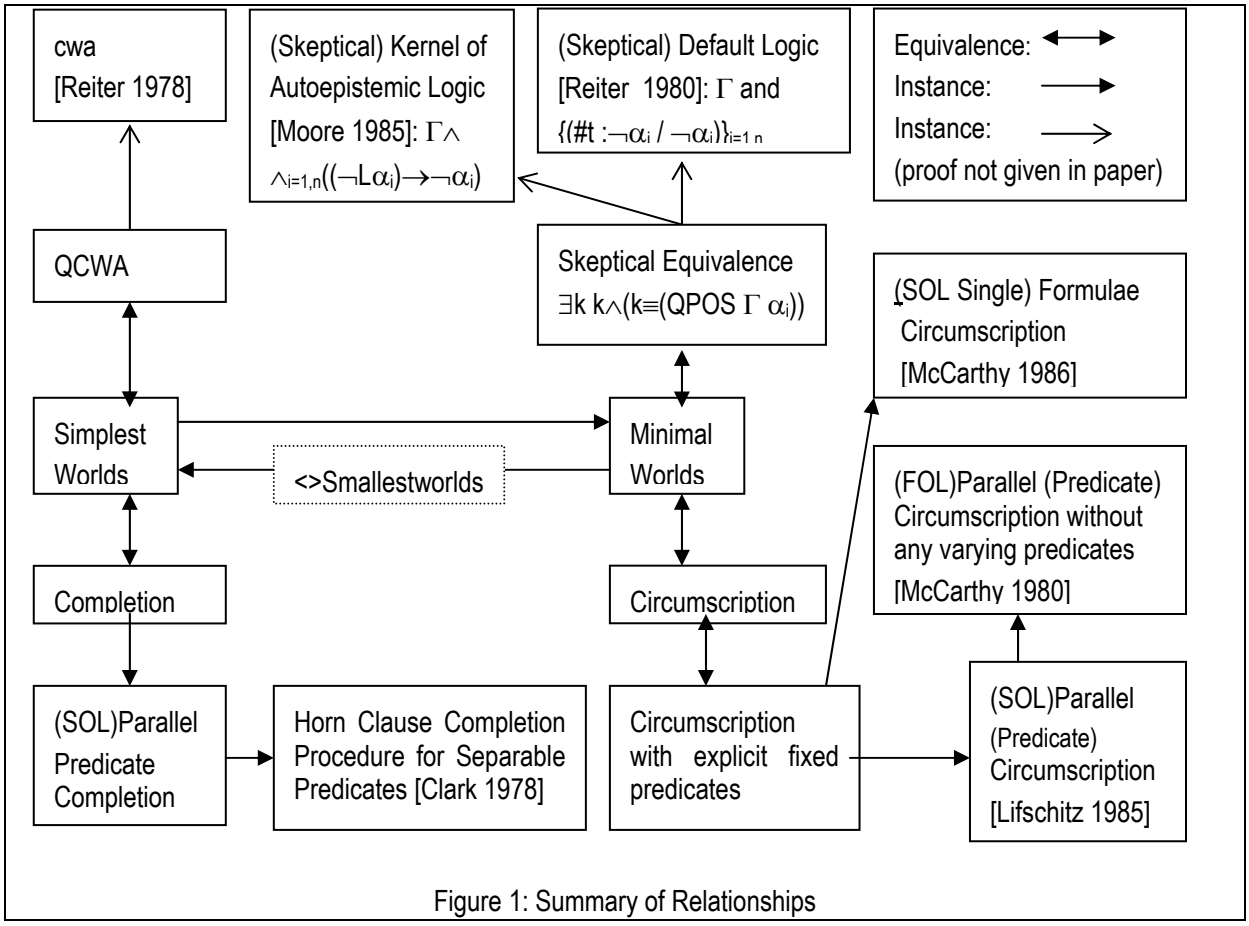


Figure 1: Summary of Relationships

We have proven in the Z Priorian Modal Second Order Logic that Smallestworlds are equivalent to both QCWA (theorem QCWA2) and Completion (theorem CL1). Likewise, we have proven that Minworlds is equivalent to both Skeptical Necessary Equivalences (theorem SK2) and Circumscription (theorem CIRC1). Thus, for each of these two types of worlds we have shown that they are represented in three different ways: namely in Z World Logic, Z Modal Logic, and Second Order Logic. The relationships among these systems are summarized in Figure 1 using the bi-arrows.

In addition, the two horizontal arrows show that Simplest Worlds entail Minimal Worlds (theorem SM1) and that Minimal Worlds entail Simplest worlds if Simplest Worlds are logically possible (theorem SM2). The other arrows show how a number of previously developed nonmonotonic systems are related to the representations discussed herein. These include three versions of Circumscription which are instances of the more general Circumscription with explicit fixed predicates given herein is in turn equivalent to Circumscription as defined herein. Clark's Completion Procedure for Horn Clauses whose consequences are implied by Parallel Predicate Completion given herein which in turn is an instance of Completion as herein defined. The relationships between QCWA and the Closed World Assumption cwa were discussed in [Brown 2005a]. The relationships between Skeptical Necessary Equivalences and the Skeptical Autoepistemic Kernel and Skeptical Default Logic were discussed in [Brown 2005b].

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Numbers: MIP-9526532, EIA-9818341, and EIA-9972843.

Bibliography

Antoniou, Grigoris 1997, *NonMonotonic Reasoning*, MIT. Press.

Brown, Frank M., 1986, "A Commonsense Theory of Nonmonotonic Reasoning", Proceedings of the 8th International Conference on Automated Deduction, Oxford England, July 1986, Lecture Notes in Computer Science 230, Springer-Verlag.

Brown, Frank .M. 1987, "The Modal Logic Z", In *The Frame Problem in AI"; Proceedings of the 1987 AAAI Workshop*, Morgan kaufmann, Los Altos, CA .

Brown, Frank M. 1989, "The Modal Quantificational Logic Z Applied to the Frame Problem", advanced paper *First International Workshop on Human & Machine Cognition*, May 1989 Pensacola, Florida. Abbreviated version published in *International Journal of Expert Systems Research and Applications, Special Issue: The Frame Problem. Part A.* eds. Kenneth Ford and Patrick Hayes, vol. 3 number 3, pp169-206, JAI Press 1990. Reprinted in *Reasoning Agents in a Dynamic World: The Frame problem*, editors: Kenneth M. Ford, Patrick J. Hayes, JAI Press 1991.

Brown, Frank M., 2005a, "Representing the Closed World Assumption in Modal Logic", Submitted to KDS 2005.

Brown, Frank M., 2005b, "Representing Skeptical Logics in Modal Logic", Submitted to KDS 2005.

Brown, Frank M., 2005c, "Z Priorian Modal Second Order Logic", Submitted to KDS 2005.

Clark, K [1978], Negation as failure. In: Logic and Databases, Gallaire, H. and Minker, J. (eds.), Plenum Press, New York, pp 293-322.

Konolige, Kurt, 1987, "On the Relation Between Default Theories and Autoepistemic Logic", *IJCAI87 Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 1987.

Konolige, Kurt, 1987b, "On the Relation Between Default Theories and Autoepistemic Logic", personally circulated new version of IJCAI87 paper correcting it to account for a counterexample found by Gelfond and Przymusinska.

Konolige, Kurt 1989, "On the Relation between Autoepistemic Logic and Circumscription Preliminary Report", IJCAI89.

Lifschitz, V. 1985, "Computing Circumscription" *IJCAI 9*,pages 121-127.

Lifschitz, V. 1985b, "Closed World Databases and Circumscription", *Artificial Intelligence*, 27.

McCarthy, J. 1980, "Circumscription -- A Form of Nonmonotonic Reasoning", *Artificial Intelligence*, volume 13.

McCarthy, J. 1986, "Applications of Circumscription to Formalizing Common-Sense Reasoning", *Artificial Intelligence*, volume 28.

Moore, R. C. 1985, "Semantical Considerations on Nonmonotonic Logic" *Artificial Intelligence*, 25.

Reiter, R 1978, On closed world databases. In: Logic and Databases, Gallaire, H. and Minker, J. (eds.), Plenum Press, New York, pp 55-76.

Reiter, R. 1980, "A Logic for Default Reasoning" *Artificial Intelligence*, 13.

Author's Information

Frank M. Brown – University of Kansas, Lawrence, Kansas, 66045, e-mail: brown@ku.edu.

Z PRIORIAN MODAL SECOND ORDER LOGIC

Frank M. Brown

***Abstract**: Multiple representations of a concept allow different automatic deduction approaches to be developed for theorems involving. Herein, we discuss three different monotonic logics. These monotonic logics are World Logic, Modal Logic, and Second Order Logic. In all these representations quantifiers obey the normal laws of both classical logc.*

***Keywords**: Z Priorian Modal Second Order Logic, Modal Logic, World Logic, Second Order Logic*

1. Introduction

Leibniz [Leibniz 1686] said that the truths of reason are true in all possible worlds. In Modal Logic this idea is explicated by saying that the logically necessary propositions are those which hold in all worlds. In a Priorian Modal Logic, worlds are represented in the logic itself as those propositions which are possible and which for

every other proposition, entail that other proposition or its negation [Prior & Fine 1977]. Since possibility and entailment are defined in terms of necessity, the properties of a world depend on the properties of the necessity symbol. Unfortunartely, even a modal logic as strong as S5 is not sufficient for axiomatizing worlds since its says nothing about what combinations of predicates are possible. For example In S5 we cannot even prove that a sentence consisting of a single predicate symbol is possible. Herein, we give axiom schemes sufficient to address this problem. We call the resulting Logic the Z Priorian Modal Second Order Logic.

Section 2 presents the syntax and axioms of the Z Priorian Modal Second Order Logic and three distinct sublanguages, namely Z World Logic, Z Modal Logic, and Second Order Logic, used for representing nonmonotonic logics in different ways. Some basic properties of the worlds are presented in Section 3. Finally, in Section 4 we draw some conclusions.

2. Z Priorian Modal Second Order Logic

The syntax and laws of the Z Priorian Modal Second Order Logic is given in sections 2.1 and 2.2 respectively. Its subllogis are defined in section 2.3.

2.1 Syntax of Z Priorian Modal Second Order Logic

The syntax of Z Priorian Modal Second Order Logic is an amalgamation of 3 parts:

The first part is a First Order Logic (i.e. FOL) represented as the six tuple: ($\rightarrow$, #f, $\forall$, *vars*, *predicates*, *functions*) where $\rightarrow$, #f, $\forall$, are logical symbols, *vars* is a set of object variable symbols, *predicates* is a set of predicate symbols each of which has an implicit arity specifying the number of terms associated with it, and *functions* is a set of function symbols each of which has an implicit arity specifying the number of terms associated with it. Roman letters x, y, and z possibly indexed with digits are also variables.

The second part is the extension to Second Order Logic (i.e. SOL) which consists of a set of predicate variables *predvars* and quantification over predicate variables (using $\forall$). Roman letters (other than x, y and z) possibly indexed with digits are used as predicate variables.

The third part is Modal Logic [Lewis 1918] which adds the necessity symbol: [].

Greek letters are used as syntactic metavariables. $\pi,\pi_1...\pi_n,\rho,\rho_1...\rho_n$ range over the predicate symbols, ϕ, $\phi_1...\phi_n$ range over function symbols, $\delta,\delta_1...\delta_n$ range over terms, $\gamma,\gamma 1,...\gamma_n$, range over the object variables ξ, $\xi_1...\xi_n,\zeta,\zeta_1...\zeta_n$ range over a sequence of object variables of an appropriate arity, $f,f_1...f_n$ range over predicate variables, and $\alpha,\alpha_1...\alpha_n,\beta,\beta_1...\beta_n,\chi,\chi_1...\chi_n,\Gamma$, κ range over sentences. Thus, the terms are of the forms: γ and (ϕ $\delta_1...\delta_n$), and the sentences are of the forms: ($\alpha \rightarrow \beta$), #f, ($\forall\gamma$ α), (π $\delta_1...\delta_n$), (f $\delta_1...\delta_n$), ($\forall f$ α), and ([]α). A zero arity predicate π, predicate variable f, or function ϕ is written as a sentence or term without parentheses, i.e., π instead of (π), f instead of (f), and ϕ instead of (ϕ). $\alpha\{\pi/\lambda\xi\beta\}$ is the sentence obtained from α by replacing all unmodalized occurences of π by $\lambda\xi\beta$ followed by lambda coversion. $\alpha\{\pi_i/\lambda\xi\beta_i\}_{i=1,n}$, abreviated as, $\alpha\{\pi_i/\lambda\xi\beta_i\}$ represents simultaneous substitutions. $\wedge_{i=1,n}\beta_i$ represents ($\beta_1\wedge...\wedge\beta_n$). The primitive symbols are listed in Figure 1. The defined symbols are listed in Figure 2 with their intuitive interpretations.

Symbol	Meaning
$\alpha\rightarrow\beta$	if α then β.
#f	falsity
$\forall\gamma\ \alpha$	for all γ, α.
$\forall f\ \alpha$	for all f, α.
[] α	α is logically necessary

Figure 1: Primitive Symbols

Symbol	Definition	**Meaning**
$\neg\alpha$	$\alpha \rightarrow$ #f	not α
#t	$\neg$ #f	truth
$\alpha\vee\beta$	$(\neg\ \alpha)\rightarrow\ \beta$	α or β
$\alpha\wedge\beta$	$\neg(\alpha \rightarrow \neg\beta)$	α and β
$\alpha\leftrightarrow\ \beta$	$(\alpha\rightarrow\ \beta) \wedge (\beta\rightarrow\ \alpha)$	α if and only if β
$\exists\gamma\ \alpha$	$\neg\forall\gamma\ \neg\alpha$	some γ is α
$\exists f\alpha$	$\neg\forall f\ \neg\alpha$	some f is α
<> α	$\neg[]\neg\alpha$	α is logically possible
[β] α	$[](\beta\rightarrow\alpha)$	β entails α
<β> α	$<>(\beta\wedge\alpha)$	α is possible with β
$\alpha\equiv\ \beta$	$[](\alpha\leftrightarrow\beta)$	α and β are synonymous
$\delta_1=\delta_2$	$(\pi\ \delta_1)\equiv(\pi\ \delta_2)$	δ_1 necessarily equals δ_2
$\delta_1\neq\delta_2$	$\neg(\delta_1=\delta_2)$	δ_1 does not necessarily equal δ_2
(det α)	$\forall f(([\alpha]f)\vee([\alpha]\neg f))$ where f is not in α	α is deterministic
(world α)	$(<>\alpha)\wedge$(det α)	α is a world

Figure 2: Defined Symbols of Z

2.2 Axiomatization of Z Priorian Modal Second Order Logic

The law of the Z Priorian Modal Second Order Logic is an amalgamation of five parts:

(a) The first part, which is given in Figure 3, consists of the laws of a FOL. The laws are: FOLR1, FOLR2, FOLA3, FOLA4, FOLA5, FOLA6, and FOLA7 [Mendelson 1964].

(b) The second part, which is given in Figure 4, consists of the additional laws needed for second order quantification. The three following laws SOLR2, SOLA4 and SOLA5 are the analogues of FOLR2, FOLA4 and FOLA5 for predicate variables.

(c) The Third part, which is given in Figure 5, consists of the laws MR0, MA1, MA2 and MA3 which constitute an S5 modal logic [Hughes &Cresswell 1968] [Carnap 1956]. When added to parts 1 and 2, they form Second Order Modal Quantificational logic similar to a Second Order version of [Bressan 1972].

(d) The fourth part, which is given in Figure 6, consists of the Priorian World extension of S5 Modal Logic. The PRIOR law [Prior and Fine 1977] states that a proposition is logically true if it is entailed by every world.

(e) The fifth part, which is given in Figure 7, consists of some laws axiomatizing what is logically possible. The law MA1a allows one to derive theorems such as $<>\forall x((\pi\ x)\leftrightarrow(x=\phi))$ and therefore extends the work on propositional possibilities in [Hendry and Pokriefka 1985] to possibilities in First Order Logic. The Laws MA1b and MA4 allow one to derive $(\phi_1\ \xi_1)\neq(\phi_2\ \xi_2)$ when ϕ_1 and ϕ_2 are distinct function symbols. MA4 states that at least two things are not necessarily equal just as there are at least two propositions: #t and #f.

FOLR1: from α and $(\alpha\rightarrow\ \beta)$ infer β
FOLR2: from α infer $(\forall\gamma\ \alpha)$
FOLA3: $\alpha \rightarrow (\beta\rightarrow\ \alpha)$
FOLA4: $(\alpha\rightarrow\ (\ \beta\rightarrow\ \rho)) \rightarrow ((\alpha\rightarrow\ \beta)\rightarrow\ (\alpha\rightarrow\ \rho))$
FOLA5: $((\neg\ \alpha)\rightarrow\ (\neg\ \beta))\rightarrow\ (((\neg\ \alpha)\rightarrow\ \beta)\rightarrow\alpha)$
FOLA6: $(\forall\gamma\ \alpha)\rightarrow\ \beta$ where β is the result of substituting an expression (which is free for the free positions of γ in α) for all the free occurrences of γ in α.
FOLA7: $(\forall\gamma\ (\alpha \rightarrow \beta)) \rightarrow\ (\alpha \rightarrow(\forall\gamma\ \beta))$ where γ does not occur in α.

Figure 3: The Laws of FOL

SOLR2: from α infer $(\forall f\ \alpha)$
SOLA6: $(\forall f\ \alpha)\rightarrow \beta$ where β is the result of substituting an expression (which is free for the free positions of f in α) for all the free occurrences of f in α.
SOLA7: $(\forall f(\alpha \rightarrow \beta)) \rightarrow (\alpha \rightarrow (\forall f\ \beta))$ where f does not occur in α.

Figure 4: Additional Laws of SOL

MR0: from α infer $([]\ \alpha)$ provided α was derived only from logical laws
MA1: $([]\alpha) \rightarrow \alpha$
MA2: $([\alpha]\beta) \rightarrow (([]\alpha)\rightarrow ([]\beta))$
MA3: $([]\alpha) \vee ([]\neg[]\alpha)$

Figure 5: Additional Laws of S5 Modal logic

PRIOR: $(\forall w((\text{world } w) \rightarrow ([w]\alpha)))\rightarrow []\alpha$ where w does not occur free in α.

Figure 6: Additional Law of a Piorian World Logic

MA1a: $([]\alpha) \rightarrow \alpha\{\pi/\lambda\zeta\beta\}$ where α may not contain an unmodalized occurence of a higher order variable.
MA1b: $([]\alpha) \rightarrow \alpha\{\phi/\lambda\zeta\delta\}$ where α may not contain an unmodalized occurence of a higher order variable. nor an unmodalized free object variable.
MA4: $\exists x\exists y(x\neq y)$

Figure 7: Additional Laws of a Z Modal Logic

2.3 Axiomatization of three sublanguages of the Z Priorian Modal Second Order Logic

The three sublanguages are described as follows:
(1) Z World Logic – The syntax consists of the symbols of FOL, SOL as it pertains to zero arity predicate variables (i.e. propositional variables) which are worlds, and []. The laws consist of the laws in Figure 3, 5, 6, and 7, and in addition the laws in Figure 4 only for the case of zero arity predicate variables which are worlds.[1]
(2) Z Modal Logic - The syntax consists of the symbols of FOL, SOL as it pertains to zero arity predicate variables (i.e. propositional variables), and []. The laws consist of the laws in Figures 3, 5, and 7, and in addition the laws in Figure 4 only for the case of zero arity predicate variables.
(3) Second Order Logic (i.e. SOL) – which consistets of the symbols of FOL and SOL and the laws in Figures 3 and 4.

3. Worlds and their Possibility

In Section 3.1 we show that worlds function as models do. These results follow in S5 Modal Logic. In section 3.2 we illustrate how worlds are related to Necessity using the PRIOR axiom in Figure 7. In Section 3.3 we generalize MA1a and MA1b to involve multiple substitutions. Finally, in section 3.4 we relate necessity to classical Second Order Logic using axioms MA1a and PRIOR.

3.1 World Entailments

Some basic theorems of S5 modal quantificational logic are given in Figure 8. Since many of the relations are only implications, necessity cannot be pushed through all the logical connectives. This makes automatic deduction somewhat difficult [Brown 1978]. However, if necessity is replaced by entailment by a world proposition then in S5 Modal logic that entailment may be pushed through all logical connectives. Thus world propositions play the role that models play in the metatheoretic semantics. Entailment by a world proposition is thus the analogue of satisfibility by a model. This provides a "semantics" divorced from any specific syntax, and thus allows general semantical laws to be expressed more easily without being cluttered up with syntax. Theorems W1, W2, W3, W4, W5, W6, W7, W8, W9, W10, and W11 in Figure 9 show how such entailments are reduced through the symbols of classical logic. Likewise, theorems W12 and W13 show that such entailments

[1] [Areces&Heguiabehere 2002] describes a World Logic with special symbols which may be defined as follows: $@w\alpha$ =df $[w]\alpha$ and $\downarrow w\alpha$ =df $\exists w(w\wedge(\text{world } w)\wedge([w]\alpha))$

disappear in front of modal symbols as one would expect in the framework of a modal logic at least as strong as S5. These entailments were used in [Brown 1979].

B1: $([]($\alpha\wedge$q)) $\leftrightarrow$ (([]α)$\wedge$([]q))
B2 $([]($\alpha\vee$q)) $\leftarrow$ (([]α)$\vee$([]q))
B3: $([]($\alpha\rightarrow$q)) $\rightarrow$ (([]α)$\rightarrow$([]q))
B4: $([]($\alpha\leftrightarrow$q)) $\rightarrow$ (([]α)$\leftrightarrow$([]q))
B5: ([]#t) $\leftrightarrow$ #t
B6: ([]#f) $\leftrightarrow$ #f
B7: $([]\neg\alpha) \rightarrow (\neg([]\alpha))$
B8: $([](\forall\gamma\alpha))\leftrightarrow (\forall\gamma([]\alpha))$
B9: $([](\exists\gamma\alpha))\leftarrow (\exists\gamma([]\alpha))$
B10: $([](\forall f\alpha))\leftrightarrow (\forall f([]\alpha))$
B11: $([](\exists f\alpha))\leftarrow (\exists f([]\alpha))$
B12: $([]([]\alpha)) \leftrightarrow ([]\alpha)$
B13: $([](<>\alpha)) \leftrightarrow (<>\alpha)$

Figure 8: Basic Laws of S5

W1: $([w](\alpha\wedge\beta))\leftrightarrow (([w]\alpha)\wedge([w]\beta))$
W2: (det w) $\rightarrow$ $(([w](\alpha\vee\beta)) \leftrightarrow (([w]\alpha)\vee([w]\beta)))$
W3: (det w) $\rightarrow$ $(([w](\alpha\rightarrow\beta)) \leftrightarrow (([w]\alpha)\rightarrow([w]\beta)))$
W4: (det w) $\rightarrow$ $(([w](\alpha\leftrightarrow\beta)) \leftrightarrow (([w]\alpha)\leftrightarrow([w]\beta)))$
W5: $(([w]\#t) \leftrightarrow \#t)$
W6: (<> w) $\rightarrow$ $(([w]\#f) \leftrightarrow \#f)$
W7: (world w)$\rightarrow$ $(([w](\neg\alpha)) \leftrightarrow (\neg([w]\alpha)))$
W8: $(([w](\forall\gamma\alpha))\leftrightarrow (\forall\gamma([w]\alpha)))$
W9: (det w) $\rightarrow$ $(([w](\exists\gamma\alpha))\leftrightarrow (\exists\gamma([w]\alpha)))$
W10: $(([w](\forall f\alpha))\leftrightarrow (\forall f([w]\alpha)))$
W11: (det w) $\rightarrow$ $(([w](\exists f\alpha))\leftrightarrow (\exists f([w]\alpha)))$
W12: (<> w) $\rightarrow$ $(([w]([]\alpha)) \leftrightarrow ([]\alpha))$
W13: (<> w) $\rightarrow$ $(([w](<>\alpha)) \leftrightarrow (<>\alpha))$

Figure 9: Reduction of World Entailments

3.2 Consequences of Prior's axiom

PRIOR says that a proposition is logically necessary if it is entailed by every world proposition. This law was implied in [Leibniz 1686] and has been used by a number of authors including [Prior&Fine 1977, Fine 1970]. It is equivalent to saying that every possible proposition has a world that entails it:

$$(<>\alpha) \rightarrow \exists w((\text{world } w)\rightarrow([w]\alpha))$$

The PRIOR axiom allows any necessity symbol to be replaced by an entailment by all world propositions in accordance with the laws in Figure 10 below. This is important because the entailments can then be reduced to lowest scope using the laws in Figure 9.

N1: $([]\alpha) \leftrightarrow (\forall w((\text{world } w)\rightarrow([w]\alpha)))$
P1: $(<>\alpha) \leftrightarrow (\exists w((\text{world } w)\rightarrow([w]\alpha)))$
D1: $(\exists w(w\wedge(\text{world } w)\wedge[w]\alpha))\leftrightarrow\alpha$
C1: $(\forall w((w\wedge(\text{world } w))\rightarrow[w]\alpha))\leftrightarrow\alpha$
DC:$(\exists w(w\wedge(\text{world } w)\wedge(fw))\leftrightarrow(\forall w((w\wedge(\text{world } w))\rightarrow(fw)))$ (w may not occur free in α in laws: N1, P1, D1, C1)

Figure 10: Necessity and Possibility as Worlds

3.3 Parallel Versions of MA1a and MA1b

First we note that MA!a and MA1b may be modified to include necessary conclusions:

TMA1a[]: $([]\alpha) \rightarrow []([\alpha\{\pi/\lambda\xi\beta\})$

where α does not contain any unmodalized occurence of a higher order variable

proof: From Axiom MA1a: $([]\alpha) \rightarrow \alpha\{\pi/\lambda\xi\beta\}$ the inference rule MR0 gives: $[](([]\alpha) \rightarrow (\alpha\{\pi/\lambda\xi\beta\}))$. By MA2 this implies: $([]([]\alpha))\rightarrow ([](\alpha\{\pi/\lambda\xi\beta\}))$ which by the laws of S4 Modal logic gives: $([]\alpha)\rightarrow ([](\alpha\{\pi/\lambda\xi\beta\}))$. QED

TMA1b[]: $([]\alpha) \rightarrow []([\alpha\{\phi/\lambda\xi\delta\})$

where α does not contain any unmodalized occurence of a higher order variable

proof: From Axiom MA1b: $([]\alpha) \rightarrow \alpha\{\pi/\lambda\xi\delta\}$ the inference rule MR0 gives: $[](([]\alpha) \rightarrow (\alpha\{\pi/\lambda\xi\delta\}))$. By MA2 this implies: $([]([]\alpha))\rightarrow ([](\alpha\{\pi/\lambda\xi\delta\}))$ which by the laws of S4 Modal logic gives: $([]\alpha)\rightarrow ([](\alpha\{\pi/\lambda\xi\delta\}))$. QED

MA1a and MA1b may be generalized to deal with multiple predicates and functions in parallel. Generalizing MA1a is difficult because unmodalized higher order variables may occur in any of the β_i:

TMA1a*: $([]\alpha) \rightarrow \alpha\{\pi_i/\lambda\xi_i\beta_i\}$ where α does not contain any unmodalized occurence of a higher order variable.

proof: By MA1 it suffices to prove: $([]\alpha)\rightarrow([]\alpha\{\pi_i/\lambda\xi_i\beta_i\})$. By N1 this is:

$([]\alpha)\rightarrow\forall w((\text{world } w)\rightarrow ([w]\alpha\{\pi_i/\lambda\xi_i\beta_i\}))$ which is: $(([]\alpha)\wedge(\text{world } w))\rightarrow([w]\alpha\{\pi_i/\lambda\xi_i\beta_i\}))$. Letting the ρ_j be any other unmodalized predicates in α other than the π_i predicates gives: $(([]\alpha)\wedge(\text{world } w))\rightarrow([w]\alpha\{\pi_i/\lambda\xi_i\beta_i\}\{\rho_j/\lambda\zeta_i\rho_j\}))$. Since α does not contain any unmodalized occurence of a higher order variable, the laws in Figure 9 allow the [w] to be pushed down to the predicates giving: $(([]\alpha)\wedge(\text{world } w))\rightarrow\alpha\{\pi_i/\lambda\xi_i([w]\beta_i)\}\{\rho_j/\lambda\zeta_j([w]\rho_j)\}))$. Starting from $([]\alpha)$, we now use TMA1a[] (many times) to replace, in any order, each predicate π_i or ρ_j with $\lambda\xi_i([w]\beta_i)$ or $\lambda\zeta_j([w]\rho_j)$, respectively, giving: $(([]\alpha\{\pi_i/\lambda\xi_i([w]\beta_i)\}\{\rho_j/\lambda\zeta_j([w]\rho_j)\})\wedge(\text{world } w)) \rightarrow\alpha\{\pi_i/\lambda\xi_i([w]\beta_i)\}\{\rho_j/\lambda\zeta_j([w]\rho_j)\})$ which holds by axiom MA1. QED

TMA1b*: $([]\alpha) \rightarrow \alpha\{\phi_i/\lambda\xi_i\delta_i\}$

proof: Since there are no unmodalized higher order variables in any δ_i we use TMA1b[] to replace each function symbol ϕ_i with δ_i giving $[]\alpha\{\phi_i/\lambda\xi_i\delta_i\}$. The conclusion then follows using MA1. QED

3.4 Relation of [] to SOL

The PRIOR and MA1a laws provide a direct method for translating FOL Modal sentences into sentences of (non modal) Second Order Logic[1]:

T1: If Γ does not contain any unmodalized occurences of a higher order variable and if $\pi_1\ldots\pi_n$ are all the unmodalized predicate symbols in Γ then:

$([]\Gamma) \leftrightarrow \forall P_1\ldots\forall P_n (\Gamma\{\pi_i/P_i\})$

proof: The proof divides into two cases:

(a) $([]\Gamma) \rightarrow \forall P_1\ldots\forall P_n \Gamma\{\pi_i/P_i\}$ which gives: $([]\Gamma) \rightarrow \Gamma\{\pi_i/P_i\}$

By TMA1a* letting π_i be P_i gives: $\Gamma\{\pi_i/P_i\} \rightarrow \Gamma\{\pi_i/P_i\}$ which is a tautology.

(b) $(\forall P_1\ldots\forall P_n \Gamma\{\pi_i/P_i\}) \rightarrow []\Gamma$. By PRIOR we get: $(\forall P_1\ldots\forall P_n \Gamma\{\pi_i/P_i\}) \rightarrow \forall w((\text{world } w) \rightarrow ([w]\Gamma))$

which is equivalent to: $(\forall P_1\ldots\forall P_n \Gamma\{\pi_i/P_i\}) \wedge (\text{world } w) \rightarrow ([w]\Gamma)$. Letting P_i be $\lambda\xi_i[w](\pi_i\ \xi_i)$ gives:

[1] This theorem may be constrasted with the Logic described in [Montague 1960] where $([]\Gamma)$ would be $\forall P_1\ldots\forall P_n,\forall F_1\ldots\forall F_m(\Gamma\{\pi_i/P_i\}\{\phi_j/F_j\})$ where Γ does not contain any unmodalized occurences of a higher order variable, $\pi_1\ldots\pi_n$ are all the unmodalized predicate symbols in Γ and $\phi_1\ldots\phi_m$ are all the unmodalized function symbols in Γ. Therein, each F_j is a functional variable of the same arity as ϕ_j. Montague points out that his system does not obey the normal laws of logic across modal scopes (but thinks that that is a virtue). For example, unfolding = in $\forall x\exists y(x=y)$ gives, $\forall x\exists y((\pi\ x)\equiv(\pi\ y))$, which in Montagues system is equivalent to $\forall x\exists y\forall P((P\ x)\leftrightarrow(P\ y))$ which is #t. Likewise, unfolding = in $\exists y(\phi=y)$ gives: $\exists y((\pi\ \phi)\equiv(\pi\ y))$, which in Montague's system is $\exists y\forall P\forall f((P\ f)\leftrightarrow(P\ y))$ which is false if there are at least two distinct things. Thus $\forall x\exists y(x=y)$ cannot imply $\exists y(\phi=y)$ in Montague's system thereby contradicting the law of universal instantiation (over a modal scope).

$(\Gamma\{\pi_i/\ \lambda\xi_i[w](\pi_i\ \ \xi_i)\})\ \wedge\ (\text{world w}) \rightarrow ([w]\Gamma)$. Since Γ does not contain any unmodalized higher order variables and every unmodalized predicate is replaced by a [w] entailment, the laws in Figure 9 allow the [w] entailments to be pulled out giving $([w]\Gamma\{\pi_i/\ \lambda\xi_i(\pi_i\ \ \xi_i)\})\ \wedge\ (\text{world w}) \rightarrow ([w]\Gamma)$ or rather: $([w]\Gamma)\ \wedge\ (\text{world w}) \rightarrow ([w]\Gamma$ which is a tautology. QED.

An instance of this theorem is Leibniz's Law about the identity of indescernables: $(x=y)\leftrightarrow(\forall P((P\ x)\leftrightarrow(P\ y)))$.

4. Conclusion

The existence of the three different representations (Z World Logic, Z Modal Logic and SOL) for representing concepts will allow different automatic theorem proving approaches to be developed for those concepts. The World representation serves as semantics analogous to Model Theory and the Modal representation serves as the analogue of proof Theory, both with the advantage that quantified variables are allowed to cross the scope of the analogues to both the model theoretic satisfaction and proof theoretic concept of being derivable. An application of these three logics to nonmonotonic reasoning is given in [Brown 2005].

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Numbers: MIP-9526532, EIA-9818341, and EIA-9972843.

Bibliography

Areces, Carlos and Heguiabehere, Juan HyLoRes 1.0: Direct Resolution for Hybrid logics, CADE18, LNAI 2392, 2002.

Bressan, Aldo 1972. *A General Interpreted Modal Calculus,* Yale University Press,.

Brown, Frank M., 1978. "A Sequent Calculus for Modal Quantificational Logic", *3rd AISB/GI Conference Proceedings,* Hamburg, July 1978.

Brown, Frank M. 1979. "A Theorem Prover for Meta theory", *Proceedings Fourth Workshop on Automated Deduction*, Austin, Texas.

Brown, Frank M., 1986 "A Commonsense Theory of Nonmonotonic Reasoning", Proceedindings of the 8th International Conference on Automated Deduction, Oxford England, July 1986, Lecture Notes in Computer Science 230, Springer-Verlag.

Brown, Frank .M. 1987. "The Modal Logic Z", In *The Frame Problem in AI"; Proceedings of the 1987 AAAI Workshop*, Morgan kaufmann, Los Altos, CA .

Brown, Frank M. 1989. "The Modal Quantificational Logic Z Applied to the Frame Problem", advanced paper *First International Workshop on Human & Machine Cognition*, May 1989 Pensacola, Florida. Abreviated version published in *International Journal of Expert Systems Research and Applicatiions, Special Issue: The Frame Problem. Part A.* eds. keneth Ford and Pattrick Hayes, vol. 3 number 3, pp169-206 JAI Press 1990. Reprinted in *Reasoning Agents in a Dynamic World: The Frame problem*, editors: Kenneth M. Ford, Patrick J. Hayes, JAI Press 1991.

Brown, Frank M. 2005, "Nonmonotonic Systems based on Smallest and Minimal Worlds represented in World Logic, Modal Logic, and Second Order Logic", Summitted to this conference.

Carnap, Rudolf 1956., *Meaning and Necessity: A Study in the Semantics of Modal Logic*, The University of Chicago Press,.

Fine, k. 1970. "Propositional Quantifiers in Modal Logic" *Theoria* 36, p336--346.

Hendry, Herbert E. and Pokriefka, M. L. 1985. "Carnapian Extensions of S5", *Journal of Phil. Logic 14*,.

Hughes, G. E. and Creswell, M. J., 1968. *An Introduction to Modal Logic*, Methuen & Co. Ltd., London.

Leibniz, G. 1686, "Necessary and Contingent Truths" 1686, *Leibniz Philosophical Writings*, Dent&Sons. 1973.

Lewis, C. I. 1936. Strict Implication, *Journal of Symbolic Logic*, vol I..

Mendelson, E. 1964. *Introduction to Mathematical Logic*, Van Norstrand, Reinhold Co., New York.

Montague, Richard 1960, "Logical Necessity, Physical Necessity, Ethics, and Quantifiers", Inquiry 4: p259-269.

Prior, A. N. & Fine, k. 1977. *Worlds, Times, and Selves*, Gerald Duckworth.

Author's Information

Frank M. Brown – University of Kansas, Lawrence, Kansas, 66045, e-mail: brown@ku.edu.