



Canadian Artificial Intelligence Intelligence Artificielle au Canada

Autumn 1998

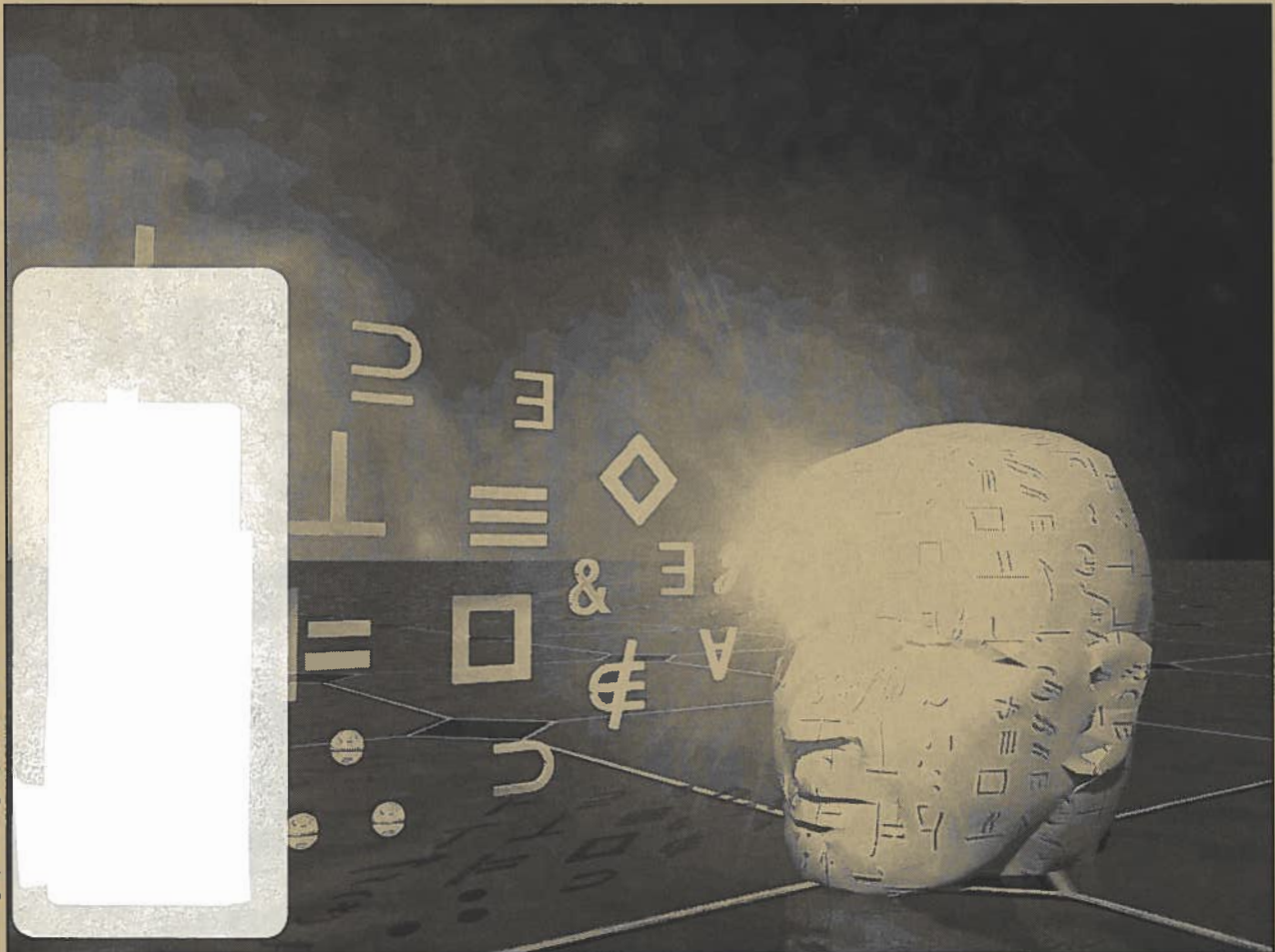
No. 42

automne 1998

An official publication of CSCSI, the Canadian Society for Computational Studies of Intelligence
Une publication officielle de la SCEIO, la Société canadienne pour l'étude de l'intelligence par ordinateur

Special Issue
Artificial Intelligence in Network Management

Edition speciale
L'intelligence Artificielle dans la gestion de réseaux





Canadian Artificial Intelligence

Intelligence Artificielle au Canada

Autumn 1998

No. 42

automne 1998

Canada's National AI magazine.

Editor · Rédacteur en Chef

Sue Abu-Hakima

Guest Editor · Rédacteur Invité

André Vellino

Editor Emeritus · Rédacteur Emeritus

Peter Turney

Managing Editor · Chef de Pupitre

Arlene Merling

Production Team · Equipe de production

Arlene Merling, Greg Klymchuk, Carol Tubman

Academia · Académiques

Vacant

Canadian AI Success Stories

Récits de Succès d'IA au Canada

Peter Turney

PRECARN Update · Nouvelles de PRECARN

Harry Rogers

Book Reviews · Critiques de livres

Graeme Hirst

Translation · Traduction

Alain Désilets, Benoit Farley, Norman Vinson

Advertising · Publicité

Arlene Merling

Canadian Artificial Intelligence is published three times a year by the Canadian Society for Computational Studies of Intelligence (CSCSI). Intelligence Artificielle au Canada est publiée trimestriellement par la Société canadienne pour l'étude de l'intelligence par ordinateur (SCEIO). Canadian Publications Mail Product Sales Agreement No. 507032.

ISSN 0823-9339

Copyright © 1998, Canadian Society for Computational Studies of Intelligence. All rights reserved; Canadian Artificial Intelligence may not be reproduced in any form without the written permission of the editors. Printed in Canada by Sundog Printing Limited. The opinions expressed herein are those of their respective authors and are not necessarily those of their employers, CSCSI, Canadian Artificial Intelligence, the editors, CIPS, or the National Research Council.
Copyright © 1998, Société canadienne pour l'étude de l'intelligence par ordinateur. Tout droit réservé; Intelligence artificielle au Canada ne doit être reproduite par quelque moyen que ce soit sans le consentement écrit des éditeurs. Imprimée au Canada par Sundog Printing Limited. Les opinions exprimées dans ce magazine sont celles de leurs auteurs respectifs et non pas nécessairement celles de leurs employeurs de la SCEIO, de l'Intelligence artificielle au Canada, des éditeurs, de l'Association canadienne informatique ou de Conseil National de Recherches du Canada.

CSCSI Executive

President · Président:

Fred Popowich, School of Computing Science, Simon

Fraser University, Burnaby BC V5A 1S6;

popowich@cs.sfu.ca

Past-President · Président Précédent:

Stan Matwin, Computer Science Dept., University of

Ottawa, Ottawa, ON K1N 6N5; stan@csi.uottawa.ca

Vice-President · Vice-Président:

Renee Elio, Department of Computer Science, University

of Alberta, Edmonton, AB T6G 2E1;

ree@cs.ualberta.ca

Secretary · Secrétaire:

Guy Mineau, Département d'informatique, Université

Laval, Sainte-Foy, Québec G1K 7P4;

Guy.Mineau@IFT.ulaval.ca

Treasurer · Trésorier:

Howard J. Hamilton, Department of Computer Science

University of Regina, Regina, SK S4S 0A2

hamilton@cs.uregina.ca

Editor · Rédacteur en Chef

Sue Abu-Hakima, Institute for Information Technology,

NRC, Ottawa, ON K1A 0R6; suhayya@ai.iit.nrc.ca

Contents Contenu

Communications 2 Communications

Feature Articles Gros Titres

Logic Based Formalisms for Natural Language Processing 4 Un modèle de satisfaction de contrainte pour Natural Language Processing
Fred Popowich

The Complexity of Propositional Proofs 8 Solutions de l'intelligence artificielle aux
Alasdair Urquhart

Non-Monotonic Logic is Impossible 19 La logique non-monotone est impossible
Charles G. Morgan

From Assumptions to Meaning 26 Test de signal intelligent minimu:
Veronica Dahl and Paul Tarau

Book Reviews 30 Critiques de livres

Canadian Artificial Intelligence welcomes submissions on any matter related to artificial intelligence.

Please send your contribution, electronic preferred, with an abstract and a short bio to:

Dan Fass

Editor, Canadian Artificial Intelligence

School of Computing Science

Simon Fraser University

Burnaby, B.C. V5A 1S6 or — fass@cs.sfu.ca

Book reviews and candidate books to review should !

Graeme Hirst

Canadian Artificial Intelligence

Department of Computer Science

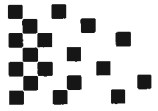
University of Toronto

Toronto, Ontario, Canada

M5S 1A4 or — gh@ai.toronto.edu



Recycled / Recyclable



Changes in Store for *Canadian Artificial Intelligence Magazine*

Welcome to our final long format magazine issue. Due to the amount of time required to put the long format magazine together, the Executive has decided to move the magazine to a shorter format. The new format will have more timely information for our members as well as some of the favourite long format information such as book reviews and a short article on a research topic.

The change to the new format will also bring a new editor for the magazine. Dan Fass of Simon Fraser University has agreed to take on the responsibility of putting out the shorter magazine. The production of *Canadian Artificial Intelligence* will also move to SFU.

I would like to take this opportunity to thank Arlene Merling for her tremendous efforts and dedication over six years in putting out the magazine from Calgary.

On a personal note, I am moving on to other responsibilities, namely my NRC spin-off company AmikaNow! Corporation which will be bringing intelligent software agent products into personalized workspaces for people.

I have enjoyed being the editor of the magazine for the last two years as well as co-editor with Peter Turney for two years before that. I thank NRC for giving us the opportunity to contribute to the Canadian artificial intelligence community. I wish my successors the best of luck and hope to continue to contribute to the Artificial Intelligence community in other ways.

All the best,
Sue Abu-Hakima
suhayya@amikanow.com

Visit our Website
www.cscsi.sfu.ca

**Canadian Artificial Intelligence is accessible on our website to
CSCSI/SCEIO members.**

**If you would like to try electronic access, please send an
e-mail message to fass@cs.sfu.ca, with the following information:**

- your full name
- your e-mail address
- a short user name (5-8 characters, for web access)
- a password (5-8 characters, for web access), and
- your CSCSI/SCEIO membership identification number, located on the mailing label of the magazine.

Become a Member of CSCSI/SCEIO

For more details, see page 35.

Special Issue: Logic and Artificial Intelligence

André Vellino

Ever since Aristotle treatises on syllogisms, the view that logic formalizes the process of thinking and knowing has been at the core of western intellectual thought. Its most ambitious apogee is the idea we owe to Leibniz, that logic is a "universal language" into which all questions can be formulated and that there are rules of logic that provide the "calculus ratiocinator" with which all questions, not just mathematical questions, could be settled.

It is not surprising that this ambitious logicist worldview, should have failed so spectacularly. Starting with Russell's paradoxes and going on through to Goedel's incompleteness theorem, these failures seem not to have deterred the advance of logic both as a science in its own right and as a tool with which to formalize human thinking. In particular, the discipline of artificial intelligence, indeed the advent of computing itself, can be seen as the modern day fruit of this logicist program. Concomitantly, a variety of logics have flourished in the latter half of this century: default logics, relevance logics, deontic logics, modal logics, many-valued logics, linear logics and the list goes on. Each one was devised in the ever elusive attempt to formalize one aspect of the process of human thinking and reasoning, be it "common sense reasoning," "reasoning about time" or what have you.

In the realm of logics for computation, first and second-order logics even rival even the lambda calculus as a paradigm for reasoning about and for performing computations. Not even the disappointments with the "5th Generation Computers" of the 1980s can detract from the virtues of, for example, logic programming, in fields such as computational linguistics, constraint programming and deductive databases.

A comprehensive issue of *Canadian Artificial Intelligence* on "Logic and Artificial Intelligence" would be next to impossible to publish; there are too many research areas and too many researchers in the field, even in Canada. Yet the papers in this issue cover a small but representative sample of some of the more important aspects of logic research as it applies to computing and AI. We have two survey articles and two special-interest articles.

Urquhart's survey article addresses the question of what the lower bounds are to the length of proofs in the propositional calculus. As he amply demonstrates, establishing tight lower bounds is no trivial matter yet essential to an understanding of the inherent limitations of automated theorem proving. This topic is all the more exciting because it contains many open problems in the foundation of logic. The other survey article, by Fred Popowich, brings us up to date on the applications of logic

for one of the most difficult of AI's undertakings: natural language processing. How is logic applied to define different species of linguistic information — syntax, semantics and pragmatics — and are how well does it fare at the task of formalizing their interrelationships?

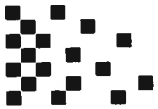
The two special-interest papers on non-monotonic logics contrast quite nicely both with each other and with the survey papers. With Charles Morgan's contribution, we are immersed in the ever-raging debate between those who argue that non-monotonic logic is a poor way of glossing over essentially statistical assumptions and those who argue that new non-monotonic inference can only be captured by a "logic" for non-monotonic connectives. These positions are articles of faith on both sides, and if nothing else, Morgan's "Proof" is sure to enrage someone (Ray Reiter, perhaps?). Who ever said that logic wasn't polemical?

Dahl and Tarau offer a no less novel report on their implementation of intuitionistic primitives in a logic-programming language and demonstrate their unexpected usefulness in assumption grammars. Assumption grammars, they claim, have more expressive power than other logic grammars because they permit the specification of rewriting transformations using only intuitionistic and (affine) linear assumption. They show how such a grammar can be used within a Prolog framework and illustrate its application with convincing examples.

As the articles in this issue illustrate, logic as a paradigm for doing AI is still alive and well. It is clear that logic still has, and will continue to have, a significant place as one of an array of conceptual frameworks with which to realize the goal of a thinking machine, even if it is no longer the universal language Leibniz had hoped it would be.



André Vellino is a research scientist at Nortel Technology, currently on secondment to the National Research Council. He has contributed to research in automated theorem proving, constraint logic programming and intelligent agents. He obtained his Ph.D. in logic at the University of Toronto in 1989 and is adjunct researcher professor in Cognitive Science at Carleton University.



Logic Based Formalisms for Natural Language

Introduction

Nous décrivons dans cet article l'évolution de la recherche sur l'utilisation, dans le traitement du langage naturel, de la logique non seulement pour représenter la sémantique ou le sens des expressions, mais aussi pour en représenter la structure, la syntaxe. On y décrit comment divers formalismes basés sur la logique ont évolué de façon à pouvoir spécifier et traiter de l'information linguistique, permettant ainsi d'exprimer différents types d'information linguistique (syntaxique, sémantique ou pragmatique) et aussi d'exprimer les dépendances entre ces types avec plus de concision.

Introduction

In the beginning there was logic. And it became the foundation of meaning representation for a great deal of natural language processing research. However, this article is not about different logics used to (attempt) to represent the meanings of natural language expressions, it is not about the shortcomings of various logics nor is it about their various strengths; McCawley (1981) has already described *Everything that Linguists have Always Wanted to Know about Logic but Were Ashamed to Ask*. Instead, it is an article about the evolution of natural language processing research to use logic for not only representing the semantics or meaning of expressions, but also for the structure or syntax of expressions. It describes how different *logic based formalisms* have evolved to allow the specification and processing of linguistic information, allowing one to express different types of linguistic information (be it syntactic, semantic or pragmatic, etc.) and to express dependencies between these different types of information more concisely.

Since the early work of Chomsky (1957), we have become familiar with the use of rewrite grammars, like context-free grammars or even transformational grammars, for describing the structure of natural language expressions. For example, the context-free grammar rule

$$\text{sentence} \rightarrow \text{noun_phrase}, \text{verb_phrase} \quad (1)$$

describes a sentence as being composed of two constructions, the *noun_phrase* and the *verb_phrase*, which have their own respective definitions, say:

$$\text{noun_phrase} \rightarrow \text{proper_noun} \quad (2)$$

$$\text{proper_noun} \rightarrow \textit{John} \quad (3)$$

$$\text{verb_phrase} \rightarrow \text{verb} \quad (4)$$

$$\text{verb} \rightarrow \textit{walks} \quad (5)$$

Thus, we have rules for describing the structure of the sentence *John walks*. For future reference, we will refer to

the above set of rules as part of our grammar G1. In our rules, the terminal symbols are italicized while the nonterminal symbols are not.

One can use grammar rules much like these to describe the syntax of natural language, and then associate a semantic rule with each grammar rule (Dowty, Wall and Peters 1981). The semantic rules allow logic formulae to be associated with each syntactic constituent in a sentence. This same approach was adopted, and expanded upon, within a linguistic theory called Generalized Phrase Structure Grammar (Gazdar, Klein, Pullum and Sag 1985). For the grammar rules from G1, we would effectively associate the following logic equations with each of the grammar rules, where for a nonterminal symbol X, the notation X' is used to refer to the logic expression associated with X.

$$\begin{aligned} \text{sentence} &\rightarrow \text{noun_phrase}, \text{verb_phrase} \\ \text{sentence}' &= \text{noun_phrase}'(\text{verb_phrase}') \quad (6) \end{aligned}$$

$$\begin{aligned} \text{noun_phrase} &\rightarrow \text{proper_noun} \\ \text{noun_phrase}' &= \text{proper_noun}' \quad (7) \end{aligned}$$

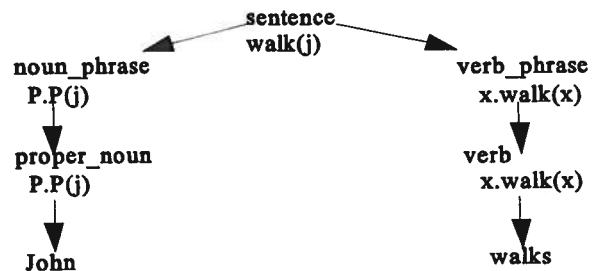
$$\begin{aligned} \text{proper_noun} &\rightarrow \textit{John} \\ \text{proper_noun}' &= \lambda P.P(j) \quad (8) \end{aligned}$$

$$\begin{aligned} \text{verb_phrase} &\rightarrow \text{verb} \\ \text{verb_phrase}' &= \text{verb}' \quad (9) \end{aligned}$$

$$\begin{aligned} \text{verb} &\rightarrow \textit{walks} \\ \text{verb}' &= \lambda x.\text{walk}(x) \quad (10) \end{aligned}$$

Using these rules, the analysis of the sentence *John walks* would result in the following syntactic and semantic constituents, where we get $\text{walk}(j)$ from applying the function $\lambda P.P(j)$ to its argument $\lambda x.\text{walk}(x)$. That is

$$\lambda P.P(j)(\lambda x.\text{walk}(x)) = \lambda x.\text{walk}(x)(j) = \text{walk}(j) \quad (11)$$



When exploring the usefulness of grammars like G1, one quickly becomes aware of the limitations and the need for finer grained distinctions in the linguistic knowledge. Indeed, once we introduce another form of the verb, a pronoun, and a generic common noun into the simple grammar to allow for coverage of sentences like *People walk* and *Everyone walks*, by naively introducing rules like:

noun_phrase \rightarrow generic_noun
 noun_phrase' = generic_noun' (12)

noun_phrase \rightarrow pronoun
 noun_phrase' = pronoun' (13)

generic_noun \rightarrow *People*
 generic_noun' = $\lambda P. \forall x. (\text{person}(x) \rightarrow P(x))$ (14)

pronoun \rightarrow *Everyone*
 pronoun' = $\lambda P. (\forall x. P(x))$ (15)

verb \rightarrow *walk*
 verb' = $\lambda x. \text{walk}(x)$ (16)

we also get analyses for the undesired ungrammatical sentences: *People walks* and *John walk*.

As an aside, grammar engineers, who develop large grammars constantly run into subtle variations of this same problem; the expansion of coverage can result in undesired side effects. In this simple example, the problem is that finer grained distinctions need to be made between different types of noun phrases and verb phrases. For instance, a noun phrase for *People* needs to state that it corresponds to a third person plural form, while *John* is third person singular. This information needs to be shared between the proper noun (or pronoun) and the noun phrase constituent. Similarly, we need to distinguish between the third person singular form of the present tense of the verb, *walks*, and the form associated with all other persons and numbers in the present tense, *walk*. Generalizations like these are easy to express if we replace our atomic grammar symbols like *noun_phrase* with parameterized versions like *noun_phrase(X)*, where the single argument could in this case be representing the person information as described above. What we are effectively doing is allowing logical terms, containing predicates, constants and/or variables, to appear as grammar symbols. The result is a more powerful formalism for describing the syntax of natural language: definite clause grammars (DCGs) (Pereira and Warren 1980), probably the most familiar example of what are known as logic grammars.

The popularity of DCGs is undoubtedly related to their close relationship to the Prolog programming language. It is straight forward to convert a DCG into Prolog clauses, and many implementations of Prolog perform this conversion automatically. This allows students and researchers to quickly develop simple grammars for use in

prototype artificial intelligent applications. The various argument positions in a term can be used for a wide range of linguistic information, and arguments are frequently used to constrain semantic information as well, thus providing an even closer link between syntactic and semantic information. The following example illustrates how some agreement and semantic information might be incorporated into a grammar related to G1, based on an approach introduced in Pereira and Shieber (1987) for processing natural language with Prolog, where lambda abstraction and function application can be simulated with unification. Note that $X^{\text{walk}(X)}$ is equivalent to $\lambda x. \text{walk}(x)$, and $(j^{\text{S}})^{\text{S}}$ is equivalent to $\lambda P. P(j)$: Prolog does not allow expressions of the form $P(x)$ where X is a variable.

sentence(S) \rightarrow noun_phrase(Person, X^{S}),
 verb_phrase(Person, X) (17)

noun_phrase(Person, Semantics) \rightarrow
 proper_noun(Person, Semantics) (18)

verb_phrase(Person, Semantics) \rightarrow
 verb(Person, Semantics) (19)

proper_noun(third, $(j^{\text{S}})^{\text{S}}$) \rightarrow *John* (20)

verb(third, $X^{\text{walk}(X)}$) \rightarrow *walks* (21)

The limitations of DCGs become apparent when dealing with larger grammars though. Many people have noted (see for instance Covington 1994) that it can be difficult to maintain DCGs in which the terms used as grammar symbols contain a large number of arguments. Not only does one have to remember which position corresponds to which, but one also has to have place holders for all the features which are not relevant. Moreover, it would be nice to have rules like

verb(\sim third) \rightarrow *walk*(22)

and have this be consistent with a first person pronoun.

Instead of having a logical term as a grammar symbol, it is possible to instead have a feature structure as a grammar symbol. Just as terms can be related through unification as is done in DCGs, so can feature structures. This is the approach adopted by PATR-II and related formalisms, as described in (Shieber 1986). One can even marry these two approaches, and have a feature structure appear as an argument in a definite clause style grammar, as summarized in (Covington 1994).

But, what is a feature structure? People from a computer programming background can view it as a record, a collection of different feature names, each of which has a value. This value may be primitive (or atomic), or may be

another record (feature structure). In addition, values may be shared: the value of one feature can be token identical to the value of another record. The real power of feature structures, however, lies with the unification operation which is applied to them. The unification of two feature structures is essentially just the union of their compatible information. If the two feature structures contain incompatible information, then their unification fails. Unification failure plays the fundamental role in preventing the construction of inappropriate constituents. Formal aspects of feature structures are discussed in Johnson's (1988) *Attribute-Value Logic and the Theory of Grammar*.

So, our DCG given earlier could also be presented in the following manner, where we associated feature structures with each atomic grammar symbol, where the sharing of values within a rule is represented through common numeric indices enclosed in vertical bars or boxes.

$$\begin{array}{c}
 \text{NP} \left[\begin{array}{l} \text{PER} \quad |1| \\ \text{SEM} \left[\begin{array}{l} \text{LAMBDA} \quad |2| \\ \text{BODY} \quad |3| \end{array} \right] \end{array} \right] \quad \text{VP} \left[\begin{array}{l} \text{PER} \quad |1| \\ \text{SEM} \quad |2| \end{array} \right] \\
 \text{SEM} \quad |3|
 \end{array} \quad (23)$$

$$\text{NP} \left[\begin{array}{l} \text{PER} \quad |1| \\ \text{SEM} \quad |2| \end{array} \right] \quad \text{PN} \left[\begin{array}{l} \text{PER} \quad |1| \\ \text{SEM} \quad |2| \end{array} \right] \quad (24)$$

$$\text{VP} \left[\begin{array}{l} \text{PER} \quad |1| \\ \text{SEM} \quad |2| \end{array} \right] \quad \text{V} \left[\begin{array}{l} \text{PER} \quad |1| \\ \text{SEM} \quad |2| \end{array} \right] \quad (25)$$

$$\begin{array}{c}
 \text{N} \left[\begin{array}{l} \text{PER} \quad \text{third} \\ \text{SEM} \left[\begin{array}{l} \text{LAMBDA} \left[\begin{array}{l} \text{LAMBDA} \quad j \\ \text{SEM} \quad |1| \end{array} \right] \\ \text{SEM} \quad |1| \end{array} \right] \end{array} \right] \quad \text{John} \\
 \text{SEM} \quad |1|
 \end{array} \quad (26)$$

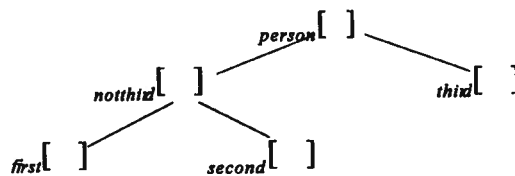
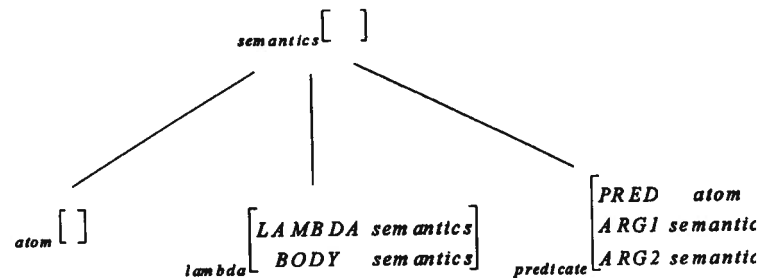
$$\begin{array}{c}
 \text{V} \left[\begin{array}{l} \text{PER} \quad \text{third} \\ \text{SEM} \left[\begin{array}{l} \text{LAMBDA} \quad x \\ \text{SEM} \left[\begin{array}{l} \text{PRED} \quad \text{walk} \\ \text{ARG1} \quad x \end{array} \right] \end{array} \right] \end{array} \right] \quad \text{walks} \\
 \text{SEM} \quad |1|
 \end{array} \quad (27)$$

we are dealing with greater amounts of information and more intricate dependencies the feature structure notation is preferable. There are also several standard non-graphic forms of the feature structure notation that could be used.

Additional descriptive power can be obtained by using typed feature structures as opposed to the untyped feature structures that we have just described. Detailed formal definitions related to typed feature structures and associated operations can be found in Carpenter's (1992) *The Logic of Typed Feature Structures*.

For our needs, we can think of a typed feature structure as an untyped feature structure, except that it also has associated with it a type from a type hierarchy. Unification of two typed feature structures work much like untyped feature structure unification, except that the two feature structures being unified must not be from incompatible parts of the hierarchy: one feature structure must be a subtype of the other (they can be of the same type). What's more, we allow a type to have more than one immediate supertype: we have a multiple inheritance hierarchy.

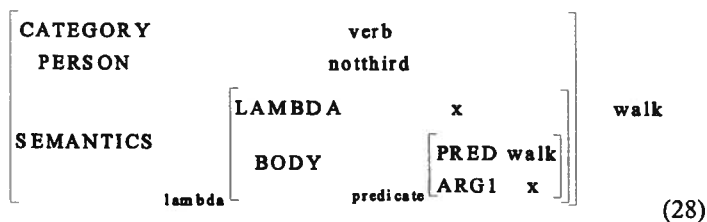
With respect to our simple grammar we introduced showing the use of feature structures, we could introduce the following type definitions for our different types of semantic and person information. Note that each feature



The semantic structure of the form X^S from our DCG example is represented as a feature structure itself, where X is considered to be the LAMBDA and S is considered to be the BODY of the lambda expression. Similarly, a predicate and its argument, such as walk(x), is also represented as a feature structure. Although for the simple example we have been looking at, the DCG notation is more concise, once

structure is preceded by a type name. The values of the individual features in a feature structure are also restricted to being of a specific type, as reflected by the appearance of the type name as the value of a feature.

Developing typed feature structures to encode linguistic information shares much in common with object oriented design and object oriented programming. A given typed feature structure inherits features, and values for its features from its supertypes. Thus certain pieces of linguistic information need only be specified in one location to be 'felt' in a wide range of locations. So any subtypes of *predicate* from the above hierarchy would have all the features shown above, plus any others introduced in the subtype. The types



of *semantics* and *predicate* are unifiable, but those of *predicate* and *lambda* are not. Using the above type hierarchy, one could have a grammar rule like the following, if we assume that *x* and *walk* are subtypes of *atom*.

Similarly, just as there was a close relationship between DCGs and a programming environment, specifically Prolog, there is a close relationship between typed feature structures and programming environments, notably ALE (Carpenter and Penn 1994) and LIFE (Ait-Kaci 1993).

Feature structures play an important role in many contemporary linguistic and computational linguistic theories. The logic-based formalisms that we have discussed here can be used to provide implementations of formalisms like GPSG and Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994), one of the most popular contemporary linguistic formalisms. This formalism is the basis of large scale applications in the area of natural language processing. Readers interested in obtaining more information about HPSG and related linguistic formalisms, along with information about applications and tools should visit <http://ling.ohio-state.edu/HPSG/Hpsg.html> (29)

So, what have we seen. We have seen how logic based formalisms can be used to describe linguistic information, and we have seen the increase in sophistication in the way that knowledge can be described and used by these formalisms. Computational tools are also widely available so that these formalisms can be used as the basis of applied and theoretical research. Indeed, there are a range of grammar development systems available for logic based grammars that make the development and maintenance of large amounts

of linguistic information possible.

References

1. Ait-Kaci, Hassan (1993). An Introduction to LIFE - Programming with Logic, Inheritance, Functions, and Equations,' Proceedings of the 10th International Logic Programming Symposium, Vancouver, BC, pp. 1- 17.
2. Carpenter, Bob (1992). *The Logic of Typed Feature Structures*. Cambridge University Press, Cambridge.
3. Carpenter, Bob and Gerald Penn (1994). *ALE - the Attribute Logic Engine User's Guide*. Philosophy Department, Canegie Mellon University, Pittsburgh, PA.
4. Chomsky, Noam (1957). *Syntactic Structures*. Mouton & Co.,TheHague.
5. Covington, Michael A. (1994). *Natural Language Processing for Prolog Programmers*, Prentice Hall, Englewood Cliffs,NJ.
6. Dowty, David R., Robert E. Wall and Stanley Peters (1981). *Introduction to Montague Semantics*. D. Reidel, Dordrecht.
7. Gazdar, Gerald, Ewan Klein, Geoffrey Pullum and Ivan Sag (1985). *Generalized Phrase Structure Grammar*. Basil Blackwell,Oxford.
8. Johnson, Mark. (1988). *Attribute-Value Logic and theTheory of Grammar*. CSLI Lecture Notes, Stanford University, CA.
9. McCawley, James D. (1981), *Everything that Linguists have Always Wanted to Know about Logic but Were Ashamed to Ask*. University of Chicago Press, Chicago.
10. Pereira, Fernando C.N. and Stuart M. Shieber (1987). *Prolog and Natural Language Analysis*, CSLI Lecture Notes, University of Chicago Press.
11. Pereira, Fernando C.N. and D.H.D. Warren (1980). Definite Clause Grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13(3): 231-278.
12. Pollard, Carl, and Ivan Sag (1987). *Information-Based Syntax and Semantics, Volume 1: Fundamentals*. CSLI, Stanford University, CA.
13. Pollard, Carl, and Ivan Sag (1994). *Head-Driven Phrase Structure Grammar*. Centre for the Study of Language and Information, StanfordUniversity, CA.
14. Shieber, Stuart M. (1986). *An Introduction to Unification-Based Approaches to Grammar*, CSLI Lecture Notes, Stanford University, CA.

Fred Popowich is an Associate Professor of Computing Science and an Associate Member of the Department of Linguistics at Simon Fraser University. He received his Ph.D. in Cognitive Science/ Artificial Intelligence from the University of Edinburgh in 1989. His current research interests include the development and processing of unification based grammars, machine translation, natural language interfaces to databases, the structure of the lexicon, the use of inheritance in the lexicon, and the use of lexical resources in natural language processing applications.

The Complexity of Propositional Proofs *

Alasdair Urquhart †

Résumé

LA COMPLEXITÉ DES PREUVES PROPOSITIONNELLES

Le calcul propositionnel classique a, aux yeux des logiciens, la réputation non méritée d'être essentiellement triviale. J'espère convaincre le lecteur qu'il présente au contraire quelques-uns des problèmes les plus difficiles et les plus intéressants de la logique moderne. Cet article fait un survol du domaine et de quelques-unes des techniques avec lesquelles on a pu réduire avec succès la complexité des preuves.

1 Introduction

The classical propositional calculus has an undeserved reputation among logicians as being essentially trivial. I hope to convince the reader that it presents some of the most challenging and intriguing problems in modern logic.

Although the problem of the complexity of propositional proofs is very natural, it has been investigated systematically only since the late 1960s. Interest in the problem arose from two fields connected with computers, automated theorem proving and computational complexity theory. The earliest paper in the subject is a ground-breaking article by Tseitin [25], the published version of a talk given in 1966 at a Leningrad seminar. In the three decades since that talk, substantial progress has been made in determining the relative complexity of proof systems, and in proving strong lower bounds for some restricted proof systems. However, major problems remain to challenge researchers.

The present paper provides a survey of the field, and of some of the techniques that have proved successful in deriving lower bounds on the complexity of proofs. A major area not covered here is the proof theory of bounded arithmetic and its relation to the complexity of propositional proofs. The reader is referred to the book by Buss [1] for background in bounded arithmetic. The recent book by Krajíček [16] also gives a good introduction to bounded arithmetic, as well as covering most of the basic results in complexity of propositional proofs.

2 Proof systems and simulation

The literature of mathematical logic contains a very wide variety of proof systems. To compare their efficiency, we need a general definition of a proof system. In this section, we give such a definition, together with another that formalizes the relation holding between two proof systems when one can simulate the other efficiently. The definitions are adapted from Cook and Reckhow [5].

Let Σ be a finite alphabet; we write Σ^* for the set of all finite strings over Σ . A *language* is defined as a subset of Σ^* , that is, a set of strings over a fixed alphabet Σ . The length of a string x is written as $|x|$.

Definition 2.1 *If Σ_1 and Σ_2 are finite alphabets, a function f from Σ_1^* into Σ_2^* is in \mathcal{L} if it can be computed by a deterministic Turing machine in time bounded by a polynomial in the length of the input.*

The class \mathcal{L} of polynomial-time computable functions is a way of making precise the vague notion of "feasibly computable function".

Definition 2.2 *If $L \subseteq \Sigma^*$, a proof system for L is a function $f : \Sigma_1^* \rightarrow L$ for some alphabet Σ_1 , where $f \in \mathcal{L}$ and f is onto. A proof system f is polynomially bounded if there is a polynomial $p(n)$ such that for all $y \in L$, there is an $x \in \Sigma_1^*$ such that $y = f(x)$ and $|x| \leq p(|y|)$.*

The intention of this definition is that $f(x) = y$ is to hold if x is a proof of y . The crucial property of a proof system as defined above is that, given an alleged proof, there is a feasible method for checking whether or not it really is a proof, and if so, of what it is a proof. A standard axiomatic proof system for the tautologies,

*This paper is a condensed version of a paper [29] of the same title published in the Bulletin of Symbolic Logic. This article is copyrighted by the Association for Symbolic Logic. All rights reserved. This reproduction is by special permission for this publication only. All the detailed proofs in this version of the paper have been omitted as well as sections 7 on Frege Systems, section 8 on Extended Frege Systems and section 9 on Open Problems.

†The author gratefully acknowledges the support of the National Sciences and Engineering Research Council of Canada.

for example, can be brought under the definition by associating the following function f with the proof system \mathcal{F} : if a string of symbols σ is a legitimate proof in \mathcal{F} of a formula A , then let $f(\sigma) = A$; if it is not a proof in \mathcal{F} then let $f(\sigma) = T$, where T is some standard tautology, say $P \vee \neg P$.

Let us recall here some of the basic definitions in computational complexity theory (for details the reader is referred to [11, 14, 18]). A set of strings is in the class \mathcal{P} (\mathcal{NP}) if it is recognized by a deterministic (non-deterministic) Turing machine in time polynomial in the length of the input. A set of strings is in the class $co\text{-}\mathcal{NP}$ if it is the complement of a language in \mathcal{NP} . In more logical terms, a set S of strings is in \mathcal{P} if its characteristic function is in \mathcal{L} , while it is in \mathcal{NP} if the condition $y \in S$ can be expressed in the form $(\exists x)(|x| \leq p(|y|) \wedge R(x, y))$, where p is a polynomial, and R is a polynomial-time computable relation. Thus \mathcal{P} is the polynomial-time analogue of the recursive sets, while \mathcal{NP} corresponds to the recursively enumerable sets. Thus the basic question $\mathcal{P} = ?\mathcal{NP}$ is the polynomial-time analogue of the halting problem.

The importance of our main question for theoretical computer science lies in the following result of Cook and Reckhow [5].

Theorem 2.1 $\mathcal{NP} = co\text{-}\mathcal{NP}$ if and only if there is a polynomially-bounded proof system for the classical tautologies.

This equivalence result underlines the very far-reaching nature of the widely believed conjecture $\mathcal{NP} \neq co\text{-}\mathcal{NP}$. The conjecture implies that even ZFC , together with any true axioms of infinity that are thought desirable (provided that they have a sufficiently simple syntactic form) is not a polynomially-bounded proof system for the classical tautologies (where we take a proof of $TAUT(\ulcorner A \urcorner)$ as a proof of the tautology A).

We can say nothing of interest about the complexity of such powerful proof systems as the above (in effect, the strongest we can imagine). We can, however, order proof systems in terms of complexity, and prove some non-trivial separation results for systems low down in the hierarchy.

Definition 2.3 If $f_1 : \Sigma_1^* \rightarrow L$ and $f_2 : \Sigma_2^* \rightarrow L$ are proof systems for L , then f_2 p -simulates f_1 provided that there is a polynomial-time computable function $g : \Sigma_1^* \rightarrow \Sigma_2^*$ such that $f_2(g(x)) = f_1(x)$ for all x .

Thus g is a feasible translation function that translates proofs in f_1 into proofs in f_2 . We have assumed in the above definition that the language of both proof systems is the same. Reckhow's thesis [19, §5.1.2] contains a more general definition of p -simulation that eliminates this restriction. It is easy to see that the p -simulation relation is reflexive and transitive, and also

that the following theorem can be proved from the definitions.

Theorem 2.2 If a proof system f_2 for L p -simulates a polynomially bounded proof system f_1 , then f_2 is also polynomially bounded.

The intersection of the p -simulation relation and its converse is an equivalence relation; thus we can segregate classes of proof systems into equivalence classes within which the systems are "equally efficient up to a polynomial".

3 A map of proof systems

Since the complexity class \mathcal{P} is closed under complementation, it follows that if $\mathcal{P} = \mathcal{NP}$ then $\mathcal{NP} = co\text{-}\mathcal{NP}$. This suggests that we might attack the problem $\mathcal{P} = ?\mathcal{NP}$ by trying to prove that $\mathcal{NP} \neq co\text{-}\mathcal{NP}$; by Theorem 2.1, this is the same as trying to show that there is no polynomially-bounded proof system for the classical tautologies. This line of research was first suggested in papers by Cook and Reckhow [4, 5]. At the moment, the goal of settling the question $\mathcal{NP} \neq co\text{-}\mathcal{NP}$ seems rather distant. However, progress has been made in classifying the relative complexity of well known proof systems, and in proving lower bounds for restricted systems. An attractive feature of the research programme is that we can hope to approach the goal step by step, developing ideas and techniques for simpler systems first.

The diagram in Figure 1 is a map showing the relative efficiency of various systems. The boxes in the diagram indicate equivalence classes of proof systems under the symmetric closure of the p -simulation relation. Systems below the dotted line have been shown to be not polynomially bounded, while no such lower bounds are known for those that lie above the line. Hence, the dotted line represents the current frontier of research on the main problem. Although systems below the line are no longer candidates for the role of a polynomially bounded proof system, there are still some interesting open problems concerning the relative complexity of such systems. Questions of this sort, although not directly related to such problems as $\mathcal{NP} = ?co\text{-}\mathcal{NP}$, have some relevance to the more practical problem of constructing efficient automatic theorem provers. Although the more powerful systems above the dotted line are the current focus of interest in the complex of questions surrounding the $\mathcal{NP} = ?co\text{-}\mathcal{NP}$ problem, the systems below allow simple and easily mechanized search strategies, and so are still of considerable interest in automated theorem proving.

An arrow from one box to the other in the diagram indicates that any proof system in the first box can p -simulate any system in the second box. In the case of

cut-free Gentzen systems, this simulation must be understood as referring to a particular language on which both systems are based. An arrow with a slash through it indicates that no p-simulation is possible between any two systems in the classes in question. If a simulation is possible in the reverse direction, then we can say that systems in one class are strictly more powerful than systems in the other (up to a polynomial). The diagram shows that all such questions of relative strength have been settled for systems below the dotted line, with the exception of the case of the relative complexity of resolution and cut-free Gentzen systems where connectives other than the biconditional and negation are involved.

The diagram shows only a selection from the wide variety of proof systems that have been considered in the literature of logic, automatic theorem proving and combinatorics. A more detailed diagram, showing a wider selection of proof systems, though not reflecting work after 1976, is to be found in Reckhow [19].

Before proceeding to consider particular proof systems, let us fix our notation. We assume an infinite supply of propositional variables and their negations; a variable or its negation is a *literal*. We say that a variable P and its negation $\sim P$ are *complements* of each other; we write the complement of a literal l as \bar{l} . A finite set of literals is a *clause*; it is to be interpreted as the disjunction of the literals contained in it. A set of clauses is to be interpreted as their conjunction. A clause *mentions* a literal l if either l or \bar{l} is in the clause. The *length* of a clause is the number of literals in it. We shall sometimes write a clause by juxtaposing the literals in it.

An *assignment* is an assignment of truth-values to a set of propositional variables; some variables may remain unset under an assignment. If Σ is a set of clauses, and ϕ an assignment, then we write $\Sigma|\phi$ for the set of clauses that results from Σ by replacing variables by their values under ϕ and making obvious simplifications. That is to say, if a clause in Σ contains a literal made true by ϕ , then it is removed from the set, while if a literal in a clause is falsified by ϕ then it is removed from the clause. The notation $[l := 1]$ denotes the assignment that sets the literal l to 1 and is otherwise undefined, similarly for $[l := 0]$.

It is useful to fix terminology relating to graphs and trees here. A *graph* consists of a finite set of vertices, a finite set of edges and an incidence relation so that every edge is incident with exactly two distinct vertices (the endpoints of the edge). That is to say, the graphs considered here can contain multiple edges, but not loops; a graph is *simple* if it has at most one edge between any two vertices. Trees should be visualized as genealogical

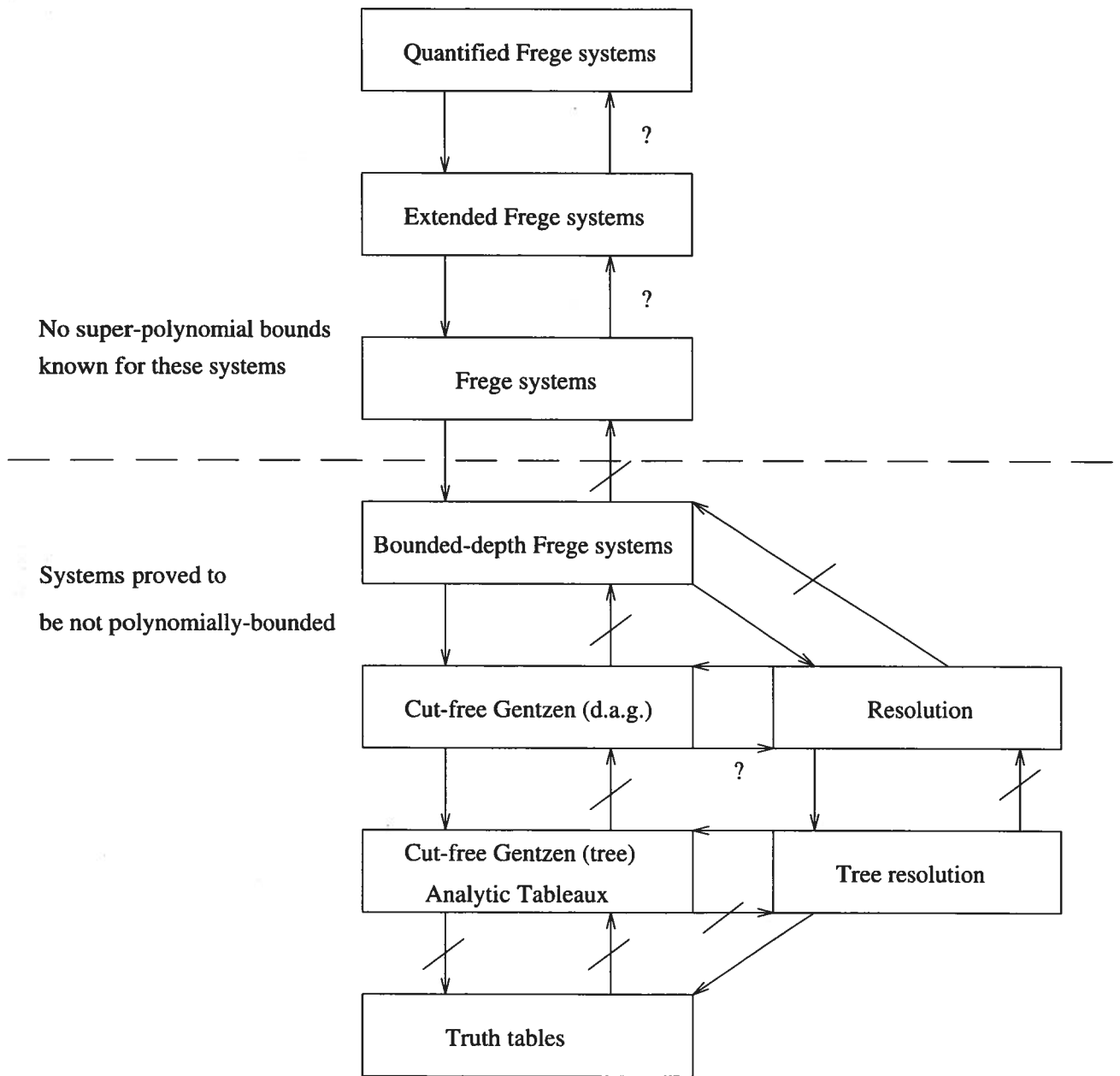
trees, with the root at the top; the nodes immediately below a given node in a tree are its children. The *depth* of a tree T , written $Depth(T)$, is the maximum length of a branch in T .

Derivations in a proof system can be represented either as trees, or as sequences of steps (where a step could be a formula or a sequent). It is normal in the proof-theoretic literature to represent derivations as trees. It is clear, though, that this representation is inefficient, since a step must be repeated every time it is used. If S is a proof system, we denote the corresponding proof system in which derivations are represented as trees by S_{Tree} , reserving the notation S for the system in which derivations are represented as sequences.

4 Analytic Tableaux

The method of analytic tableaux, or truth trees, is employed in many introductory texts; it is given a particularly elegant formulation in Smullyan's monograph [23]. Here we shall only consider the simple form of the method where all formulas are clauses. If Σ is a contradictory set of clauses, then a *tableau* for Σ is a tree in which the interior nodes are associated with clauses from Σ ; if a node is associated with a given clause, then the children of that node are labeled with the literals in the clause. Note that the node associated with a clause is not labeled with that clause itself, so that the root of the tree remains unlabeled. A tableau for Σ is a *refutation* of Σ if every branch in the tableau is closed (i.e. contains a literal and its negation). We define the *size* of a tableau refutation as the number of interior nodes in the tableau (this measure of complexity, omitting the leaves of the tree, is convenient for inductive proofs). If Σ is a set of clauses, then $t(\Sigma)$ is defined to be the minimum size of a tableau refutation of Σ . Because of the simple structure of tableau refutations, it is possible to prove exact lower bounds on their complexity.

A truth table for a formula with n variables, represented as a vector of 0's and 1's, has length 2^n , so that the truth table method is inefficient for large values of n . Of course, we are only considering asymptotic complexity measures here. In practice, the truth table method may be quite efficient for formulas containing a small number of variables, given a reasonably sophisticated implementation. It is easy, however, to find contradictory sets of clauses containing n variables that can be refuted quickly by elementary proof methods, for example the sets A_n containing all the variables P_1, \dots, P_n together with the formula $\sim P_1 \vee \dots \vee \sim P_n$. The set A_n has a tableau refutation of size $n + 1$.



No super-polynomial bounds known for these systems

Systems proved to be not polynomially-bounded

Figure 1: Proof system map

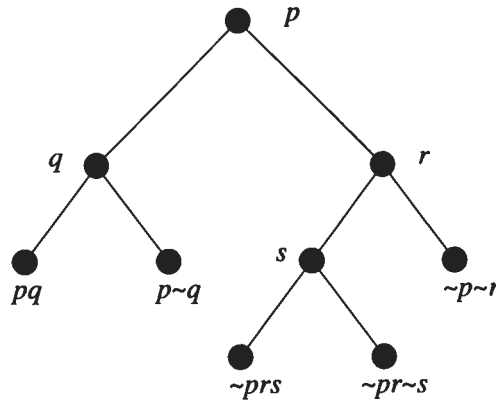


Figure 2: $\Sigma(T) = \{pq, p\sim q, \sim prs, \sim pr\sim s, \sim p\sim r\}$

Somewhat surprisingly, there are cases where truth tables are more efficient than analytic tableaux. This fact was first observed by Marcello D'Agostino, who proved the next result [6]; he noted that truth tables are more efficient than analytic tableaux in the case of the set of all clauses of length n in n variables.

Theorem 4.1 *The analytic tableau proof system cannot p -simulate the method of truth tables.*

Although analytic tableaux work well on simple examples, there are cases where any tableau refutation necessarily contains a great deal of repetition. This is shown by a set of examples due to Cook [3]. Cook's construction associates a set of clauses with a labeled binary tree as follows. Let T be a binary tree in which the interior nodes are labeled with distinct variables. We associate a set of clauses $\Sigma(T)$ with T , in such a way that each branch b in $\Sigma(T)$ has a clause $C_b \in \Sigma(T)$ associated with it. The variables in C_b are those labeling the nodes in b ; if P is such a variable, then P is included in C_b if b branches to the left below the node labeled with P , otherwise C_b contains $\sim P$. Figure 2 shows a simple example.

Cook's clauses are the sets of clauses $\Sigma_n = \Sigma(T_n)$ associated with the complete binary tree T_n of depth n . To include the case where $n = 0$, we take T_0 to consist of a single node, counted as an interior node; the set of clauses $\Sigma(T_0)$ is $\{\Lambda\}$, where Λ is the empty clause.

If one of the variables in $\Sigma(T)$ is set to 0 or 1, then the resulting simplified set of clauses is also of the form $\Sigma(T')$ for some binary tree T' . Let l be a literal in $\Sigma(T)$, and P the variable in l . Define $T \upharpoonright [l := 1]$ to be the tree resulting from T by replacing the subtree whose root is labeled with P by either its immediate left or right subtree, depending on whether l is negated or not. Then it is easy to see that $\Sigma(T) \upharpoonright [l := 1] = \Sigma(T \upharpoonright [l := 1])$.

Theorem 4.2 1. *The clauses Σ_n satisfy the recursion equations: $t(\Sigma_0) = 1$, $t(\Sigma_{n+1}) = t(\Sigma_n) \cdot [t(\Sigma_n) + 1]$;*

2. *There is a constant c , where $0.67618 < c < 0.67819$, such that for all n , $t(\Sigma_n) = \lfloor 2^{c2^n} \rfloor$.*

The preceding theorem is due to Cook; a version of it appeared without proof in [4]. Cook's original unpublished proof [3] contains a gap; the proof found in [29] is joint work of Cook and the present author. Murray and Rosenthal [17] prove a lower bound of $2^{2^{n-1}}$ for the size of analytic tableau refutations of Σ_n .

Theorem 4.2 has some significance for automated theorem proving based on simple tableau methods. The set Σ_6 contains only 64 clauses of length 6, but the minimal tableau refutation for Σ_6 has 10,650,056,950,806 interior nodes. This shows that any practical implementation of the tableau method must incorporate routines to eliminate repetition in tableau construction (for details of such an improved tableau procedure, see Vellino [30]).

5 Resolution

The resolution rule is a simple form of the familiar cut rule. If A_l and $B\bar{l}$ are clauses, then the clause AB may be inferred by the resolution rule, *resolving on* the literal l . A *resolution refutation* of a set of clauses Σ is a derivation of the empty clause from Σ , using the resolution rule. Refutations can be represented as trees or as sequences of clauses; the worst case complexity differs considerably depending on the representation. We shall distinguish between the two by describing the first system as "tree resolution," the second simply as "resolution."

Although resolution operates only on clauses, it can be converted into a general purpose theorem prover for tautologies by employing an efficient method of conversion to conjunctive normal form, first used by Tseitin [25]. Let A be a formula containing various binary connectives such as \rightarrow and \equiv ; associate a literal with each subformula of A so that the literal associated with a subformula $\sim B$ is the complement of the literal associated with B . If the subformula is a propositional variable, then the associated literal is simply the variable itself. We write l_B for the literal associated with the subformula B . If B is a subformula having the form $C \circ D$, where \circ is a binary connective, then $Cl(B)$ is the set of clauses making up the conjunctive normal form of $l_B \equiv (l_C \circ l_D)$. For example, if B has the form $(C \equiv D)$, then $Cl(B)$ is the set of clauses

$$\{ \overline{l_B} \overline{l_C} l_D, \overline{l_B} l_C \overline{l_D}, l_B \overline{l_C} \overline{l_D}, l_B l_C l_D \}.$$

The set of clauses $Def(A)$ is defined as the union of all $Cl(B)$, where B is a compound subformula of A .

If A is a tautology, then the set $Def(A) \cup \{ \overline{l_A} \}$ is contradictory. Thus we define a proof of A in the resolution system to be a proof of Λ from $Def(A) \cup \{ \overline{l_A} \}$. Such a proof of A we shall refer to as a proof by *resolution with limited extension* for the set of connectives (other than \sim) occurring in A . In particular, we shall discuss below the system of resolution with limited extension for the biconditional; we refer to this system as $Res(\equiv)$. Note that the size of the set of clauses $Def(A) \cup \{ \overline{l_A} \}$ is linear in the size of A , whereas the same is not true for the conjunctive normal form of $\sim A$ itself (the conjunctive normal form of $P_1 \equiv P_2 \equiv \dots \equiv P_n$ has size $2^{O(n)}$).

The size of a tree resolution proof is defined as the number of leaves in the tree; if Σ is a contradictory set of clauses, then $tr(\Sigma)$ is defined as the minimal size of a tree resolution refutation of Σ . We shall refer to the clauses at the leaves of a tree resolution derivation as the "input clauses" of the derivation.

Theorem 5.1 1. *Tree resolution p -simulates the method of analytic tableaux.*

2. *The method of analytic tableaux cannot p -simulate tree resolution.*

A sequence of clauses C_1, \dots, C_k in a resolution derivation is an *irregularity* if each C_i , $i < k$, is a premiss for C_{i+1} , and there is a literal l that appears in C_1 and C_k , but does not appear in any clause C_j , where $1 < j < k$. That is to say, the literal l is removed by resolution from C_1 , and is then later re-introduced in a clause depending on C_2 . A derivation is *regular* if it contains no irregularity.

It can be shown that a tree resolution refutation of minimal size is regular. However, corresponding lemma for resolution fails. Andreas Goerdts [12] shows

that there is an infinite sequence of contradictory sets of clauses having polynomial-size resolution refutations for which the size of any regular resolution refutation grows faster than any fixed polynomial. Goerdts's examples are modified versions of the pigeonhole clauses described in Section 7 of [29].

Regular tree resolution is closely related to the method of *semantic trees* introduced by Robinson [20] and Kowalski and Hayes [15]. A semantic tree is a binary tree in which the nodes have assignments associated with them. The assignment associated with the root is empty. If ϕ is an assignment associated with an interior node in the tree then the assignments associated with the children of the node are the assignments ϕ_1 and ϕ_2 extending ϕ with $\phi_1(P) = 0$ and $\phi_2(P) = 1$, where P is a variable not in the domain of ϕ . A semantic tree T is a *refutation* of a set of clauses Σ if the variables assigned values in T all belong to Σ and each of the assignments at the leaves of T falsify a clause in Σ .

We can rewrite a regular tree resolution refutation of a set of clauses as a semantic tree by the following technique. First, associate the empty assignment with the root. Second, if $A \vee B$ is a clause in the tree derived by resolution from AVP and $BV\sim P$, and ϕ is associated with the conclusion of the inference, then we associate with the premisses the extensions of ϕ obtained by setting P to 0 and 1 respectively. Conversely, a semantic tree refutation of minimal size can be converted into a resolution refutation by associating with a leaf a clause falsified at that leaf, and then performing resolutions by resolving on the literals labeling the edges.

Regular refutations of a special kind are produced by the *Davis-Putnam procedure*. Given a set of Σ of input clauses, this procedure involves choosing a variable and then forming all possible non-tautologous resolvents from Σ that result from eliminating the chosen variable. This procedure is repeated until the empty clause is produced or no more resolvents can be formed (in which case the input set must be satisfiable). Clearly the refutation produced depends uniquely on the order of elimination adopted. The name of the procedure derives from a well known paper by Davis and Putnam on automated theorem proving [8].

The phrase "Davis-Putnam procedure" is unfortunately ambiguous, since in the literature of automated theorem proving, it refers to a decision procedure for satisfiability involving the recursive construction of a semantic tree. The confusion stems from the fact that during the implementation of the algorithm described in [8], Davis, Logemann and Loveland [7] replaced the original method by this second one, mainly for reasons of space efficiency. In the present article, the phrase "Davis-Putnam procedure" refers to the restricted version of the resolution proof procedure where the refutations are produced by the first method described above.

In the remainder of this section, the lower bounds proved for various forms of resolution are given for the graph-based examples introduced by Tseitin [25]. This paper of Tseitin is a landmark as the first to give non-trivial lower bounds for propositional proofs; although it pre-dates the first papers on \mathcal{NP} -completeness, the distinction between polynomial and exponential growth of proofs is already clear in it.

If G is a graph, then a *labeling* G' of G is an assignment of literals to the edges of G , so that distinct edges are assigned literals that are distinct and not complements of each other, together with an assignment $\text{Charge}(x) \in \{0, 1\}$ to each of the vertices x in G . If G' is a labeled graph, and x a vertex in G' , and l_1, \dots, l_k the literals labeling the edges attached to x , then $\text{Clauses}(x)$ is the set of clauses equivalent to the conjunctive normal form of the modulo 2 equation $l_1 \oplus \dots \oplus l_k = \text{Charge}(x)$. That is to say, a clause C in $\text{Clauses}(x)$ contains the literals l_1, \dots, l_k , and the parity of the number of complemented literals in $\{l_1, \dots, l_k\}$ in C is opposite to that of $\text{Charge}(x)$. The set of clauses $\text{Clauses}(G')$ is the union of all the sets $\text{Clauses}(x)$, for x a vertex in G . Let us write $\text{Charge}(G')$ for the sum modulo 2 of the charges on the vertices of G' ; a labeling G' of G is *even* or *odd* depending on whether $\text{Charge}(G')$ is 0 or 1.

For the remainder of this section, we assume that G' is a graph with an odd labeling; we identify an edge with the literal labeling it. The proof of the preceding lemma shows that any two sets of clauses associated with an odd labeling of a connected graph G are logically isomorphic, so we shall sometimes write $\text{Clauses}(G)$ to represent any such set.

Let G' be a labeled graph, and l an edge in G' . Define $G' \upharpoonright [l := 0]$ to be the labeled graph resulting from G' by deleting l , and $G' \upharpoonright [l := 1]$ the labeled graph resulting from G' by deleting l and complementing the charges on the vertices incident with l .

Regular resolution refutations of sets of clauses based on graphs can be visualized in terms of joining together connected subgraphs, and we can compute the complexity function $tr(\text{Clauses}(G))$ directly from the graph G . Thus, the problem of proving lower bounds for tree resolution can be reduced to that of finding graphs that require large deletion trees. The next result is due in its essentials to Tseitin [25].

Theorem 5.2 *Tree resolution cannot p -simulate the Davis-Putnam procedure.*

By considering a different sequence of graphs, we can find a family of clauses for which the smallest resolution refutations are exponentially big. The basic idea of the lower bound proof given below, from Urquhart [26], is due to Armin Haken [13], who introduced an ingenious “bottle-neck” counting argument to prove the corresponding result for the pigeonhole clauses. The

sequence of graphs G_m in the next theorem is chosen in such a way that $\text{Clauses}(G_m)$ contains at most $128n$ clauses of length 7, where $n = m^2$.

Theorem 5.3 *There is a constant $c > 1$ such that for sufficiently large m any resolution refutation of $\text{Clauses}(G_m)$ contains c^n distinct clauses.*

Chvátal and Szemerédi, in a far reaching generalization of the preceding theorem, proved a lower bound for sets of randomly chosen clauses, by showing that sets of random clauses satisfy appropriately generalized forms of these properties. Define the *random family of m clauses of length k over n variables* to consist of a random sample of size m chosen with replacement from the set of all clauses of length k with variables chosen from a set of n variables. Chvátal and Szemerédi [2] prove the following result.

Theorem 5.4 *For every choice of positive integers n, c, k such that $k \geq 3$ and $c2^{-k} \geq 0.7$, there is a positive number ϵ such that, with probability tending to one as n tends to infinity, the random family of cn clauses of size k over n variables is unsatisfiable and its resolution complexity is at least $(1 + \epsilon)^n$.*

6 Cut-free Gentzen systems

Cut-free Gentzen systems have proved popular in work on automated deduction, since they allow simple search strategies in constructing derivations. In the present section, we consider a sequent calculus G based on the biconditional as the only connective. A sequent has the form $\Gamma \vdash \Delta$, where Γ and Δ are sequences of formulas. The axioms of G are sequents of the form $A \vdash A$. The rules of inference of G are as follows:

$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}{\Gamma_1, B, A, \Gamma_2 \vdash \Delta} \quad \frac{\Gamma \vdash \Delta_1, A, B, \Delta_2}{\Gamma \vdash \Delta_1, B, A, \Delta_2} \quad (\text{Permutation})$$

$$\frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta} \quad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A} \quad (\text{Contraction})$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, \Theta \vdash \Delta, \Xi} \quad (\text{Thinning})$$

$$\frac{\Gamma, A, B \vdash \Delta \quad \Gamma \vdash \Delta, A, B}{\Gamma, (A \equiv B) \vdash \Delta} \quad (\equiv \vdash)$$

$$\frac{\Gamma, A \vdash \Delta, B \quad \Gamma, B \vdash \Delta, A}{\Gamma \vdash \Delta, (A \equiv B)} \quad (\vdash \equiv)$$

An alternative formulation of G is possible in which the axioms are sequents of the form $\Gamma, A \vdash A, \Delta$, and the thinning rule is omitted. We denote this alternative formulation by G' . It is the version adopted by Smullyan [23, pp. 105-106], and is usually employed in automatic theorem provers; the system of Wang [31] is

of this type. The Leningrad group headed by Shanin in their work on computer search for natural logical proofs [21] used a formulation of the second type for the proof search, but then transformed the resulting derivations into simplified derivations in a system of the first type by a pruning procedure.

It is natural to use a system of the second type in a computer search, because if the usual 'bottom-up' search procedure is employed, the thinning rule can (when employed in reverse) result in potentially useful information being discarded. However, as we show below, the two formulations are quite distinct from the point of view of worst case complexity. There are certain sequents for which short proofs can be found only by employing the thinning rule.

Derivations in G have the *subformula property*, that is, any formula occurring in the derivation must occur as a subformula in the conclusion of the derivation. In fact, an analysis of derivations in G shows that *occurrences* of formulas in the derivation can be identified with *occurrences* of formulas in the conclusion. This can be seen by tracing occurrences of formulas step by step up the derivation from the conclusion. Thus in an application of $(\vdash\equiv)$, for example, the displayed occurrences of A in the premisses are to be identified with the displayed occurrence of A in the conclusion of the inference. Similarly, in an application of Contraction, both occurrences of the displayed formula A in the premiss are to be identified with the occurrence of A in the conclusion. This identification of occurrences will be used subsequently to prove lower bounds for the proof systems.

The Cut rule

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma \vdash \Delta, A}{\Gamma \vdash \Delta} \text{ (Cut)}$$

is not necessary for completeness, but in some cases results in much shorter derivations. The formula A in the Cut rule is said to be the *cut formula*. The subformula property fails for derivations in the system $G + \text{Cut}$ that results by adding the Cut rule to G . However, the property is preserved if we restrict the Cut rule appropriately. We shall say that a derivation of a sequent $\Gamma \vdash \Delta$ is a derivation in G *with the analytic Cut rule* if the derivation belongs to $G + \text{Cut}$, and all cut formulas are subformulas of formulas in the conclusion $\Gamma \vdash \Delta$.

We shall now prove some results from [28] that settle the relative complexity of resolution and cut-free Gentzen systems, at least for the case of tautologies involving the biconditional. As in the case of tree resolution, we define the complexity of a derivation in G_{Tree}

to be the number of leaves in the derivation (that is, the number of occurrences of axioms). The simulation in the following theorem is due to Tseitin [25].

Theorem 6.1 *The system $Res(\equiv)_{Tree}$ p -simulates G_{Tree} .*

We now define the sequence of biconditional tautologies that form the basis of the lower bounds in this section. For any $n > 0$, let U_n be the formula

$$P_n \equiv P_{n-1} \equiv \dots \equiv P_1 \equiv P_n \equiv P_{n-1} \equiv \dots \equiv P_1$$

where we are omitting parentheses according to the convention of association to the right; for example, $A \equiv B \equiv C$ abbreviates $(A \equiv (B \equiv C))$. All the variables in U_n occur exactly twice, so that U_n is a tautology. To distinguish between two occurrences of the same variable P_k , we shall write the first occurrence as P_k^1 , the second occurrence as P_k^2 . The subformula of U_n beginning with the subformula occurrence P_k^i will be denoted by U_k^i . Thus U_k^2 contains k occurrences of variables, while U_k^1 contains $n + k$ occurrences: in particular, $U_n^1 = U_n$.

If $\Gamma \vdash \Delta$ is a sequent, we use the term *O-assignment* to refer to an assignment of truth-values $\{0, 1\}$ to the occurrences of the variables in $\Gamma \vdash \Delta$. An O-assignment is extended to all the occurrences of subformulas in the sequent by the usual truth table method. The entire sequent takes the value 0 under an O-assignment if all occurrences of formulas in Γ take the value 1, and all the occurrences of formulas in Δ the value 0. It is essential to the notion of O-assignment that distinct truth values can be assigned to different occurrences of the same variable. In particular, by choosing an appropriate O-assignment, it is possible to falsify a tautological sequent. If D is a cut-free derivation of a sequent $\Gamma \vdash \Delta$, then any O-assignment for $\Gamma \vdash \Delta$ can be extended to all the sequents in D ; this is possible because of the identification noted earlier between occurrences of formulas in the conclusion and occurrences of formulas in D . A given occurrence of a subformula in the conclusion can correspond to multiple occurrences in a sequent earlier in the derivation; the form of the rules, however, guarantees that all of these occurrences have the same value as the occurrence in the conclusion. The notion of O-assignment was used earlier in a somewhat different form in [27] to prove an exponential lower bound for cut-free Gentzen systems. An exponential lower bound for the tree version of a cut-free Gentzen system was proved earlier by Statman [24].

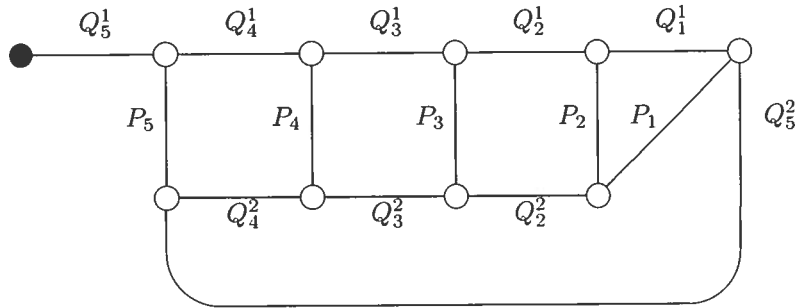


Figure 3: The labeled graph G_5

The formula U_n has 2^{2n} O-assignments associated with it. We are interested only in certain of these. We shall call an O-assignment to U_n *critical* if there is exactly one variable in U_n whose occurrences in U_n are assigned different values. If this variable is P_k , then we say that the O-assignment in question is *k-critical*. All critical O-assignments falsify the formula U_n ; a critical O-assignment is uniquely determined by k , and the values the O-assignment gives to the occurrences P_i^1 , for $1 \leq i \leq n$, so that there are $n \cdot 2^n$ distinct critical O-assignments for the sequent U_n . It is possible to show that distinct O-assignments correspond to distinct branches in a proof of U_n in the system G_{Tree} ; this gives us the next theorem.

Theorem 6.2 *The minimal complexity of a derivation of U_n in the system G_{Tree} is $n \cdot 2^n$.*

We now show that there are relatively short proofs of U_n in the resolution system. To describe the proofs, it is useful to give a graphical representation of these tautologies. The sets of clauses derived from the sequence of formulas U_n will be represented in the form $Clauses(G_n)$, for a sequence of graphs G_n .

The graph G_n associated with the formula U_n is a planar graph that we describe by giving the coordinates of its nodes. G_n has as its nodes the set of points $\{(i, 1) : 0 \leq i \leq n\} \cup \{(i, 0) : 1 \leq i \leq n-1\}$. The following nodes are joined in G_n : $(i, 1)$ to $(i+1, 1)$, $(i, 0)$ to both $(i, 1)$ and $(i+1, 0)$, $(n, 1)$ to both $(n-1, 0)$ and $(1, 0)$. The graph may be described as a ladder with a few extra attachments. The labels attached to G_n are as follows. The vertical lines, and the line joining $(n, 1)$ to $(n-1, 0)$ are labeled with the variables P_n to P_1 from left to right. The horizontal lines joining the points with y co-ordinate 1 are labeled with the variables Q_n^1 to Q_1^1 from left to right; the horizontal lines joining points with y co-ordinate 0 are labeled with the variables Q_{n-1}^2 to Q_2^2 from left to right. The line joining $(n, 1)$ to $(1, 0)$ is labeled with the variable Q_n^2 . The

node $(0, 1)$ is labeled with 0; all other nodes are labeled 1. The accompanying figure shows the labeled graph corresponding to U_5 . A node is shown filled in only if it is labeled with 0.

The set of clauses $Def(U_n) \cup \{\sim Q_n^1\}$, where the variable Q_k^i is correlated with the subformula U_k^i , is identical with $Clauses(G_n)$. The graphs G_n are similar to examples used by Galil ([10, Fig. 3.2.4]) to show that the Davis-Putnam procedure is very sensitive to the order of elimination adopted in forming resolvents; with one order of elimination, ladder-like graphs result in exponential-size refutations, while a different order gives rise to linear-size refutations.

Theorem 6.3 *The tautologies U_n have proofs in $Res(\equiv)_{Tree}$ of complexity $O(n^2)$.*

Corollary 6.1 *The system G_{Tree} cannot p -simulate $Res(\equiv)_{Tree}$.*

In contrast to the foregoing results, if derivations are presented in linear form, then resolution and cut-free Gentzen systems are equally powerful systems (up to a polynomial) when pure biconditional tautologies are considered.

Theorem 6.4 *Each of the following systems can p -simulate any of the others: G , $G + \text{analytic Cut}$, $Res(\equiv)$.*

This somewhat unexpected simulation result depends on the special features of the inference rules for \equiv in G . It extends easily to include negation, but does not appear to extend to conjunction and disjunction. Whether the simulation result holds when the cut-free Gentzen system includes these connectives is open.

We now sketch a result mentioned earlier, that the addition of the Thinning rule results in an exponential shortening of derivations in some cases.

Theorem 6.5 *A derivation of U_n in the system G' must contain at least $n \cdot 2^n$ distinct sequents.*

Techniques similar to those used in the lower bound for resolution can be used to prove exponential lower bounds for cut-free Gentzen systems. The following result is proved in Urquhart [27].

Theorem 6.6 *There is a sequence F_n of biconditional tautologies, where each formula has length $O(n^2)$, but the shortest proof of F_n in G contains at least $2^{n/16}$ distinct sequents.*

This result can be improved to a lower bound exponential in the size of a family of biconditional tautologies based on expander graphs by adapting the proof of Theorem 5.3.

We conclude this section with the observation that a cut-free Gentzen system for a given set of connectives and the corresponding analytic tableau system are p-equivalent. This can be seen most easily by using the form of Gentzen system where the thinning rule is omitted. Then (as Dowd [9] first observed) there is a straightforward and efficient translation procedure between the two systems; the details are to be found in Smullyan's book [23, Ch. XI]. The proof of equivalence is completed by showing that (in contrast to the case where proofs are represented as sequences) the system without thinning can simulate the system with the thinning rule in an efficient way.


7 Acknowledgments

The author wishes to thank Paul Beame, Andreas Blass, Samuel R. Buss, Stephen A. Cook, Jan Krajíček, Toniann Pitassi, Richard Shore, Charles Silver and the referee for helpful comments, and for pointing out errors and omissions in earlier versions of this survey.

References

- [1] Samuel R. Buss. *Bounded Arithmetic*. Bibliopolis, Naples, 1986.
- [2] Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the Association for Computing Machinery*, 35:759–768, 1988.
- [3] Stephen A. Cook. An exponential example for analytic tableaux. Manuscript, 1973.
- [4] Stephen A. Cook and Robert A. Reckhow. On the lengths of proofs in the propositional calculus. In *Proceedings of the Sixth Annual ACM Symposium on the Theory of Computing*, 1974. See also corrections for above in *SIGACT News*, Vol. 6 (1974), pp. 15-22.

- [5] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44:36–50, 1979.
- [6] Marcello D'Agostino. Are tableaux an improvement on truth-tables? *Journal of Logic, Language and Information*, 1:235–252, 1992.
- [7] Martin Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the Association for Computing Machinery*, 5:394–397, 1962. Reprinted in [22], Vol. 1, pp. 267-270.
- [8] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7:201–215, 1960. Reprinted in [22], Vol. 1, pp. 125-139.
- [9] Martin Dowd. Model-theoretic aspects of $\mathcal{P} \neq \mathcal{NP}$. Unpublished MS, 1985.
- [10] Zvi Galil. On the complexity of regular resolution and the Davis-Putnam procedure. *Theoretical Computer Science*, 4:23–46, 1977.
- [11] Michael R. Garey and David S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-completeness*. W.H. Freeman, 1979.
- [12] Andreas Goerdt. Comparing the complexity of regular and unrestricted resolution. In *Proceedings of the 14th German Workshop on A.I.* Informatik Fachberichte 251, 1990.
- [13] Armin Haken. The intractability of resolution. *Theoretical Computer Science*. 39:297–308, 1985.
- [14] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
- [15] R. Kowalski and P. J. Hayes. Semantic trees in automatic theorem-proving. In Meltzer and Michie, editors, *Machine Intelligence Vol. 4*, pages 87–101. Edinburgh U. Press, Edinburgh, 1969.
- [16] Jan Krajíček. *Bounded Arithmetic, Propositional Logic and Complexity Theory*. Cambridge University Press, 1996.
- [17] Neil V. Murray and Erik Rosenthal. On the computational intractability of analytic tableau methods. *Bulletin of the IGPL*, Volume 2, Number 2:205–228, September 1994.
- [18] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

- [19] Robert Reckhow. *On the lengths of proofs in the propositional calculus*. PhD thesis, University of Toronto, 1976.
- [20] J. A. Robinson. The generalized resolution principle. In Dale and Michie, editors, *Machine Intelligence, Vol. 3*, pages 77–94. American Elsevier, New York, 1968. Reprinted in [22], Vol. 2, pp. 135–151.
- [21] N.A. Shanin, G.V. Davydov, S. Y. Maslov, G.E. Mints, V.P. Orevkov, and A.O. Slisenko. *An algorithm for a machine search of a natural logical deduction in a propositional calculus*. Izdat. Nauka, Moscow, 1965. Reprinted in [22].
- [22] Jörg Siekmann and Graham Wrightson, editors. *Automation of Reasoning*. Springer-Verlag, New York, 1983.
- [23] Raymond M. Smullyan. *First-order Logic*. Springer-Verlag, New York, 1968. Reprinted by Dover, New York, 1995.
- [24] Richard Statman. Bounds for proof-search and speed-up in the predicate calculus. *Annals of mathematical logic*, 15:225–287, 1978.
- [25] G.S. Tseitin. On the complexity of derivation in propositional calculus. In A. O. Slisenko, editor, *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pages 115–125. Consultants Bureau, New York, 1970. Reprinted in [22], Vol. 2, pp. 466–483.
- [26] Alasdair Urquhart. Hard examples for resolution. *Journal of the Association for Computing Machinery*, 34:209–219, 1987.
- [27] Alasdair Urquhart. The complexity of Gentzen systems for propositional logic. *Theoretical Computer Science*, 66:87–97, 1989.
- [28] Alasdair Urquhart. The relative complexity of resolution and cut-free Gentzen systems. *Annals of mathematics and artificial intelligence*, 6:157–168, 1992.
- [29] Alasdair Urquhart. The complexity of propositional proofs. *The Bulletin of Symbolic Logic*, 1:425–467, 1995.
- [30] André Vellino. The relative complexity of sl-resolution and analytic tableau. *Studia Logica*, 52:323–337, 1993.
- [31] Hao Wang. Towards mechanical mathematics. *IBM Journal for Research and Development*, 4:2–22, 1960. Reprinted in [22], Vol. 1, pp. 244–264. 

Alasdair Urquhart (born 1945 Scotland) was educated at the Universities of Edinburgh (MA, 1967) and Pittsburgh (Ph.D., 1974). His research interests are in the areas of non-classical logics, history of logic, and computational complexity theory. He is an editor of the series Lecture Notes in Logic (Springer) and Trends in Logic (Kluwer), and has served as a member of the Council of the Association for Symbolic Logic.

Non-Monotonic Logic is Impossible

Charles Morgan

Résumé

Il arrive quelquefois que l'on tire une conclusion d'un ensemble de prémisses, et qu'on la rejette plus tard à la suite d'information additionnelle. C'est ce que l'on appelle la logique non-monotone, et la plus grande partie de notre raisonnement de sens commun semble être de ce type. À l'opposé, les systèmes formels développés par les logiciens classiques sont monotones; si une conclusion découle légitimement d'un ensemble de prémisses, la même conclusion découlera légitimement de ces prémisses, quelles que soient les autres prémisses qu'on leur a ajoutées. Beaucoup d'efforts de recherche ont été consacrés à tenter de formuler la logique non-monotone. Nous présentons dans cet article une preuve formelle que la logique non-monotone est impossible. Nous discutons ensuite quelques conséquences de cette preuve pour la recherche dans ce domaine.

Abstract

Sometimes we draw a conclusion from a set of premises, but when we subsequently get additional information we reject that conclusion. Arguments such that it is possible we may reject the conclusion when given additional premises are said to be non-monotonic. Much of our common sense reasoning seems to be of this sort. By contrast, formal systems developed by classical logicians are monotonic; that is, if a conclusion follows legitimately from a set of premises, then the same conclusion legitimately follows from those premises augmented with any other premises at all. Much research has been devoted to attempts to formulate non-monotonic logics. After some motivational material, we give a formal proof that non-monotonic logic is impossible. We then discuss some consequences for research in this area.

Introduction

Most of our ordinary, commonsense reasoning about everyday matters is tentative at best, our conclusions being subject to revision in the light of new information. It will be instructive to consider a slightly revised version of a now classic example.

- A: Tilly is an animal.
B: So Tilly cannot fly.
A: But Tilly is a bird.
B: So Tilly can fly.
A: Tilly is a penguin.
B: So Tilly cannot fly.

In this sequence, B's conclusion concerning the ability of Tilly to fly flips back and forth as A supplies more specific information about Tilly. Sometimes the very names of individuals give hints about their properties that can influence the conclusions we draw. Using "Tweety" instead of "Tilly" would make us more likely to attribute flying ability on the basis of knowing only animality, while using "Rover" would make us less likely to make such an attribution. Simply put, "Tweety" is a name we typically associate with a small (usually cartoon character) bird which can fly, while "Rover" is a name we typically associate with some large (usually fictional) dog which cannot fly.

At least part of the goal of artificial intelligence (AI) research is to design systems that can reason in much the same way that humans do. Historically speaking, one of the primary tools of AI research has been formal logic. But formal logics of the sort usually encountered do not seem well suited to the analysis of commonsense reasoning of the kind illustrated by our example. In the phraseology common in the literature, the usual formal logics are "monotonic". Let us use the notation $\Gamma \vdash_L A$ to indicate that using the resources of logic L (classical, many-valued, modal, whatever), sentence A may be legitimately inferred from the set of assumptions Γ . To say that logic L is monotonic means that: if $\Gamma \vdash_L A$, then $\Gamma \cup \Delta \vdash_L A$, for all assumption sets Δ . In ordinary parlance, if it is legitimate to conclude A from the assumptions in Γ , then no additional assumptions added to Γ can make it illegitimate to conclude A .

Ordinary reasoning seems to be undeniably non-monotonic in character. There is nothing peculiar about the example with which we began. Such examples may be enumerated *ad infinitum* from virtually every area of our lives. We constantly revise our conclusions in light of additional information. Indeed, in most contexts one who refuses to revise beliefs in the light of new information can justly be charged with being irrational. If our usual formal logics cannot capture this non-monotonic aspect of our reasoning, then it seems obvious that we need to develop a new logic that can, i.e., we need a non-monotonic logic. In recent years a great deal of research in AI has been devoted to the task of developing non-monotonic logics.

As reasonable a task as it may seem in light of our discussion, the attempt to develop a non-monotonic logic is fundamentally misguided. To put it bluntly, non-monotonic logic is impossible. In the following sections, we will give an intuitive motivation for a simple formal framework, and then within that framework we will give a simple proof of this rather shocking sounding claim; finally we will use the proof to draw some instructive lessons.

Intuitive Motivation

Much of the activity of biological organisms revolves around three important goals: (1) energy acquisition, (2) reproduction, and (3) death avoidance. For simple organisms, like a paramecium, food is plentiful and randomly distributed, enemies are randomly distributed, and reproductive activities are randomly distributed. In such cases, a purely random strategy requiring virtually no computation is very effective. More complicated organisms are able to exploit more complex environments in which food is scarce and not randomly distributed, enemies are often plentiful and not randomly distributed, and reproductive opportunities are rare and not randomly distributed. In order to be able to survive and thrive in a highly non-random environment, an organism must develop strategies for anticipating important aspects of that environment.

In order to be useful, predictive abilities must satisfy a number of criteria. First, predictions must be made sufficiently in advance of an event to allow the organism to act on the prediction. For example, if the predictions of a predator concerning the location of its prey take so long to make that the prey has left the predicted location, then the predator is going to get very hungry. Second, predictions must concern matters for which the organism possesses a behavioral repertoire. For example, it will do a lion no good at all to recognize that if it could fly then it would be better able to locate its prey. Thirdly, the time and energy costs of the prediction must not **on average** exceed the utility of the prediction. Predictions do not have to be correct or lead to success every single time. When rushing its prey, if a lion consistently badly misjudges the path that it and the prey will take, then it will be unsuccessful in killing the prey, and will simply use up more energy. On the other hand, the lion need not be accurate to within an angstrom; even if the lion's lunge is out by a few centimetres, it will still be successful.

As the number of behavioral strategies and the predictive capacity of an organism increase, the greater the diversity of environments it can exploit. But an increase in behavioral and predictive capacity yields an increase in the logical state space complexity, and hence the physical computational machinery must be more complex. However, the more complex the computational machinery, the greater are its energy requirements. Devices which can store information in the environment and systematically recover that information as needed will **in general** have an advantage over devices not so endowed, for at least two reasons: first, such devices will require fewer internal states for memory; and second, such devices often have superior computational ability. For example, we may think of a Turing machine as just a finite state machine "brain" in control of a read/write head, storing information in and recovering information from its environment. There are many functions that are Turing computable but not finite state machine computable.

But even if we consider just finite input/output sequences, it will often be possible to design a smaller Turing machine to carry out the required computations than to devise a massive finite state machine implementing the look-up table.

So, it follows that organisms with the ability to exploit very diverse environments need computationally efficient languages that can be used to model the environment and to predict future events. In general, time and energy efficiency prohibit totally accurate models and totally accurate predictions. The important conclusion of these rather mundane observations is just this: the human use of natural languages is based, at least in part, on their utility for modelling and predicting aspects of the environment under conditions of uncertainty.

Formal Preliminaries

Part of the task of logic is to identify the logical components of natural languages and clearly distinguish them from the non-logical components. Logicians then attempt to construct formal languages reflecting the essential logical components; they then use such constructs to formulate inferential regularities. There is often much debate about these logical components and how they operate, and such debates lead to the formulation of alternative logics, intuitionism and the many flavours of modal logic being just a few examples.

We do not wish to beg any important logical questions here, so we will not concentrate on any particular formal language or class of such languages. In order to be completely general, we simply assume that our formal language consists of some non-empty set of sentences:

$$\mathcal{L} = \{E_1, E_2, \dots\}$$

We use the language to model our world and to make inferences about it. Our model of the world, our set of beliefs about the world, changes from time to time; but it is not the evolution of belief that is of interest at the moment. An instantaneous snapshot of our beliefs constitutes a model under uncertainty of the world. What are the properties of such a belief structure?

Perhaps the most rigorous theory of belief structures expressed in formal languages is classical probability theory. But there are many questionable aspects of probability theory. For example, it is unreasonable to expect real people to be able to linearly order all possible beliefs, much less be able to assign precise numerical values to them. Further, the principles of classical probability are generally expressed in terms of the connectives of classical logic, e.g., conjunction, disjunction, and negation. What are we to say about languages which are lacking these specific logical particles or have others?

Again, to avoid begging any important questions, we will make minimal assumptions about belief structures. We will **not** assume any particular set of "belief values." We will **not**

assume beliefs can be linearly ordered, nor even that all beliefs are comparable. We will only assume that a belief structure is some quasi-ordering relation on sets of sentences of the language. Formally, we define a simple **belief structure** to be a binary relation LE , relating sets of sentences, that is both reflexive and transitive:

B.0 Field definition: $LE \subseteq \mathcal{P}(\mathcal{L}) \times \mathcal{P}(\mathcal{L})$

B.1 Reflexivity: $\Gamma LE \Gamma$

B.2 Transitivity: $\Gamma LE \Gamma$ and $\Gamma' LE \Gamma' \Rightarrow \Gamma LE \Gamma'$

Intuitively, we think of a set of sentences as a theory about the world in the ordinary sense; which is to say that sets are treated in a conjunctive way. For example, we think of the set $\{A, B\}$ as asserting both that A and that B . However, we do not assume that there is any conjunctive connective in the formal language, and we allow sets to be infinitely big. We may think of a simple belief structure as representing something like “degrees of belief” in various sets of sentences. “I am at least as sure of this as I am of that.” So a simple belief structure can be thought of as a very general model of the world under uncertainty.

Our intuitive characterization of belief structures leads to another constraint in addition to reflexivity and transitivity. We call the constraint the “subset principle”, and simply put, it says that if Γ makes no more claims about the universe than Δ , then Γ is at least as likely as Δ . The principle is expressed formally as follows:

B.3 Subset principle: If $\Gamma \subseteq \Delta$, then $\Delta LE \Gamma$.

This principle is easily justified on relative frequency grounds. If we regard a set of sentences as a theory about the universe, then each set of sentences will be compatible with some possible universes (universe designs, if you prefer) and incompatible with others. If theory Γ is a subset of theory Δ , then the set of alternative universes (universe designs) compatible with Γ will be a superset of those compatible with Δ .

We now turn our attention to the notion of logic. As observed earlier, one of the roles of logic is to formulate inferential regularities. For example, in standard classical logic, we are told that from $A \supset B$ and A , it is legitimate to conclude B (where “ \supset ” is the usual material conditional). So in terms of belief structures, classical logic would tell us that all belief structures must satisfy the following:

$$\{A \supset B, A\} LE \{B\}$$

Again, we do not wish to prejudge any issues concerning the nature of permissible logics. We will adopt the very general view that a logic is any prescription for what is to count as a *rational* belief structure. From this point of

view, a logic L is just any set of belief structures:

$$L = \{LE_1, LE_2, \dots\}$$

Different logics will pick out different belief structures. For example, classical logic will correspond to a different set of belief structures than intuitionistic logic. From the point of view of any particular logic L , all and only those belief structures in L are rational.

Finally, we turn our attention to the notion of logical entailment. Logical entailment is a relation between a set of premises and a conclusion; we write $\Gamma \vdash X$. Typically logical entailment relations are based on iterated applications of formalized inference rules, and the inference rules are formulated in terms of the logical particles of the language. Once again, we do not wish to beg any important questions here, so we will make only minimal assumptions about the entailment relation.

Obviously the entailment relation for different logics will be different. But once a logic is chosen, the entailment relation is fixed. Whatever its other characteristics, **logical entailment** is not a matter of psychology; logicians are not attempting to determine psychological associations individuals may make between sentences. So logical entailment does not depend on the characteristics of any single belief structure. Rather, logical entailment under logic L must be based on some universal properties of all rational belief structures permitted by logic L .

This dependence on only universal properties is expressed by two relationships between the set of rational belief structures and the entailment relation. These relationships between the set of rational belief structures and the entailment relation are usually spoken of as properties of logical entailment.

The first property is called soundness: If Γ logically entails A , then it is always rational to have at least as much confidence in A as in Γ . Briefly we sometimes express this idea by the claim that logical entailment should not lead us astray. Formally, we can express the soundness property as follows:

(1) Soundness: If $\Gamma \vdash A$, then $\Gamma LE \{A\}$ for all rational belief structures $LE \in L$.

The second property is called completeness: If no matter what our belief structure, it is always rational to have at least as much confidence in A as in Γ , then Γ logically entails A . Briefly we sometimes express this idea by saying that we want our entailment relation to be strong enough to capture any argument which is sanctioned by every rational belief structure. Formally, we can express the completeness property as follows:

(2) Completeness: If $\Gamma LE \{A\}$ for all rational belief structures $LE \in L$, then $\Gamma \vdash A$.

In formal terms familiar to most logicians, the entailment relation corresponds to the formal proof theory of the logic, while the belief structures correspond to the formal semantics. But nothing of any import turns on these labels. What is important is that they are well rooted in our intuitive understanding of the reasoning process. With this simple framework, we are now in a position to consider the possibility of developing a non-monotonic logic.

Non-Monotonic Logic Is Impossible

To this point, we have been extremely general. We have imposed no restrictions on our formal language. It could contain any logical particles whatever, or perhaps even none at all. Our characterization of belief structures was extremely general, requiring neither linear ordering, nor even pairwise comparability. Our only requirements were those of quasi-ordering and the most minimal adherence to relative frequency concerns. We have imposed no restrictions on the kinds of logic under consideration, nor on the types of inference rules (if any) that are sanctioned by the logic. We required only that the logic and rational belief structures should be related in such a way that (1) logical entailment should never lead us astray by violating the ordering imposed by any rational belief structure, and (2) logical entailment should be strong enough to capture any argument sanctioned by all rational belief structures. In spite of our generality, it is now possible to prove that logical entailment is monotonic.

Theorem: Logical entailment is monotonic;
that is: if $\Gamma \vdash A$, then $\Gamma \cup \Delta \vdash A$.

Proof: 1. $\Gamma \vdash A$ given
 2. $\Gamma \vdash LE \{A\}$ for all $LE \in L$ 1, soundness
 3. $\Gamma \cup \Delta \vdash LE$ for all $LE \in L$. subset principle
 4. $\Gamma \cup \Delta \vdash A$ 2,3, transitivity
 5. $\Gamma \cup \Delta \vdash A$ 4, completeness

What the heck is going on? We do reason non-monotonically, and it does seem to be rational. We know (see[5]) that probability theory can be used as a formal semantics for just about any logic. Further, probability theory has non-monotonic aspects; we can certainly have $\Pr(A, \Gamma)$ very high but $\Pr(A, \Gamma \cup \Delta)$ very low. And we think probability theory is related to rational belief. Surely we should be able to define an entailment relation that is non-monotonic by using probability theory.

In brief, the answer is that we cannot. Although individual conditional probability distributions are non-monotonic in the way indicated, we can still prove the same theorem concerning any entailment relation based on universal properties of conditional probability distributions. Although not completely general in all respects, the main thrust of the proof may be found in [6].

So, non-monotonic "algebras" cannot correspond to

logical entailment based on rational belief. No matter how you formulate it, if logical entailment is to correspond to universal principles of rational belief, then logical entailment is monotonic. In brief, non-monotonic logic is impossible.

There is a very large and growing literature on the topic of non-monotonic logic, and much research is concentrated on the development of non-monotonic inference systems. We cannot possibly review that literature here. However, the interested reader will find [1] and [9] to be good starting points for summaries and references to the main streams in the literature. In light of the impossibility theorem, much of the research seems to be misguided. We will now try to extract a few lessons from the theorem in order to shed some light on various trends in the field, but we will avoid commenting on particular research programs.

Some Important Lessons

There are a number of important lessons to be drawn from our impossibility theorem. Perhaps the first lesson is that we **cannot** account for non-monotonic inference by simply introducing new connectives into the language and formulating new formal rules concerning those connectives. The proof of the impossibility theorem does not in any way depend on the connectives in the language nor on the sort of inference rules used. Thus non-classical logics incorporating new conditionals or modal operators are just as subject to the theorem as classical logic. There may be other reasons to look at variations on, or deviations from, the formal structures of classical logic (e.g., the introduction of epistemic operators), but changes in formal logic cannot be justified by claims that the resulting system yields an entailment relation that is non-monotonic. Any such system must fail to accord with rational belief, or it must have a monotonic entailment relation.

The second lesson is that we must distinguish between non-monotonic **inference** and non-monotonic **logic**. Because the term "logic" is used in such a loose way, it is easy to slip from an informal use to a more formal use, without being explicitly aware of the shift.

For example, in a very loose sense, we use "logical" to mean "reasonable." So to ask for the logic behind someone's action is to ask for information that would make that action appear rational to us; we want to see that given another's motivations and beliefs, we might do the same thing. In this sense, there certainly is a logic to most cases of non-monotonic inference; we can see that in some circumstances, when given more information, changing your conclusion is rational. However, we should not conclude from this trivial observation that there is some formal system, with different inference rules and/or connectives from standard logic, that must be used to account for such inferences. For purposes of illustration, we will briefly describe two approaches to non-monotonic inference systems that can be developed using just the principles of classical logic.

In many cases, non-monotonic reasoning is based on simple statistical information derived from past experience. Our first system is based on the calculation of conditional probability by means of relative frequency based on past experience. It then relies on the rule that if $\Pr(A, \Gamma) > .5$ then it is reasonable, given Γ , to act as if A is the case.

Much of so-called common sense reasoning is concerned with predicting simple, classificatory properties of specific individual items, such as humans, cars, or chairs. In general, we use our past experiences of such items as a guide to our expectations concerning the item in question. This process is relatively easy to model with standard conditional probability theory, using the approach pioneered by Carnap in [2]. Suppose our language contains individual constants, predicates of arbitrary degree (monadic, dyadic, etc.), and the standard connectives from classical logic, but no quantifiers. Since our experience is finite, we assume we have a finite set of background information Σ about only a finite number of individuals:

$$I = \{a_1, a_2, \dots, a_n\}$$

Suppose we are given information $\Gamma(b)$ about some new individual b , and we want to know whether or not to conclude $E(b)$. The set $\Gamma(b)$ contains sentences from the language concerning b and perhaps any of the other individuals in I . The sentence $E(b)$ can be any complex sentence concerning b and perhaps also members of I . For the purposes of the inference, we may adopt the "closed world" hypothesis, and **act as if** the only items in the universe are those in I . We restrict our attention to those items x in I for which $\Gamma(x)$ is true, as determined by Σ . If there are no such items or if 50% or fewer have $E(x)$, then we cannot draw the conclusion. If more than 50% of such items have $E(x)$, then we conclude $E(b)$.

This simple scheme has some similarity to circumscription, and is applicable to many of the problems for which circumscription was designed. However, this scheme is finitary, easy to implement, and computationally tractable. It does not require any new logic and accords well with how people actually reason in such cases; in fact, it is a general version of reasoning by analogy. It is obviously non-monotonic, since $\Pr(E(x), \Gamma(x))$ may be greater or less than $\Pr(E(x), \Gamma(x) \cup \Delta(x))$; and as we gain more experience, i.e., as the sets Σ and I grow, the probability distribution \Pr itself may change. Without going into great detail, it is easy to see how our earlier example about Tilly, animals, birds, penguins, and flying could be handled with this technique. If in our past experience, most animals do not fly, most birds do fly, and most penguins do not fly, then our simple scheme will draw the right conclusions each time.

There is a very obvious lesson which this technique serves nicely to illustrate. The lesson is the importance of the concept of "acting as if". No one seriously assumes that they have had experience with every object in the universe, and

it would be pernicious to suggest that the closed world assumption commits one to that view. Rather, we **act as if** that assumption were correct, even though we know it is not. Scientific practice is full of such artificial abstractions, such as the ideal gas laws. The point is that when we must "get around" in the world, we frequently act as if A is true, even when we know it isn't. In conditions of uncertainty, adopting hypotheses **and** making inferences are more akin to "acting as if" rather than "asserting the truth of". Nor is "acting as if" simply a matter of high probability. No one claims that A must be true just because it has a high probability. Indeed, it may well be that in some instances we would be irrational to act as if A is the case, even when A does have high probability; for instance, if there are terrible consequences of acting on A when it is false, then we should be very cautious. In short, utilities as well as probabilities must be brought to bear. Such subtleties are beyond the scope of this discussion.

There are some obvious difficulties with the technique. As sketched, it does not really specify how one is to handle counter-factuals, i.e., situations in which $\Gamma(x)$ contradicts Σ . Nor does it specify how to deal with "missing information", i.e., cases in which neither P nor $\neg P$ is in Σ , but the value of P is needed for the computation. But these are minor technical problems which we cannot discuss here because of space limitations. A more fundamental difficulty is that the scheme is limited to rather simple languages; there are a great many examples which essentially involve quantifiers. One solution is to emphasize the closed world assumption, and assume all quantifiers range over just the individuals mentioned in Σ . This approach works for a great many cases, but not all. For example, Σ may in fact commit us to an infinite universe, so all finite expansions will be inconsistent. Again, space limitations prohibit a detailed consideration of such matters. (I do not wish to suggest that this approach is the only way of utilizing probability in a non-monotonic inferencing system; for a very good alternative, see [3].)

The astute reader may at this point suggest that we just develop a formal language in which we incorporate a special conditional corresponding to the conditional of conditional probability theory. Briefly put, it is impossible to do so. It is demonstrable that any such conditional fails to be monotonic or violates principles of classical probability, or both. See [4] and [8].

We now turn to our second proposal for a non-monotonic inference system. A little bit of empirical research in the psychology of human reasoning (indeed, a little bit of self-reflection) suggests that we often adopt unstated background assumptions to fill in gaps in our knowledge. These "default" assumptions are abandoned or changed in light of new information, and such changes in the "default" assumptions lead to changes in the conclusions we draw. Non-monotonic inferences are often the result of alterations in our background beliefs: "Gee, I guess I was mistaken about that..." In many cases, the background beliefs under which we are operating

are not explicitly stated, and frequently they are not even consciously entertained. Default logic is an area of very active research in AI.

The fact that humans do reason this way is an empirical, rather than a logical fact. Which default assumptions people actually adopt in various situations is a matter for empirical, rather than logical, investigation. Abstract logical and mathematical research **alone** will not yield systems that reflect human reasoning in a majority of real situations. How do we get those background beliefs? How are they revised? These are very good questions concerning belief acquisition and revision, but these questions do not have an immediate impact on formal logic. They certainly will have an impact on inference construed as a human activity and on our attempts to use computers to model that activity.

As a human activity or process, non-monotonic inferences based on defaults occur with classical logic and classical connectives all the time. Classical logic does not represent the evolution of background assumptions. At best it represents a static, snapshot-like, relation between premises and conclusions. Making inferences and drawing conclusions are human **activities**. Reasoning in a non-monotonic way does not necessarily require a formal logic that differs from classical principles. When we change our background beliefs, the justified conclusions are likely to change, and this fact is independent of the formal logic.

Once again, no new logic is required in order for us to construct models of default reasoning. Indeed, there are simple models of such processes for virtually any monotonic logic. One such technique is derived from that used to construct so-called canonical probability distributions in [7], and we will briefly outline the process here.

As before, we will use " \vdash_L " to represent an arbitrary logical entailment relation. We will say that a set Γ is **L-consistent** (consistent with respect to logic **L**) if and only if there is at least one sentence A such that it is not the case that $\Gamma \vdash_L A$. Now, let us assume to be given some (preference) ordered set of sentences (potential background assumptions):

$$S = \{H_1, H_2, \dots\}$$

We will use the notation $\Delta = >_{L,S} E$ as shorthand for the English claim: It is rational to infer E from Δ , given logic **L** and the ordered set S of potential background assumption. First, we will use the specified premise set Δ and the ordered hypothesis set S to determine an expanded premise set $\Delta_{L,S}$. We make the following definitions:

$$\begin{aligned} \Delta_0 &= \Delta \\ \Delta_{i+1} &= \Delta_i \cup \{H_{i+1}\}, \text{ if } \Delta_i \cup \{H_{i+1}\} \\ &\quad \text{is } \mathbf{L}\text{-consistent} = \Delta_i, \text{ otherwise} \\ \Delta_{L,S} &= \cup \Delta_i \end{aligned}$$

$$\Delta = >_{L,S} E \text{ if and only if } \Delta_{L,S} \vdash_L E$$

The set $\Delta_{L,S}$ is just Δ augmented by the largest number of auxiliary hypotheses, considered in the specified order, that can be consistently added. Carefully note that there is no requirement that S itself be consistent. Even if S is inconsistent taken as a whole, individual members of S may well be consistent with Δ . Of course if Δ is not consistent, then $\Delta_{L,S}$ will just be Δ .

So, the definition says that it is legitimate to infer E from Δ just in case E is logically entailed by Δ augmented by hypotheses that may be consistently added, in order, from the set S of potential background assumption. The relation " $= >_{L,S}$ " will be non-monotonic, since $\Delta \cup \Delta'$ may very well produce a very different augmented premise set than that produced by Δ alone.

As a simple example, we return to the one from the introduction concerning animals, birds, penguins, and flying. First, we specify the ordered set of potential background assumptions, using the obvious notation:

$$S = \{(x)((Px \wedge Bx \wedge Ax) \supset \sim Fx), (x)((Bx \wedge Ax) \supset Fx), (x)(Ax \supset \sim Fx)\}$$

Note that in our example, the hypotheses are simple universal conditionals, and they are ordered by the decreasing specificity of the antecedents.

$$\{At\}_{L,S} = \{At\} \cup S$$

$$\{At, Bt\}_{L,S} = \{At, Bt\} \cup \{(x)((Px \wedge Bx \wedge Ax) \supset \sim Fx), (x)((Bx \wedge Ax) \supset \sim Fx)\}$$

$$\{At, Bt, Pt\}_{L,S} = \{At, Bt, Pt\} \cup \{(x)((Px \wedge Bx \wedge Ax) \supset \sim Fx)\}$$

When told only that Tilly is an animal, we conclude that Tilly cannot fly. When told in addition that Tilly is a bird, we revise our conclusion and say that Tilly can fly. And finally when it is further revealed that Tilly is a penguin, we conclude that Tilly cannot fly.

Many researchers feel uncomfortable with any analysis which employs notions of consistency and inconsistency, since consistency is undecidable for first-order logic. There are a number of things to say in response to such a criticism. First note that the entailment relation for first-order logic is undecidable, but we still want to employ it in our models of reasoning. Further, most humans have great difficulty reasoning with full first-order logic. In addition, there are many well-known cases for which the decision problem is solvable; for example, the logic of monadic predicates is decidable. So if we only want to model human reasoning, it is probably sufficient to rely on notions of consistency and entailment relative to some resource limited computational process.

We have briefly examined two quite general examples of how non-monotonic inference systems may be based on standard monotonic logic. Although the two systems are very different, we have shown how both of these systems may be used to handle the same sequence of simple inferences about Tilly. So an additional lesson is that there may be many non-equivalent inferential systems which would handle any given set of examples. In short, even if we could settle on one standard monotonic logic, it may be the case that there is no single "correct" non-monotonic system. The situation is analogous to what we find in automated theorem proving. There are many ATP systems for first-order logic. For certain kinds of problems, one approach works well, whereas for other problems, a different approach is needed.


There is one more lesson we should draw from these inference systems, and that concerns empirical content. One might suppose that each of these systems determines some sort of relationship between premise sets and conclusions that we might be tempted to think of as a non-monotonic entailment relation. How is it that these examples avoid the impossibility proof? The answer is that these systems abandon the soundness principle. The violation of soundness is easy to see when considering conditional probability. Essentially, we want to conclude A from Γ when $\Pr(A, \Gamma) > .5$. For a particular probability distribution, we may well have $\Pr(A, \Gamma) > .5$, but at the same time in many cases there will be other distributions such that $\Pr(A, \Gamma) < .5$. Further just because $\Pr(A, \Gamma) > .5$, it does not follow that $\Pr(\Gamma) < \Pr(A)$, where these latter values are properly defined on the basis of conditional distributions. Further, given the proper definitions, $\Pr(\Gamma, \Gamma) = 1$, no matter what the value of $\Pr(A, \Gamma)$. So there is no reasonable way in which soundness may be construed to be satisfied by the rule: draw conclusions whose probability, given the premises, is greater than .5.

The abandonment of soundness by non-monotonic inference systems emphasizes the fact that non-monotonic inference is just a fancy name for the process historically known as induction. In non-monotonic inference, the "conclusion" goes beyond what is sanctioned by the premises. Indeed, introductory students are often taught that if additional premises could be adduced that would change the degree of support a given set of premises provide for the conclusion, then the original argument is inductive in character. That is, non-monotonicity is the mark of induction.

In formal terms, non-monotonic inferences are model specific. In other words, in order to know which inferences are acceptable, we must know details concerning the model. Thus non-monotonic inference systems inescapably contain empirical information; they are not content neutral. We cannot construct non-monotonic inference systems without building in empirical assumptions about how our world works. Any system that is truly non-monotonic necessarily

has some empirical assumptions built into it, over and above the premises of any specific argument. It would be better if researchers in the area realized this fact and tried to be more explicit about the empirical assumptions behind their systems.

References

- [1] Brewka, G. *Nonmonotonic Reasoning: Logical Foundations of Commonsense*, Cambridge University Press, Cambridge (1991).
- [2] Carnap, R. *Logical Foundations of Probability*, 2nd ed., University of Chicago Press, Chicago (1962).
- [3] Kyburg, H. "Believing on the basis of the evidence," *Computational Intelligence*, vol. 10 (1994), pp. 3-20.
- [4] Morgan, C. and Mares, E. "Conditionals, probability, and non-triviality," *Journal of Philosophical Logic*, vol. 24 (1995), pp. 455-467.
- [5] Morgan, C. "Logic, probability theory, and artificial intelligence — Part I: the probabilistic foundations of logic," *Computational Intelligence*, vol. 7 (1991), pp. 94-109.
- [6] Morgan, C. "Evidence, belief, and inference," *Computational Intelligence*, vol. 10 (1994), pp. 79-84.
- [7] Morgan, C. "Canonical models and probabilistic semantics," *Poznan Studies*, forthcoming.
- [8] Morgan, C. "Conditionals, comparative probability, and triviality," forthcoming.
- [9] Smets, P. *et al.*, eds. *Non-Standard Logics for Automated Reasoning*. Academic Press, London (1988). 

Charles Morgan is currently professor of philosophy at University of Victoria, and director of the Varney Bay Institute for Advanced Study. He holds M.Sc. degrees in applied mathematics (Johns Hopkins, 1970), in computing science (U. of Alberta, 1972), and in astrophysics (U. of Victoria, 1985), as well as a Ph.D. in philosophy (Johns Hopkins, 1970). He published some of the earliest work in automated inductive reasoning and in automated theorem proving for non-classical logics. His recent research concerns the relationships between probability theory and logic.

From Assumptions to Meaning *

Veronica Dahl and Paul Tarau

Abstract

LES ASSOMPTIONS

Les linguistes et les logiciens ont mis au jour les multiples facettes du rôle des assumptions dans les communications humaines, et ont développé une quantité de façons de mener un raisonnement basé sur les assumptions. Dans cet article, nous décrivons les extensions à BinProlog qui permettent la conception d'outils puissants de traitement du langage naturel pour un raisonnement basé sur des assumptions intuitionnistes et linéaires.

1 Introduction

In human communication, assumptions play a central role. Linguists and logicians have uncovered their many facets. For instance, the assumption of a looking glass' existence and unicity in "The looking glass is turning into a mist now" is called a *presupposition*; the assumption that a polite request, rather than a literal question, will be "heard" in "Can you open the door?" is an *implicature*.

Much of AI work is also concerned with the study of assumptions in one form or another: abductive reasoning "infers" (assumes) p from q given that $p \Rightarrow q$; uncertain information with high probability is used (assumed) in some frameworks, and so on. Recently, systems for hypothetical reasoning that extend the capabilities of deductive databases have been studied e.g [4] and in the area of logic programming, assumptions have also been widely used, most notably for default reasoning (e.g. negation as failure).

Independently, work on intuitionistic logic and, more recently, linear logic [12] has provided formally characterized embodiments of assumptions which have been influential on logic programming and logic grammars [15, 16, 14]. The result is not only a better understanding¹ of their proof-theoretical characteristics, but also a growing awareness on the practical benefits of integrating them in conventional Prolog systems.

Such a state of affairs, coupled with our desire to design powerful tools to deal with the complex reasoning problems which arise in natural language processing, led us to endow BinProlog [19] with intuitionistic and linear assumptions scoped over the current continuation. Their use for logic grammars motivated in turn a variant of Extended DCGs [22] that handles multiple streams invisibly without the need of preprocessing. Surprisingly, the outcome went beyond the intended ap-

plication domain. A unified approach to handle *backtrackable state information in non-deterministic logic languages*, based on a simplified form of linear affine and intuitionistic implication (assumptions) has emerged.

Using this approach, we can draw meaning from discourse by making assumptions that are backtracked upon when and if they prove wrong, until only appropriate assumptions persist. Three natural language uses of Assumption Grammars have been examined: free word order, anaphora and coordination. We have also shown two results which were surprising to us, namely: a) Assumption grammars allow a direct and efficient implementation of link grammars — a context-free like formalism developed independently from logic grammars; and b) they offer the flexibility of switching between data-driven or goal-driven reasoning, at no overhead in terms of either syntax or implementation. We have also shown Datalog grammars [7, 2, 6] to be an instance of Assumption Grammars. Given space limitations, here we shall only describe our hypothetical reasoning tools, and exemplify their use for gleaning implicit meaning from coordinated sentences (i.e., sentences with "and", "or", "but", etc.).

2 Assumption Grammars

Assumption Grammars are basically like definite clause grammars, except that they can handle multiple accumulators invisibly, and that they possess linear and intuitionistic implications scoped over the current continuation.

Intuitionistic assumption (*/1) temporarily adds a clause that can be used an indefinite number of times in subsequent proofs. Linear assumption (+/1) adds a clause that may be used at most once in subsequent proofs (as in *affine* linear logic, rather than in Girard's original framework where linear assumptions should be

*This extended abstract is based on research first reported in two conference papers [21] and [9]. A full version has been submitted for publication.

¹For instance the presence of *uniform proofs* [17]) indicate that a formalism is likely to be computationally interesting.

used *exactly* once). A builtin predicate ($-/1$) was introduced to allow for the removal of either intuitionistic or linear assumptions.

Both types of assumption have scoped versions and, unlike `assert/1`, vanish upon backtracking. We can see them respectively as linear affine and intuitionistic implication scoped over the current AND-continuation, i.e. having their assumptions available in future computations on the *same* resolution branch.

For instance, `*a(13), a(X), a(Y)` succeeds, whereas `+a(13), a(X), a(Y)` will fail. Note however that `*a(13), *a(12), a(X)` will succeed with $X=12$ and $X=13$ as answers, while its non-affine counterpart `a(13) -o a(12) -o a(X)` as implemented in Lolli or Lygon, would fail².

As an example of a scoped version of assumptions, the intuitionistic implications `Clause=>Goal` and `[File]=>Goal` make `Clause` or respectively the set of clauses found in `File`, available only during the proof of `Goal`. For instance, `a(13) => (a(X), a(Y))` will succeed.

3 Timeless Assumptions

These simple first builtins proved excellent for clean and efficient solutions to interesting computational linguistics problems, such as free word order, sentence coordination and many cases of anaphora [18, 8, 9].

For instance, for handling anaphoric reference (in which a given noun phrase in a discourse is referred to in another sentence, e.g. through a pronoun), all we have to do is assume each noun phrase parsed as a potential referent of some pronoun that might appear later, and let pronouns consume those assumed noun phrases which agree in gender, number, etc., backtracking if necessary.

However, we ran into problems when trying to describe backwards anaphora (as in “Near her, Alice saw a Bread-and-butter-fly”, where the referent *Alice* appears after the pronoun *her* that refers to it), or in some cases of coordination where again, potential referents need to be consumed *before* they have been assumed (as in “Tweedledum proposed and Tweedledee agreed to a battle”, in which the missing object of “proposed” needs to be consumed from an assumed potential referent before this referent (“a battle”) has had a chance of being assumed).

These problems motivated an alternative kind of assumption, which we named “timeless”, or `=` for short. A timeless assumption is consumed if made, and if not yet made, consumption blocks until a matching assumption shows up.

Timeless assumptions not only elegantly solved the problems of backward reference that motivated them

[9], but also proved to be a beautiful basis for coordinating agent programming in virtual worlds, with synchronization information built in the “container objects” and allowing more flexible wait/notify negotiations between consumer/producer agent components, as well as inheritance and agent component reuse [20].

Further investigation of timeless assumptions in the context of knowledge-based systems [9] uncovered the need for subtly different types of timeless assumptions than the ones described here.

4 A sample grammar

Coordination (grammatical construction with the conjunctions “and”, “or”, “but”) has long been one of the most difficult natural language phenomena to handle, because it can involve such a wide range of grammatical constituents (or non-constituent fragments), and ellipsis (or reduction) can occur in the items conjoined. In most grammatical frameworks, the grammar writer desiring to handle coordination can get by reasonably well by writing enough specific rules involving particular grammatical categories; but it appears that a proper and general treatment must recognize coordination as a “metagrammatical” construction, in the sense that metarule, general system operations, or “second-pass” operations such as transformations, are needed for its formulation. Early attempts at such a general treatment [23, 3] were inefficient due to combinatorial explosion. A logic grammar rendition of coordination in terms of logic grammars [5] solved these inefficiencies through the addition of a semantic interpretation component that produced a logical form from the output of the parser and dealt with scoping problems for coordination. We next exemplify a metagrammatical treatment to coordination through AGs. Terminals are preceded by ‘#’.

```
sent(and(S1,S2)):- s(S1), +and, s(S2).
% conjunction of two sentences- assumes that
% there will be an "and" between them.

s(S):- name(K), verb(K,P,S), np(P).

np(P):- det(X,P1,P), noun(X,P1),
        =(ref_np(P)).
% keep it as potential referent for a missing np
np(P):- #and, -and,
        % a conjunction appears where an np is
        % expected: consume "and"
        =-(ref_np(P)).
% consume an assumed (or to be assumed) np
% in lieu of the missing one

det(X,P,the(X,P)):- #the.
noun(X,cupboard(X)):- #cupboard.
```

²The linear implication is denoted `-o` in Lolli.

```
name(tim):- #tim.  
name(anne):- #anne.
```

```
verb(X,Y,built(X,Y)):- #built.  
verb(X,Y,painted(X,Y)):- #painted.
```

For the sentence *Tim built and Anne painted the cupboard*, for instance, we obtain the semantic representation:

```
and(built(tim,the(X,cupboard(X)),  
painted(ann,the(X,cupboard(X))))
```

which is just what we intend in this simplified example. Subtler analyses can be implemented as in [10].

5 Conclusion and Related Work

Assumption Grammars, although theoretically no more powerful than previous logic grammars, have more expressive power in that they permit the specification of rewriting transformations involving components of a string separated by arbitrary strings with the sole resource of intuitionistic and (affine) linear assumption scoped over the current AND continuation.

Existing work on Linear Logic based Natural Language processing [13, 1] is mostly at sentence level, while ours covers text level constructs. This is made easy by using hypothetical assumptions which range over the current continuation, instead of locally scoped implications.

The semantics of our constructs is an instance of the sequent calculus based descriptions of Horn Clause Logic and the more powerful *uniform proof* based systems of [14]. We can see AGs and accumulator processing in general as an even more specific instance of linear operations.

Compared to previous frameworks based on Linear (Intuitionistic) Logic, ours is portable and runs on top of generic Prolog systems. This is a practical advantage over systems like Lolli or λ Prolog.

Acknowledgements

We thank for support from NSERC (grants OGP0107411 and 611024), and from the FESR of the Université de Moncton. We also thank Andrew Fall for work reported in [11].

References

[1] J.-M. Andreoli and R. Pareschi. Linear objects: Logical processes with built-in inheritance. In D.H.D. Warren and P. Szeredi, editors, *7th Int. Conf. Logic Programming*, Jerusalem, Israel, 1990. MIT Press.

- [2] J. Balsa, Dahl V., and Pereira Lopez J. G. Datalog Grammars for abductive syntactic error diagnosis and repair. In Springer, editor, *Proceedings NLULP'95, LNCS*, May 1997.
- [3] M. Bates. The theory and practice of augmented transition network grammars. In *Natural Language Communication with Computers*, Bole, L. (ed), pages 191–259, 1978.
- [4] A. J. Bonner. A logical semantics for hypothetical rulebases with deletion. *Journal of Logic Programming*, 32(2):119–170, 1997.
- [5] V. Dahl and M. McCord. Treating coordination in logic grammars. In *American Journal of Computational Linguistics* 9, pages 69–91, 1983.
- [6] V. Dahl, P. Tarau, and J. Andrews. Extending Datalog Grammars. In *Proc. of NLDB'95, Paris*, May 1995.
- [7] V. Dahl, P. Tarau, and Y. N. Huang. Datalog Grammars. In *Proc. 1994 Joint Conference on Declarative Programming*, pages 268–282, Peniscola, Spain, September 1994.
- [8] Veronica Dahl, Andrew Fall, Stephen Rochefort, and Paul Tarau. A Hypothetical Reasoning Framework for NL Processing. In *Proc. 8th IEEE International Conference on Tools with Artificial Intelligence*, Toulouse, France, November 1996.
- [9] Veronica Dahl, Paul Tarau, and Renwei Li. Assumption Grammars for Processing Natural Language. In Lee Naish, editor, *Proceedings of the Fourteenth International Conference on Logic Programming*, pages 256–270, MIT press, 1997.
- [10] A. Fall, V. Dahl, and P. Tarau. Resolving co-specification in contexts. In *Proc. of IJCAI Workshop on Context in Natural Language Processing*, Montreal, Canada, 1995.
- [11] Andrew Fall, Veronica Dahl, and Paul Tarau. Resolving Co-specification in Contexts. In *Proc. of IJCAI'95 Context in Natural Language Workshop*, August 1995.
- [12] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, (50):1–102, 1987.
- [13] J. Hodas. Specifying Filler-Gap Dependency Parsers in a Linear-Logic Programming Language. In Krzysztof Apt, editor, *Logic Programming Proceedings of the Joint International Conference and Symposium on Logic programming*, pages 622–636, Cambridge, Massachusetts London, England, 1992. MIT Press.

- [14] Joshua S. Hodas. *Logic Programming in Intuitionistic Linear Logic: Theory, Design, and Implementation*. PhD thesis, University of Pennsylvania, Department of Computer and Information Science, May 1994. Available as University of Pennsylvania Technical Reports MS-CIS-92-28 or LINC LAB 269.
- [15] D. Miller and G. Nadathur. Some uses of higher-order logic in computational linguistics. In *24th Annual Meeting of the Association for Computational Linguistics*, pages 247–255, 1986.
- [16] D.A. Miller. A logical analysis of modules in logic programming. *J. Logic Programming*, 6(1–2):79–108, 1989.
- [17] D.A. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, (51):125–157, 1991.
- [18] P. Tarau, V. Dahl, and A. Fall. Backtrackable State with Linear Assumptions, Continuations and Hidden Accumulator Grammars. In *ILPS'95 Workshop on Visions for the Future of Logic Programming*, November 1995.
- [19] Paul Tarau. BinProlog 5.75 User Guide. Technical Report 97-1, Département d'Informatique, Université de Moncton, April 1997. Available from <http://clement.info.umoncton.ca/BinProlog>.
- [20] Paul Tarau and Veronica Dahl. A Coordination Logic for Agent Programming in Virtual Worlds. In Wolfram Conen and Gustaf Neumann, editors, *Proceedings of Asian'96 Post-Conference Workshop on Coordination Technology for Collaborative Applications*, Singapore, December 1996. to appear in LNCS, Springer.
- [21] Paul Tarau, Veronica Dahl, and Andrew Fall. Backtrackable State with Linear Affine Implication and Assumption Grammars. In Joxan Jaffar and Roland H.C. Yap, editors, *Concurrency and Parallelism, Programming, Networking, and Security*, Lecture Notes in Computer Science 1179, pages 53–64, Singapore, December 1996. Springer.
- [22] Peter Van Roy. A useful extension to Prolog's Definite Clause Grammar notation. *SIGPLAN notices*, 24(11):132–134, November 1989.
- [23] W.A. Woods. An experimental parsing system for transition network grammars. In *Natural Language Processing*, Rustin R. (ed.), pages 145–149, 1973.

Veronica Dahl is internationally known for having pioneered the introduction of logic programming into the fields of deductive databases, expert systems and natural language front ends. She is also known for her extensive research on building logic programming tools for processing language, and on producing useful syntheses between linguistic theory and logic grammars. email veronica@cs.sfu.ca

Paul Tarau holds a Ph.D. from Université de Montreal and is an Associate Professor of Computer Science at the Université de Moncton, Canada. He is the author of BinProlog, a high performance continuation passing Prolog system. He has worked on compilation of logic programs, translation from Prolog to C, logic grammars and Internet technologies. Recently, he has developed the LogiMOO system, a BinProlog based Virtual World for live interaction and collaborative work on the Internet. email tarau@info.umoncton.ca



Book Reviews

On the Origin of Objects Brian Cantwell Smith, The MIT Press, 420 pp.

Reviewed by
André Vellino

National Research Council, Ottawa

On the Origin of Objects is a seminal book. It challenges us to rethink our entire world-view and revitalizes the perennial questions of philosophy: what is knowledge?, what is it knowledge of?, what is the nature of consciousness?, what is meaning?, what is personal identity?, how ought I to act?, what is politics?, what is beautiful? It is also a courageous book: to address such sweeping questions entails creative metaphysical contemplations of a kind that will no doubt ruffle the feathers of academic philosophers. Even though he does not try to answer all these questions nor even tackle most of them head-on, Smith offers the reader a fresh foundation from which pluralistic solutions to those problems may be fashioned.

Smith's arguments are complex and demand the reader's careful attention. To understand the core of Smith's book the reader must follow an exacting journey through its labyrinth of concepts, distinctions and arguments: it is useless to go straight to the conclusion without first going through the phases of problem "analysis" and solution "construction." The book is also impossible to compress. Any attempt to summarize or paraphrase even the general thrust of the argument will inevitably do it some injustice or other.

The starting point for Smith's multi-threaded excursion into a new metaphysics is not a new one for either AI or cognitive science. The desire to provide a solid foundation for a cognitivist computational theory of mind (the theory that mind is computational in nature) has motivated many a tome in recent years [1-5]. Smith's take on the problem, however, quickly leads into murkier philosophical waters than his predecessors had dared to venture before. In trying to break out of the mould of contemporary cognitive science--mind/body duality, representation schemas, semantics of computation etc.--Smith attempts to re-orient the reader to a new way of thinking about the questions.

The foundations of any field are either philosophical or mathematical, and this one is certainly not mathematical. Notwithstanding his disclaimer that this book is not philosophical either, Smith might have considered another

Future Computational Theory of Mind", perhaps. Indeed, this book is a sort of preamble to a much more detailed and finely articulated 5-volume encyclopedic analysis of computational theory which Smith promises us for the future. We are left anticipating that the results will be to the foundations of computing as Knuth's trilogy was for the art of computer programming, i.e., a *tour de force*.

The basic problem with which Smith is wrestling that if we want to build software systems that are able to represent and reason about the world in which they operate, we need to understand both what is represented and the representation, not to mention the process of reasoning and norms of rationality. Yet the conventional model of unambiguous "objects" that have "relations" to one another and which need only to be represented in some symbolic computer language is a completely non-explanatory and inadequate account of both the world and its representation in a computer. Some new and foundational perspective is required.

Like other philosophical system builders that propose new metaphysical and epistemological foundations, Smith believes this task is necessary because of the failures of the past. He is right, for example, in noting that there has been some fuzzy thinking about semantics. The problem of semantics isn't solved by the proposition that computing is symbol manipulation, he says, since symbol manipulation *is* intentional (it is easy to guess where Smith stands on the "Chinese room" argument). Nor is the problem of intentionality eliminated by equating computation with the behaviour of finite-state automata. Indeed, it turns out that, for Smith, intentionality isn't reducible to anything.

Where then, does Smith stand on philosophical issues? In terms of familiar categories, one could say that his epistemology is that of a realist (in the sense that "there is a world out there" which our theories attempt to capture), that his methodology is naturalist (in the sense that explanations of all phenomena must be grounded in nature), and, although he denies it, his metaphysics is largely that of a materialist (in the sense that the furniture of our experience--and the essence of computation--is a material/physical phenomenon). But for many reasons with which it is easy to sympathize, Smith distrusts these familiar categories and the sorts of prejudices they conjure up. He wants his analysis to make room for different ways of speaking and different formalisms (i.e. pluralism) and so he doesn't side with any particular camp. Instead he tries to position himself *vis à vis* well-known theories (of truth, of semantics, of representation, etc.).

What then do these philosophical theories have to do with everyday objects like tables and chairs, or even with the problem of how to represent them for a computer? What Smith wants to do in his inquiry into “objects” is

“..[to find out what] is needed for a complete picture of intentionality, semantics and ontology ...[for which one needs] to understand how a conception of objects can arise on a substrate of infinitely extensive fields of particularity” [p. 191].

There are all sorts of things wrong with this clumsy formulation, as Smith himself readily admits, and he eventually offers us the notion of “registration” to shed some light on the matter. But to understand how our conception of objects arises at all we first need to understand the distinction between “particulars” and “individuals.” Examples of particulars include the constitutive elements of what we ordinarily call (physical) objects (the physical cause of this patch of green/blue experience) but also events (such as Diana’s accident) and processes (the writing of this review). Contrast this with universals (e.g. the generic concept “canoe”) and abstract categories. The distinction between particulars and individuals—and this is an important one—is that individuality is what makes an object *discrete* and separable from its background. An individual is constituted of particulars, but what makes it an individual is “... *what allows us to say of one object that it is one; or two that they are two*” [p. 119].

One might well think that the science of physics would do quite well as a foundational theory of particularity, individuality and objecthood. After all, physics is the star candidate for an explanatory theory of all there is, and Smith has to explain why it is not enough. The argument, which reads like a romp through 80 years of philosophy of science—ranging from the ontological presuppositions of classical and quantum mechanics to Popper’s “three worlds” theory—concludes that physics has something to say about particulars but is silent about individuals. One of the lessons drawn from this foray into physics is the *criterion of ultimate concreteness*:

No naturalistically palatable theory of intentionality—of mind, computation, semantics, ontology, objectivity—can presume the identity or existence of any individual object whatsoever. [p. 184, italics in the original]

It turns out that what distinguishes an individual from the “rest of the world” is something abstract and dependent on the subject that is doing the abstraction. Hence the problem of what “objects” are and how we “register” them remains.

What, then, is “registration”? In brief, we “register” the world (we take it to be a certain way, to be composed of certain things) not because we *conceive* it to be this way or even *perceive* it to be that way (as the 18th century British Empiricists might have it) but because its being what it is

requires an intentional relation to a subject. For Smith, the intentional nature of objects and their registration in our minds will never be reduced, as an eliminative materialist would have it, to some causal story about neurons firing in some particular way as a result of stimuli of a certain kind. Registration, it seems, is a primitive and irreducible intentional relation between mind and object.

The modalities of what precisely counts as registration, by whom and how it takes place, obliges Smith to sketch a theory of causality. Here again, nothing is settled but a metaphysical picture is painted that is plausible: the world is neither entirely deterministic—a world of intertwined cogs and gears—nor is everything physically and causally disconnected from everything else. Smith goes for the middle ground, one that captures both the intuition that what I am doing now is not terribly relevant or causally influential to whether you are or are not currently drinking a glass of water, and the intuition that my use of garlic as an ingredient for the pasta sauce I cooked last night had a significant influence on my guests’ enjoyment of the meal.

That bears on registration in the following way: an object is registered by a subject not only by the causal link between them (the “world” impinging on the senses, for example), but also by how it does *not* causally relate to the subject. In particular, an object’s continued existence or movement in space bears an intentional relation to the subject that has as much to do with its causal relations to the subject as it does to the causal *disconnection* between the subject and the object.

In addition to a theory of causality, Smith needs a theory of reference (and, therefore, Truth). For it must be possible for me to register objects not only as present, here and now, in my perceptual field, but also as enduring, “stable” objects that preserve their identity somehow. This process of *stabilization* requires the ability to deal with indexicals (such as *this* and *now*) which, in turn, demands a theory of reference. To explain this process of object registration, Smith spends chapter 7 (one of two on registration) speaking of “s-regions” and “o-regions” instead of the more vexed “subjects” and “objects” and of the important (and, I take it, intentional) “letting go” of one by the other, as an essential requirement for registration. I think Smith means that the process of individuation that takes place in selecting a “thing” from its environment and registering it *as* a thing (or object) requires an (intentional) act of separation or discrimination.

In any event, the registration of an object is no trivial matter. Apart from the problem of context (“A teacup’s exemplification of the property of being a teacup does not inhere in its meager six ounces; it reaches back into British society” [p. 269]), there is the question of how we register higher-order cognitive objects like concepts and the process of non-cognitive registration—that which we register simply

as a consequence of being alive and derives from our “encounter” with things. This means, that there must be an equilibrium (a “middle distance”) between the degree to which an object is causally connected with the subject and the degree to which it is separate. Ditto with the extent to which the subject’s concepts participate in the object’s being what it is and the extent to which we are viscerally “engaged” with the object.

The upshot of Smith’s analysis of registration is an affirmation of what he calls “a philosophy of presence”. He writes:

[Its metaphysical viewpoint is] a commitment to One world, a world with no other, a world in which both subjects and objects—we and the things we interact with and think about and eat and build and till and are made of and give away as presents—are accorded appropriate place. ... that world is depicted as one of cosmic and ultimately ineffable particularity.... Neither formally rigid nor nihilistically sloppy, the flux [of particulars] sustains complex processes of *registration*: a form of interaction, subsuming both representation and ontology, in which “s-regions” or subjects stabilize patches of the flux, in part through processes of intervention, adjusting and building them and beating them into shape, and also through patterns of disconnection and long-distance coordination, necessary in order to take the patch to be an object, or more generally, to be something *in and of the world*. [p. 347]

But this book is not merely a metaphysical treatise on the philosophical foundations of computer science. Smith aims to touch the reader on many other planes as well. He warns us early on that:

The story to be told represents an attempt to unify and therefore, with luck to help heal, the schism between the academic-cum-intellectual-cum-technological on the one hand, and the curious, the erotic, the spiritual, the playful, the humane, the moral, the artistic, the political and the sheerly obstreperous on the other. [p. 94]

Although we get intimations about how this might be as the narrative unfolds, it is only in the conclusion that Smith explicitly sketches how a “philosophy of presence” helps to shed light on a wide range of issues: the representation problem in AI (how does a neural net “register” a face?); the realist/constructivist (instrumentalist) debate in philosophy of science (how does a physicist register a neutrino?); the Platonist/intuitionist/constructivist debate in mathematics (how does a mathematician register a number?) etc.

One wonders whether the aesthetics of the book itself is intended to achieve the goal of conveying the many-layered implications of his theory at another “register” in the reader’s consciousness: the layout is original and creative, its structural composition is exceptionally ergonomic and the style of

prose is not always complex and philosophical. Smith often tries to speak-easy and give us the gritty feeling for his intuitions, a welcome break from the often strong academic-cum-intellectual-cum-technological flavour of the remainder of the text. Every other page has an instructive drawing of some sort that illustrates a point and there are many sidebars that take the place of stretched-out parenthetical remarks. Near the end of the book, just as the reader is led into Smith’s metaphysics and conclusion, there is an especially beautiful picture of a canvas by Adam Lowe. It is as though Smith thought that a contemplation of the art-piece by itself could convey the ineffable intuition for his philosophy of presence.

There is a lot to be grateful for in this book. It stimulates not just the intellect but also the spirit and the imagination. It offers a way to understand the stuff of objects and their inextricable ties to mentality in a way that evenly balances reason and logic with intuition and mystery. Perhaps its greatest value is to have rekindled the possibility of conversations about the ineffable, about the inter-relatedness among all things and to have shown how completely relevant these metaphysical discussions are to the science of computing.

Further Reading

- [1] Jerry Fodor, *The Modularity of Mind*, MIT Press 1983.
- [2] Zenon Pylyshyn, *Computation and Cognition*, MIT Press 1984.
- [3] John Haugeland, *Artificial Intelligence: The Very Idea*, MIT Press 1985.
- [4] Patricia Churchland, *Neurophilosophy*, MIT Press 1989.
- [5] Daniel Dennett, *Consciousness Explained*, Boston: Little, Brown 1991.
- [6] John Searl, *The Rediscovery of the Mind*, MIT Press, 1992.
- [7] David Chalmers, *The Concious Mind*, Oxford, 1996.

More on ECONOPHYSICS (the Physics of Money)

Charles Morgan

From our last lecture, you will recall the central principle:

(1) Time is Money.

We have all noted that if a rich person travels from home to a big city like Vancouver, New York, or Tokyo, for business, they are able to make considerably more money than we would if we made the same trip. This observation is frequently phrased as "the rich get richer," or "them that has, gets." Using "del d" for the distance travelled and "del \$" for the change in money, we have:

$$(2) \quad \frac{\text{del } d}{\text{del } \$} (\text{rich}) < \frac{\text{del } d}{\text{del } \$} (\text{poor})$$

since del d is the same for both, but del \$ (rich) > del \$ (poor).

Using (1), we can replace money by time in (2) to obtain:

$$(3) \quad \frac{\text{del } d}{\text{del } t} (\text{rich}) < \frac{\text{del } d}{\text{del } t} (\text{poor})$$

But del d / del t is just velocity. Hence using v_r for the velocity of the rich v_p for the velocity of the poor, we have in general:

$$(4) \quad v_r < v_p$$

Now, for simplicity let us consider an isolated system of two particles. one rich and one poor, orbiting about each other under the influence of mutual gravitational attraction. Since the system is isolated, linear momentum is conserved, which gives:

$$(5) \quad m_r v_r = m_p v_p$$

where m represents mass.

So from (4) and (5) we know that in general:

$$(6) \quad m_r > m_p$$

In other words, the richer you are, the greater your effective mass.

In this isolated system, the two particles orbit around their common center of mass, each at a different radius r from the center, but with the same period of rotation P. Their centripetal forces must balance each other. The equation for

centripetal force in a circular orbit is just:

$$(7) \quad F_c = \frac{4 \pi^2 m r}{P^2}$$

where we use "****" to indicate exponentiation. So, using (7) and setting the centripetal force of the rich equal to the centripetal force of the poor, we can rearrange terms to obtain:

$$(8) \quad \frac{r_r}{r_p} = \frac{m_p}{m_r}$$

Then (6) and (8) together yield:

$$(9) \quad r_r < r_p$$

In other words, the radius of the orbit of the rich is smaller than the radius of the orbit of the poor. The result in (9) allows us to explain two common observations:

- (A) The richer you are, the closer your orbits are to the center of the action.
- (B) The poor are always relegated to the periphery.

Before we end this lesson, we will use (6) to explain several other observations. Suppose a third "test" particle is introduced into the system with arbitrary mass m. Taking the gravitational constant to be G, the mutual gravitational force between the particle of mass m and one of m' is given by:

$$(10) \quad F_g = \frac{G m m'}{d^2}$$

where d is the distance between the two particles. From this equation we see that the force exerted on the new particle is directly proportional to the mass of the influencing particle. So the greater the mass of the influencing particle, the more force it will exert on the test particle. By (6), the richer you are, the greater your effective mass. Hence we have an explanation of the following observation:

- (C) At a fixed distance, the richer person always exerts greater influence.

Now you know why the guy next to you in the pin stripe suit and Gucci shoes always catches the eye of the waiter or the cab driver before you do.

Equation (10) also tells us that the force exerted on the test particle by the influencing particle is inversely proportional to the square of the distance, as well as directly proportional to the mass of the influencing particle. Thus the more massive the influencing particle, the further away it can bring to bear a given unit of force. But again, by (6) we know that the richer you are, the greater your effective mass, which explains the following observation:

- (D) The richer a person is, the larger will be that person's sphere of influence.

It is now time to recall another great principle of ECONOPHYSICS.

- (11) Knowledge is Power.
In any physical system, the amount of work done is just the product of the amount of power applied times the time over which it is applied.

$$(12) \quad \text{Work} = \text{Power} \times \text{Time}$$


From (11) we can substitute Knowledge for Power and from (1) we can substitute Money for Time to get:

$$(13) \quad \text{Work} = \text{Knowledge} \times \text{Money}$$

Solving (13) for Money, we get:

$$(14) \quad \text{Money} = \text{Work} / \text{Knowledge}$$

So, as Knowledge increases, Money decreases, given the same amount of Work. That is, Money is inversely related to Knowledge. In fact, as Knowledge increases to infinity, Money goes to 0, no matter what the value of Work. So the principles of ECONOPHYSICS provide an answer to that perennial question:

- (E) If you're so damn smart, why ain't you rich? 



Knowledge Engineer

Build custom intelligent applications for Fortune 500 companies. Custom development is performed onsite in a professional engagement setting, interacting closely with customers. Development tools include ART* Enterprise, C, and C++, as well as occasional use of Java, HTML, and VB. Extensive travel is required.

ART*Enterprise is the premier development tool for building Web-enabled intelligent applications, and has an unparalleled record for delivering high-performance, high ROI applications. The ART product family success record is the direct result of combining powerful Artificial Intelligence (AI) techniques for representing and automating knowledge, with a well designed and highly scaleable inference engine. ART*Enterprise provides a rich environment for rapid prototyping, object-oriented programming, and quick development of intelligent applications using case-based reasoning and powerful pattern matching rules.

Qualified candidates must be creative, articulate, personable, self-confident and possess a broad base of talents and abilities including programming, presentation, writing, and client management skills. Position requires experience in expert systems, using rule-based languages such as ART, ART-IM,

ART*Enterprise, CLIPS, OPS-5, KEE, Nexpert Object, AION-DS, KBMS; experience in Windows or UNIX/ C++ programming; and strong oral and written communication skills. Prefer experience in RDBMS (Oracle, Sybase). Position requires extensive travel; candidate can live anywhere in the 48 states within reasonable distance of a major airport.

Brightware provides an entrepreneurial environment where you can make a real difference on challenging, high visibility, cutting edge projects. We offer extremely competitive salaries, and an extensive benefits package.

**Leslie Evans, Corporate Recruiter
Brightware, Inc.**

350 Ignacio Blvd.
Novato, CA 94949
v: 415-884-4744 x183
f: 415-884-5622
e: evans@brightware.com
www.brightware.com

Join CSCSI and Receive

Canadian Artificial Intelligence

Did you know that **Canadian Artificial Intelligence** is Canada's only National AI magazine? Recommend membership to a friend or colleague in CSCSI and they too will receive this publication as a benefit of membership. CSCSI members receive reduced rates for CSCSI Conference registration, CSCSI conference tutorials, and on subscriptions to Computational Intelligence and Machine Learning. For more information, please contact a member of the CSCSI executive (*refer to Contents page*) or CIPS.

CSCSI/SCEIO Membership Application

I wish to join CSCSI/SCEIO and receive Canadian Artificial Intelligence

- Web Access Only (\$30.00* Cdn./yr.)
 Printed Copy and Web Access (\$40.00 *Cdn./yr.)

I am a student

- Web Access Only (\$15.00* Cdn./yr.)
 Printed Copy and Web Access (\$15.00* Cdn./yr.)

I am a member of CIPS

- Web Access Only (\$25.00* Cdn./yr.)
 Printed Copy and Web Access (\$30.00* Cdn./yr.)

*Includes Applicable G.S.T.

Name

Mailing

Address

E- mail

Address

Please mail this membership application to:

CIPS
One Yonge St., Suite 2401
Toronto, Ontario
M5E 1E5

Phone: (416) 861-2477

Fax: (416) 368-9972

CALL FOR PAPERS

The Twelfth International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems IEA/AIE-99

Le Meridien Cairo, Cairo, Egypt, May 31- June 3, 1999

Sponsored by:

International Society of Applied Intelligence

Organized in Cooperation with:

AAAI, ACM/SIGART, CSCSI, ECCAI, IEE, INNS, JSAI, SWT

General Chair: M. Ali, SWT U., USA
Program Chairs:
A. El-Dessouki, Egy. Res. Instit., Egypt
I. Imam, Thinking Machines Corp., USA
Y. Kodratoff, U. Sud, France

Local Chair: A. Wahab, ERI, Egypt
Publicity Chair: A. Sheta, ERI, Egypt
Exhibit. Chair: M. Imam, Cairo U., Egy.
Regist. Chair: C. Morriss, SWT U., USA

Program Committee:
B. AboElNasr, Alexandria U., Egypt
F. Anger, NSF, USA
O. Badr, Ain Shams U., Egypt
S. Barro, U. Santiago de Comp, Spain
G. Antoniou, Griffith U., Australia
F. Belli, U. of Paderborn, Germany
K. Chang, Auburn U., USA
L. Chittaro, U. di Udine, Italy
N. Darwish, Cairo U., Egypt
R. Debreceny, Nanyang Tech. U., Singap.
K. DeJong, George Mason U., USA
R. Engels, U. Karlsruhe, Netherlands
I. Farag, Cairo U., Egypt
K. Fischer, DFKI, Germany
S. Fukuda, Tokyo M. Inst. of Tech., Jap.
A. Geneid, American U. in Cairo, Egypt
A. Ghoneimy, Ain Shams U., Egypt
H. Guesgen, Auckland U., New Zealand
N. Hegazi, Egypt. Research Inst., Egypt
K. Kawamura, Vanderbilt U., USA
L. Kerschberg, George Mason U., USA
A. Kondratyev, U. of Aizu, Japan
N. Kushmerick, U. of Washington, USA
G. Lacey, Trin. College Dublin, Ireland
D. Leake, Indiana U., USA
M. Lenz, Humboldt U. Berlin, Germany
M. Matthews, U. South Carolina, USA
R. Modjeski, Florida Inst. of Tech., USA
L. Monostori, Hung. Aca. Sci., Hungary
A. del Pobil, Jaume-I U., Spain
D. Potter, U. of Georgia, USA
G. Prabhakar, Bell Lab., USA
A. Ram, Georgia Institute of Tech., USA
C. Sammut, N. South W. U., Australia
S. Shaheen, Cairo U., Egypt
S. Sin, U. of Tulsa, USA
R. Sun, U. of Alabama, USA
T. Tanaka, Fukuoka Inst. Tech., Japan
J. Treur, VU Amsterdam, Netherlands
Y. Umeda, U. of Tokyo, Japan
G. Widmer, Austr. Resear. Instit., Austria
M. Wooldridge, QMW, UK

IEA/AIE-99 continues the tradition of emphasizing applications of artificial intelligence and expert/knowledge-based systems to engineering and industrial problems. Topics of interest include, but are not limited to:

Automated Problem Solving	Intelligent Agents	Natural Language Processing
Adaptive Control	Intelligent Networks	Neural Networks
CAD/CAM	Intelligent Database	Planning & Scheduling
Case-based Reasoning	Intelligent Interfaces	Practical Applications
Computer Vision	Intelligent Tutoring	Reasoning Under Uncertainty
Connectionist Models	KBS Methodologies	Robotics
Data Mining	Knowledge Acquisition	Sensor Fusion
Distributed AI Architecture	Knowledge Discovery	Spatial & Temporal Reasoning
Expert Systems	Knowledge Representation	Speech Recognition
Fuzzy Logic	Machine Learning	System Integration Tools
Genetic Algorithms	Model-based Reasoning	Verification & Validation
Heuristic Searching		

Authors are invited to submit 1) a URL address containing a printable version of their paper; 2) a list of key words (some sample topics are listed above); 3) no more than one page (by email) describing: a) the contribution, b) the significance, and c) the results of the presented work; and 4) one hard copy of their paper to the Program Co-chair at the address given below no later than October 31, 1998. Also, the conference will contain a series of special track sessions. If an author desires to include his/her paper in a special track, please indicate that as well. A list of special tracks will be announced through the URL:

<http://mason.gmu.edu/~iimam/ieaaie99/ieaaie99.html>

Papers must be written in English, have up to 10 single-spaced pages, presenting the results of original research or innovative practical applications relevant to the scope of the conference. Practical experiences with state-of-the-art AI methodologies are also acceptable when they reflect lessons of unique value to the conference attendees. Short works, up to 6 pages, to be presented in 10 minutes, may be submitted as SHORT PAPERS representing work in progress or suggesting possible research directions. (Please, indicate "short paper" in the submission letter in this case). Submissions should be completed on the Internet and the URL addresses along with a hard copy should be received by the Program Co-Chair by October 31, 1998. Notification of the review process will be made by January 22, 1999, and the final copies of papers will be due for inclusion in the conference proceedings by February 20, 1999. Referees will be asked to nominate papers for a Best Paper Prize to be announced at the conference. All papers, but particularly those nominated for the Best Paper competition, will be automatically considered for a place in the International Journal of Applied Intelligence. General conference information can be sought from the General Chair at the following address.

Dr. Moonis Ali
General Chair of IEA/AIE-99
Southwest Texas State University
Department of Computer Science
601 University Drive,
San Marcos TX 78666-4616 USA
Email: ma04@swt.edu

Dr. Ibrahim Imam
Program Co-Chair of IEA/AIE-99
Thinking Machines Corporation
16 New England Executive Park
Burlington, MA 01803 USA
Email: ifi@think.com

Please keep me on the mailing list of the IEA/AIE-99. Complete and send to Dr. Ali at the above address

Name: _____ email: _____ Telephone: _____
Address: _____ Fax: _____

Intend to submit a paper (y/n): []

Send me registration information (y/n): []