Proc. of WoLLIC'2000

August 15–18, 2000, Natal (RN), Brazil

Contents

Editorial 1

I Tutorial Lectures **3**

Tutorial: The Optimal Implementation of Functional Programming Languages (abstract) 5
A. Asperti

Tutorial: Definability, Measure and Randomized Algorithm (abstract) 7
A. Macintyre

Tutorial: The complexity of propositional proofs and connections to bounded arithmetic and computational complexity (abstract) 9
T. Pitassi

Tutorial: Classes de complexité definie en termes d'élimination des quanteurs / Complexity classes associated to quantifiers elimination (abstract) 11
B. Poizat

Tutorial: Presheaf Models for Concurrency (abstract) 13
G. Winskel

II Invited Papers **15**

XML and the Hypertextual Electronic Library of Mathematics (abstract) 17
A. Asperti

Definability, Measure and Randomized Algorithm (abstract) 19
A. Macintyre

Translations and Normalization Procedures (abstract) 21

L. C. Pereira

A New Proof of the Weak Pigeonhole Principle (abstract) 25
T. Pitassi

Une tentative malheureuse de construction d'une structure eliminant rapidement les quanteurs / An unfortunate attempt to build a structure with fast quantifier elimination (abstract) 27
B. Poizat

Linearity in Distributed Computation (abstract) 29
G. Winskel

III Contributed Papers 31

On the structure of normal λ-terms having a certain type 33
S. Broda and L. Damas

Approximate Belief Revision - A Preliminary Report 45
S. Chopra and R. Parikh and R. Wassermann

A Model Theoretic Approach to Translations Between Logics 55
M.E. Coniglio and W.A. Carnielli

Elementary number theory in weak fragments of arithmetic (abstract) 67
P. D'Aquino

Reasoning about Dynamic Databases: Tableaux for Single-Step-Update Models 69
C.G. Fermüller

Infinite SLaKE-Tableaux 79
M. Finger

Declarative Semantics for Quantified Disjunctive Programs 89
D. Guller

Lambek Grammars as Combinatory Categorial Grammars 103
G. Jäger

Judging Unresolved Utterances in Context 111
P. Krause

Completeness of an Action Logic Featuring a δ-Operator for Timed Transition Systems 123
F. N. Amaral and E. H. Haeusler

A Fibring Semantics for the Semantic-Morphological Interface in Natural Language 133
R. Naumann and A. Latrouite

A New Spectrum of Recursive Models 147
A. Nies

Intuitionistic Finitely Many-Valued Sequent Calculi 155
E. Reznik and Ph. Curmin

The Largest Cartesian Closed Category of Domains, Considered Constructively 169
D. Spreen

Towards Decision of Testing Equivalence for Petri Nets with Dense Time 181
I. Virbitskaite

A Gradient-Based Boosting Algorithm for Regression Problems 191
R. Zemel and T. Pitassi

Towards the Mechanical Verification of Textbook Proofs 201
C. Zinn

Editorial

WoLLIC'2000, being held at Hotel Barreira Roxa, Natal, Rio Grande do Norte, Brazil, is the seventh version of a series of workshops which started in 1994 with the aim of fostering interdisciplinary research in logic, language, information and computation. Previous versions were held at: Recife (Pernambuco, Brazil) in 1994 and 1995; Salvador (Bahia, Brazil) in 1996; Fortaleza (Ceará, Brazil) in 1997; São Paulo (Brazil) in 1998; Itatiaia (Rio de Janeiro, Brazil) in 1999.

Scientific sponsorship comes from the *Interest Group in Pure and Applied Logics* (*IGPL*), the *European Association for Logic, Language and Information* (*FoLLI*), the *Association for Symbolic Logic* (*ASL*), the *Sociedade Brasileira de Computação* (*SBC*), and the *Sociedade Brasileira de Lógica* (*SBL*).

Funding was kindly given by:
(i) CNPq (*Conselho Nacional de Desenvolvimento Científico e Tecnológico*, the scientific and technological development council of the Brazilian *Ministério da Ciência e Tecnologia*);
(ii) CAPES (*Fundação Coordenação de Apoio ao Aperfeiçoamento de Pessoal de Nível Superior*, a Foundation for the Development of Higher-Education under the Brazilian *Ministério da Educação e do Desporto*);
(iii) Fundação Norte-Rio-Granensse de Pesquisa e Cultura (FUNPEC);
(iv) Pró-Reitoria de Extensão, Univ. Fed. do Rio Grande do Norte (PROEX–UFRN).

Programme Committee:
Sergei Artemov (Moscow Univ, Russia, and Cornell Univ, USA),
Ricardo Bianconi (Univ São Paulo, Brazil),
Sam Buss (Univ California San Diego, USA),
Edmund Clarke (Carnegie-Mellon Univ, USA),
Itala D'Ottaviano (Univ Campinas, Brazil),
Heinz-Dieter Ebbinghaus (Univ Freiburg, Germany),
Peter Johnstone (Cambridge Univ, UK),
Hans Kamp (Univ Stuttgart, Germany),
Maarten de Rijke (Amsterdam Univ, The Netherlands),
Colin Stirling (Edinburgh Univ, Scotland).

Organising Committee:
B. C. Bedregal (UFRN), M. E. Coniglio (UNICAMP), A. M. P. Cruz (UFRN), D. Deharbe (UFRN), A. T. C. Martins (UFC), A. Moreira (UFRN), A. G. de Oliveira (UFPE/UFBA), R. de Queiroz (UFPE) (Co-Chair), R. H. N. Santiago (UFRN).

Presentations: Apart from the contributed papers, the programme includes 5 tutorial lectures on the first day of the workshop, as well as 6 invited talks.

Tutorial lectures:

1. ***The Optimal Implementation of Functional Programming Languages***, by Andrea Asperti
2. ***Definability, Measure and Randomized Algorithm***, by Angus Macintyre

3. ***The complexity of propositional proofs and connections to bounded arithmetic and computational complexity***,
   by Toniann Pitassi
4. ***Complexity Classes Associated to Quantifiers Elimination***,
   by Bruno Poizat
5. ***Presheaf Models for Concurrency***,
   by Glynn Winskel.

Contributed papers: The programme committee received 21 (twenty-one) papers of high quality, from all over the world: Germany: 4, Brazil: 3, USA: 3, France: 2, Argentina: 1, Austria: 1, China: 1, Israel: 1, Mexico: 1, Portugal: 1, Russia: 1, Scotland: 1, Slovakia: 1.

Each submitted paper was made anonymous and sent to 2 referees for evaluation. The referees gave 0-10 ratings to: (i) overall quality; (ii) soundness; (iii) originality; (iv) relevance to the workshop; (v) presentation, in this order of priority.

We should like to wish you all a profitable and enjoyable meeting!

Sincerely,

Ruy J.G.B. de Queiroz
Benjamín C. Bedregal
(On behalf of the Organising Committee)

Part I

Tutorial Lectures

Tutorial: The Optimal Implementation of Functional Programming Languages (abstract)

ANDREA ASPERTI, *Department of Computer Science, Bologna University, Italy. E-mail: asperti@cs.unibo.it*

Abstract

In foundational research of almost thirty years ago, Jean-Jacques Lévy attempted to characterize formally what an optimally efficient reduction strategy for the λ-calculus would look like, in the sense of avoiding useless duplication of work along the reduction.

In elaborating his notion of optimal reduction, Lévy introduced a labelled variant of λ-calculus where all subterms of an expression are annotated with labels that code the history of a computation. He proposed the idea of a "redex family" – redexes in a term with identical labels – to identify similar redexes whose reduction should somehow be evaluated at once (via a so-called parallel β-reduction) by an "optimal" reduction technique.

Recent research, starting from the pioneering work of Lamping, has shown that there indeed exist λ-calculus evaluators satisfying Lévy's specification. This requires a beautiful graph reduction technology (sharing graphs) for sharing evaluation contexts dual to the sharing of values. Sharing graphs (that are a particular kind of Lafont's Interaction Nets) allow a single, shared representation of a redex family, and thus provide a means to implement Lévy's notion of parallel reduction.

The tutorial is a simple introduction to optimality, the technology of sharing graph, its semantics, and some complexity issues.

Tutorial: Definability, Measure and Randomized Algorithm (abstract)

ANGUS MACINTYRE, *Department of Mathematics and Statistics, Edinburgh University, Mayfield Road, Edinburgh, Scotland.*
E-mail: angus@maths.ed.ac.uk

Some of the most important work in the Robinson tradition of model theory involves the connection between definitions and the geometric or topological structure of the sets thereby defined. The most notable examples involve real or p-adic structure, particularly in the recent work,initiated by Wilkie, on o-minimal structures. In those situations, not only are the sets defined locally closed, and so measurable, but they satisfy a very strong uniform Law of Large Numbers (Finiteness of Vapnik–Chervonenkis Dimension). In certain cases, this has implications for very fast randomized algorithms, and even for the mathematics of learning on sigmoidal neural networks. The tutorial will cover the basic theory, and the most notable examples.

Tutorial: The complexity of propositional proofs and connections to bounded arithmetic and computational complexity (abstract)

TONIANN PITASSI, *Department of Computer Science, University of Toronto, Canada. E-mail: toni@cs.toronto.edu*

Abstract

Proof complexity, the study of the lengths of proofs in propositional logic, is an area of study that is fundamentally connected both to major open questions of computational complexity theory and to practical properties of automated theorem provers. In the last decade, there have been a number of significant advances in proof complexity lower bounds. Moreover, new connections between proof complexity and circuit complexity have been uncovered, and the interplay between these two areas has become quite rich. In addition, attempts to extend existing lower bounds to ever to extend existing lower bounds to ever stronger systems of proof stronger proof systems have spurred the introduction of new and interesting proof systems, adding both to the practical aspects of proof complexity as well as to a rich theory. In this talk, I will survey these developments and lay out the major open problems in the area.

Tutorial: Classes de complexité definie en termes d'élimination des quanteurs / Complexity classes associated to quantifiers elimination (abstract)

BRUNO POIZAT, *Institut Girard Desargues, Université Claude Bernard, Lyon-1, France. E-mail: poizat@desargues.univ-lyon1.fr*

Abstract

1. P = ? NP vu comme un problème de decision ; P = ? NP vu comme un problème d'élimination.

2. La complexité algébrique : calculs dans un corps (fini, réel-clos, algébriquement clos), ou dans une structure quelconque ; le selecteur ; formules, circuits, programmes en ligne, arbres de decision, arbres de calcul.

3. Les classes P, NBP et NP, uniformes ou non ; conventions sur les parametres.

4. Comparaison des structures : le role des parametres ; cas où on peut les eliminer.

5. Comparaison des structures : cas ou l'algorithmie d'une structure se ramène à un problème classique.

(In English)

Complexity classes associated to quantifiers elimination.

1. P = ? NP as a decision problem ; P = ? NP as an elimination problem.

2. Algebraic complexity : computations in a (finite, real-closed, algebraically closed) field, or in an arbitrary structure ; the selector ; formulae, circuits, straight line programmes, decision trees, computation trees.

3. The classes P , NBP and NP , uniforms or not ; conventions on the parameters.

4. Comparison of structures : role of the parameters ; cases where they can be eliminated.

5. Comparison of structures : cases where the algorithmic of a structure is reducible to a classical problem.

References

Lenore BLUM, Felipe CUCKER, Michael SHUB & Steve SMALE, *Complexity and real computations*, Springer.

Bruno POIZAT, *Les petits cailloux*, Nur al-Mantiq wal-Ma'rifah; Aléas Editeur, 15 quai Lassagne, 69001 Lyon, France.

http://desargues.univ-lyon1.fr/ (Thème Logique)

http://www.ens-lyon.fr/LIP/ (Hervé Fournier, Pascal Koiran, Natacha Portier)

Tutorial: Presheaf Models for Concurrency (abstract)

GLYNN WINSKEL, *BRICS, Aarhaus University, Denmark.*
E-mail: gwinskel@brics.dk

Abstract

The tutorial will introduce and motivate a model of concurrent computation, a form of domain theory, in which nondeterministic processes are represented by presheaves. In it bisimulation, an important notion of equivalence between nondeterministic processes, is obtained from "open" maps. I intend to provide notes on the relevant mathematics. The tutorial backs-up the talk.

Part II

Invited Papers

XML and the Hypertextual Electronic Library of Mathematics (abstract)

ANDREA ASPERTI, *Department of Computer Science, Bologna University, Italy. E-mail: asperti@cs.unibo.it*

Abstract

There is a compelling need of integration between the current tools for automation of formal reasoning and mechanization of mathematics (proof assistants and logical frameworks) and the most recent technologies for the development of web applications and electronic publishing. We explain the pivotal role that should be played by the Extensible Markup Language in this integration process, and how the many different pieces of technology under development at World Wide Web Consortium (W3C) should naturally fit together for the creation and maintenance of a virtual, distributed, hypertextual library of formal mathematical knowledge.

Definability, Measure and Randomized Algorithm (abstract)

ANGUS MACINTYRE, *Department of Mathematics and Statistics, Edinburgh University, Mayfield Road, Edinburgh, Scotland.*
E-mail: angus@maths.ed.ac.uk

Some of the most important work in the Robinson tradition of model theory involves the connection between definitions and the geometric or topological structure of the sets thereby defined. The most notable examples involve real or p-adic structure, particularly in the recent work,initiated by Wilkie, on o-minimal structures. In those situations, not only are the sets defined locally closed, and so measurable, but they satisfy a very strong uniform Law of Large Numbers (Finiteness of Vapnik–Chervonenkis Dimension). In certain cases, this has implications for very fast randomized algorithms, and even for the mathematics of learning on sigmoidal neural networks. The tutorial will cover the basic theory, and the most notable examples.

Translations and Normalization Procedures (abstract)

LUIZ CARLOS PEREIRA, *Departamento de Filosofia, Pontifícia Universidade Católica do Rio de Janeiro, Brazil.*
E-mail: luiz@inf.puc-rio.br

Abstract

The aim of the present paper is to explore some interesting connections between translations among logics and different normalization procedures [1].

The traditional family of double-negation translations from Classical Logic into Intuitionistic Logic (Gentzen–Gödel–Kolmogorov) were defined and studied in the context of foundational matters. Gentzen, for example, explicitly considered as one of the main consequences of his translation procedure, the reduction of the consistency problem of classical arithmetic to that of intuitionistic arithmetic. The main ideas behind these translations are:

(1) Introduce a double negation in front of every atomic formula (in the Kolmogorov-variant, in front of every part of the formula);
(2) Replace some key logical operators by adequate classical equivalents (Gödel-Gentzen variants).

The crucial property that makes these translation work is *stability*. A formula A is called *stable* if and only if $(A \leftrightarrow \neg\neg A)$. The desired results about translations then follow from the fact that formulas in appropriate language fragments of intuitionistic logic are stable (*negative* formulas, for example).

It can be easily verified that this family of translations (sometimes called *negative translations*) is proof-theoretically justified by the normalization procedure for classical logic used by Prawitz (see [Prawitz65]). Prawitz shows, as a preliminary result to normalization, that every application of the classical absurd rule in the fragment $\{\wedge, \rightarrow, \perp, \forall\}$ of the natural deduction system C can be reduced to an atomic application. Prawitz defines a set of operations (reductions) whose aim is to transform non-atomic applications of the classical absurd (in the fragment considered) into simpler ones. For example, the derivation

$$\dfrac{\begin{array}{c}[\neg(A \wedge B)]\\ \vdots \\ \perp\end{array}}{(A \wedge B)}$$

[1] A similar proof-theoretical approach to translations was used by Prawitz and Malmnäs in their seminal paper [PrawMal68]

reduces to

$$\dfrac{\dfrac{\begin{array}{c}\dfrac{\dfrac{\dfrac{[(A \wedge B)]}{A} \quad [\neg A]}{\bot}}{\neg(A \wedge B)} \\ \vdots \\ \bot\end{array}}{A} \qquad \dfrac{\begin{array}{c}\dfrac{\dfrac{\dfrac{[(A \wedge B)]}{B} \quad [\neg B]}{\bot}}{\neg(A \wedge B)} \\ \vdots \\ \bot\end{array}}{B}}{(A \wedge B)}$$

The two new applications of the classical absurd rule are obviously simpler than the original one.

The moral is: *in the given fragment, classical reasoning can be restricted to atomic formulas.* We can easily see now that everything is ready for the Gentzen-Gödel translations: we define disjunction and the existential quantifier (and implication in Gödel's translation), and double negation can be restricted to atomic formulas.

In 1989, J. Seldin presented a new normalization strategy where he showed that every derivation in the fragment $\{\wedge, \vee, \rightarrow, \bot, \exists\}$ could be transformed into a derivation with at most one application of the classical absurd rule (see [Seldin89]). Moreover, if this application occurred, it would be the last rule in the resulting derivation. Seldin defined new reduction procedures that allow us to permute applications of the classical absurd rule downwards. For example, the derivation

$$\dfrac{\dfrac{\begin{array}{c}[\neg A] \\ \vdots \\ \bot\end{array}}{A}}{(A \vee B)}$$

reduces to

$$\dfrac{\begin{array}{c}\dfrac{\dfrac{\dfrac{[A]}{(A \vee B)} \quad \neg(A \vee B)}{\bot}}{\neg A} \\ \vdots \\ \bot\end{array}}{(A \vee B)}$$

The moral now is: *in this new fragment, classical reasoning, if used at all, is used as the last step of an argument.* Every classical derivation Π in the fragment can now be put in this very special form constituted of two parts: an intuitionistic part and a classical part (possibly empty) constituted of a single application of the classical absurd rule. Normalization for Classical Logic is now reduced to normalization for Intuitionistic Logic.

This normalization procedure clearly gives rise (1) to a new translation from classical logic into intuitionistic logic, and (2) to an extension of Glivenko's theorem to first order logic. In the case of classical propositional logic, the translation function Tr would be simply: $Tr(A) = \neg\neg A$.[2]

In the second part of the paper, we extend this proof-theoretical analysis to translations from intuitionistic logic into classical linear logic. We present a new single-conclusion Natural Deduction system [3] for the $\{\otimes, \perp, \multimap, \oplus, !\}$-fragment of classical linear logic, and we prove the normalization theorem for this system. The system is based on an *involution rule* defined as follows:

$$\frac{\begin{array}{c}[A^{\perp}]\\ \vdots \\ \perp\end{array}}{A}\,inv$$

We also discuss the extension of *semantical proofs* of st rong normalizability (based on *reducibility predicates*) to the classical linear system based on the involution rule.

Although, as we have already said, traditional translations were introduced within the framework of foundational questions, it is certainly true that the interest on translations is not restricted to this kind of question. In the final part of the paper, we discuss the translation proposed by P. Krauss [Krauss95] and its relation to some general conceptual aspects of the concept of *translation*. In particular, we discuss the fact that translations that introduce double negations at the atomic level are not in general *functorial*, i.e., they do not commute with substitution [4].

This is joint work with Maria da Paz N. Medeiros.

References

[Dragalin88] Dragalin, A.G., Intuitionism - Introduction to Proof Theory, *American Mathematical Society*, 1988.

[Gentzen33] Gentzen, Gerhard, Über das Verhältnis zwischen intuitionistischer und klassischer Logik, in *Archiv für Mathematische Logik und Grundlagenforschung* 16 (1974).

[Gentzen35] Gentzen, Gerhard, Untersuchungen über das Logische Schliessen, in *Matematische Zeitschrift*, vol.39, pp.176–210, 405–431, 1935. English translation in Szabo [10].

[Girard71] Girard, J.Y., Une extension de l'Interpretation de Gödel a l'Analyse et son application a l'elimination des coupures dans l'analyse et la theorie des types, in *Proccedings of the Second Scandinavian Logic Symposium*, ed. by J.E.Fenstad, North-Holland, Amsterdam, 1971.

[Godel33a] Gödel, K., Eine Interpretation des Intuitionistischen Aussagenkalküs, in *Ergebnisse eines mathematischen Kolloquiums*, 4, 1933.

[Godel33b] Gödel, K., Zur intuitionistischen Arithmetik und Zahlentheorie, in *Ergebnisse eines mathematischen Kolloquiums*, 4, 1933.

[Kolmogorov25] Kolmogorov, R.A., On the Principle of the Excluded Middle, translation in van Heijenoort (1967).

[Krauss95] Krauss, P.H., Hypothetical Extensions of Constructive Mathematics, in *The Foundational Studies*, ed. by DePauli-Schimanovich et al., Kluwer, 1995.

---

[2] Glivenko's theorem is a trivial consequence of Seldin's normalization strategy

[3] A similar system was independently defined and studied in [Maraist99]

[4] We would like to thank Per Martin-Löf for calling our attention to this fact

[Maraist99] Maraist, J., Classical linear natural deduction and the linear $\lambda_{\triangle}$ calculus: a preliminary report, preliminary draft, 1999.

[Pereira99] Pereira, L.C., Translations and Normalization: applications to Linear Logic, abstract in *Linear Logic and Applications - Seminar-report*, Dagstuhl, 1999.

[Pottinger77] Pottinger, G., Normalization as a Homomorphic Image of Cut-Elimination, in *Annals of Mathematical Logic*, vol.12, n.3, 1977.

[Prawitz65] Prawitz, Dag, *Natural Deduction*, Almqvist and Wiksell, Stockholm, 1965.

[Prawitz71] Prawitz, Dag, Ideas and Results in Proof Theory, in *Proccedings of the Second Scandinavian Logic Symposium*, ed by J.E.Fenstad , North-Holland, Amsterdam, 1971.

[PrawMal68] Prawitz, Dag and Malmnäs, P.-E., A survey of some connections between classical, intuitionistic and minimal logic, in *Contributions to Mathematical Logic*, eds. A. Schmidt, K. Schütte and H.J. Thiele, North-Holland, 1968.

[Seldin86] Seldin, J. On the proof-theory of the intermediate logic MH, *JSL* 51, pp. 626–647, 1986.

[Seldin89] Seldin, J. Normalization and Excluded Middle I, in *Studia Logica* 48, pp. 193–217, 1989.

[Szabo69] Szabo, M.E., *The Collected Papers of Gerhard Gentzen*, North-Holland, Amsterdam, 1969.

[Troelstra96] Troelstra, A.S. and Schwichtemberg, H., *Basic Proof Theory*, CUP, Cambridge, 1996.

[vanHeijenoort67] , van Heijenoort, J., *From Frege to Gödel*, Harvard University Press, 1967.

A New Proof of the Weak Pigeonhole Principle (abstract)

TONIANN PITASSI, *Department of Computer Science, University of Toronto, Canada. E-mail: toni@cs.toronto.edu*

Abstract

The exact complexity of the weak pigeonhole principle is an old and fundamental problem in proof complexity. It is used in all areas of mathematics, and has been the primary hard example underlying lower bounds in propositional proof complexity.

Still, the exact complexity of the pigeonhole principle is not known, and is linked to several important problems in proof complexity and circuit complexity. Using a clever diagonalization argument, Paris, Wilkie and Woods present quasipolynomial-size bounded-depth proofs of the weak pigeonhole principle. Their argument was further refined by Krajicek. In this talk, we present a new proof: we show that the weak pigeonhole principle has quasipolynomial-size proofs where every formula consists of a single AND/OR of polylog fanin. Our proof is conceptually simpler than previous arguments and is optimal with respect to depth.

This is joint work with Alexis Maciel and Alan Woods.

Une tentative malheureuse de construction d'une structure eliminant rapidement les quanteurs / An unfortunate attempt to build a structure with fast quantifier elimination (abstract)

BRUNO POIZAT, *Institut Girard Desargues, Université Claude Bernard, Lyon-1, France. E-mail: poizat@desargues.univ-lyon1.fr*

Abstract

Pour illustrer le court cours, je tenterai de construire une structure M éliminant rapidement les quanteurs, dans un langage fini: à toute formule existentielle $(\exists y_1,\ldots,y_n)f(x_1,\ldots,x_m,y_1,\ldots,y_n)$, où f est sans quanteurs, doit être associée une formule $g(x_1,\ldots,x_m)$, également libre de quanteurs, qui lui soit equivalente dans M, la longueur de g étant polynomialement bornée en fonction de celle de f. Tant qu'à faire, j'aimerais aussi que g fut calculée par un algorithme polynomial à partir de f. Pourquoi je veux faire ça? Parce que si c'etait vrai de la structure a deux elements $M = \{0,1\}$, dans le langage reduit a la seule relation $x = 0$, eh bien on repondrait positivement à deux questions ouvertes en complexité, $NC_1 = P = NP$; la même chose se produirait si la structure etait finie, hormis le cas trivial d'une structure a un element que nous excluons, ou bien si son langage fini ne comportait que des relations et pas de fonctions. Cette relativisation quelque peu artificielle du problème $P =?NP$ ne semble pas si aisée, et je n'arrive seulement qu'à construire une structure qui n'a cette propriété que pour les formules existentielles dans lesquelles ne survit qu'une seule variable libre. Celà vient de ce que je me restreins à un langage qui ne comporte que des fonctions et des predicats unaires, dont l'avantage est de présenter une théorie des modèles tres rudimentaire; il est possible que la méthode que j'emploie, qui consiste à introduire un predicat de verité, puisse se géneraliser pour emporter le morceau, mais ça devrait être substantiellement plus dur.

(In English)

As an illustration of the tutorial, I will make an attempt to build a structure M with fast quantifier elimination, in a finite language: to each existential formula $(\exists y_1,\ldots,y_n)f(x_1,\ldots,x_m,y_1,\ldots,y_n)$, where f is quantifier free, should be associated an equivalent in M free formula $g(x_1,\ldots,x_m)$, the length of g being polynomially bounded in function of the length f. It would even be better if g could be calculated from f by a polynomial algorithm. Why do that? Because if the thing were

true of the two-element structure $M = \{0, 1\}$, in the language of the only relation $x = 0$, this would give a positive answer to two open questions in Complexity theory, $NC_1 = P = NP$; the same would hold in the case of a finite M, except in the trivial case of a one-element structure, or if the language is purely relationnal. This somehow artificial relativisation of the $P =?NP$ problem seems to be not totally straightforward, and all what I can do is to produce a structure which has this property only when there remains only one surviving free variable in the existential formula. This comes from the fact that I need an extremely rudimentary model-theoretic context, which is provided by a language containing only unary functions and predicates; I see nothing forbiding the extension of the method, which consists in the introduction of a truth predicate, to settle the question, but I assume that it will be considerably harder.

Linearity in Distributed Computation (abstract)

GLYNN WINSKEL, *BRICS, Aarhaus University, Denmark.*
E-mail: gwinskel@brics.dk

Abstract

Gian Luca Cattani and I have worked on a path-based domain theory for interacting processes, one aim being a general theory of bisimulation, also for higher-order process languages. The category of domains forms a model of Girard's linear logic. But what does the linearity mean in the context of interacting processes, and what are its consequences? This talk will explain, and indicate how widespread linearity is by showing that the operations of a range of process calculi can be expressed within a linear metalanguage, designed around the domain theory. Roughly, linearity is about how to manage without a presumed ability to copy. With hindsight it is not surprising that the copying of processes is limited in the context of distributed computation, either as a fact of life, often because remote networks are simply too complicated to have control over, or deliberately, as is the case in the design of cryptographic protocols. Of course, code can be sent explicitly to be copied and the domain theory allows for different possibilities for copying processes.

Part III

Contributed Papers

On the structure of normal λ-terms having a certain type

SABINE BRODA and LUIS DAMAS, *DCC & LIACC, Universidade do Porto, R. do Campo Alegre 823, 4150-180 Porto, Portugal.*
E-mail: {sbb,luis}@ncc.up.pt

Abstract

We define an alternative representation for types in the simply typed λ-calculus or equivalently for formulas in the implicational fragment of intuitionist propositional logic, which gives us a better insight on the nature of a types structure and its relation to the structure of the set of its normal inhabitants. Based on this representation of a type φ, we define the notion of a valid proof tree. Any such valid proof tree represents a finite set of normal inhabitants of φ and every normal inhabitant corresponds to exactly one valid proof tree, constructable with the primitive parts in the formula tree of φ. Precise algorithms are given to establish this relation. Finally, we give a simple characterisation of the proof trees that represent principal normal inhabitants of a type φ.

1 Introduction

Typed versions of systems of λ-calculus have been studied since their appearance in the beginning of this century in [6] and [5], due to their importance to areas of mathematical logic and more recently to computer science. One of the fields with direct application is the area of functional programming (e.g. programming languages ML, Miranda or Haskell). The concept of functional programming relies on the representation of computable functions/algorithms by λ-terms and the execution of programs corresponds to the reduction of λ-terms. Type assignment to λ-terms provides thus for partial specification of the represented algorithms and can be used to show partial correctness of programs. Besides that, types are used to improve the efficiency of the compilation of functional programs. Almost all practical systems have the system $\mathbf{T}A_\lambda$ of simply typed λ-calculus at their basis. On the other hand, there exists a direct correspondence (cf. [11]), via Curry-Howard isomorphism, between $\mathbf{TA}_\lambda$ and the implicational fragment of intuitionistic propositional logic. In fact, a type φ can by assigned to some λ-term if and only if φ is a provable formula/theorem. Furthermore, every (normal) term to which φ can be assigned, i.e. every (normal) inhabitant of φ, represents a (normal) proof of φ in the natural deduction system for the implicational fragment of intuitionistic logic.

Thus, it is natural to seek for algorithms which, for a given type φ, - decide if φ has any normal inhabitant, - decide if the number of normal inhabitants of φ is finite or infinite, - or compute the set of all normal (or even principal normal) inhabitants of φ. Decision-procedures for intuitionistic propositional logic have been known for a long time (e.g. [8], [7]) and the corresponding decision problem, as well as the infiniteness problem of the set of normal inhabitants have been shown to be polynomial-space complete, respectively by Statman in [12] and Hirokawa in [10]. An algorithm, also described in [9], to count (which is the difficult part) and list the normal inhabitants of a type φ has been given by Ben-Yelles in [2] and a similar algorithm for principal normal inhabitants was first presented in [3]. Both these algorithms are based on the generation of terms whose depth (a measure on the structure of a λ-term in β-normal form) does not exceed some limit which can be computed from φ. This fact makes both algorithms extremely inefficient. On the other hand, it seems to be very unsatisfactory that the algorithms mentioned do not take into account any information about φ, besides the number of type variables and of their occurrences. In fact, as pointed out by Hindley in [9], when we ask for the number of normal inhabitants [1] of a type, the answer is often finite and interesting patterns show up which are still not completely understood. The situation is even worse for principal inhabitants, since we do not know any "non-exhaustive" method for computing a principal normal inhabitant of a type φ.

The present paper is an attempt towards a better understanding of the nature of the structure of a type and its relation to the structure of the set of its normal inhabitants. In section 2 we define an alternative tree-like representation for a type φ, which we call its formula tree. The nodes of such formula tree consist of primitive parts which, following the rules given in section 3, can be combined to form valid proof trees. Any such a valid proof tree represents a finite set of normal inhabitants of φ and every normal inhabitant corresponds to exactly one valid proof tree, constructable with the primitive parts in the formula tree of φ. In fact, in section 4 we give precise algorithms to establish this relation. The first algorithm computes a valid proof tree for any [2] given normal inhabitant of a type φ, while the second "inverse" algorithm computes the set of all normal inhabitants which give origin to a given valid proof tree of φ. In section 5 we give a simple characterisation of the proof trees that represent principal normal inhabitants of a type φ. Finally, in section 6, we illustrate the use of our approach with another example.

2 The formula tree of a type

We assume familiarity with the basic notions in λ-calculus and use standard notation from [1] and [9]. Our notation differs from that in [1], since we denote type-variables (atoms) by "a,b,c,..."and arbitrary types by lower-case Greek letters. For type assignment we consider the system $\mathbf{TA}_\lambda$ of simply typed λ-calculus à la Curry (for an introduction see [9]) or [1], section 3.1).

In this section we define an alternative tree-like representation for a type φ, which we call its formula tree, denoted by $\mathtt{tree}(\varphi)$, and whose nodes consist of primitive parts of either one of the following forms.

(p1)

$$\begin{array}{c} | \\ a \end{array}$$

[1] Note that the number of (not necessarily normal) inhabitants of a type is always zero or infinite.

[2] More precisely, we define the algorithms for a subclass of normal inhabitants, called long normal inhabitants. But as will be argued, both algorithms can be trivially completed to apply to normal inhabitant in general.

(p2)

$$\begin{array}{ccc} & a & \\ a_1 & \cdots & a_n \end{array} \qquad (n \geq 1)$$

(p3)

$$\begin{array}{c} a \\ | \end{array}$$

DEFINITION 2.1
A tree with primitive parts as nodes is a formula tree iff

- the root of the tree is of form (p1);
- every internal node of the tree is of form (p2);
- every leaf of the tree is of form (p2) or (p3).

The following algorithm computes the formula tree $\texttt{tree}(\varphi)$ of a type φ. We will use dashed lines for the edges of the formula tree in order to distinguish them from the edges in the primitive parts (nodes) of the tree. Note that every type φ can be written uniquely in the form $\varphi = \alpha_1 \to \ldots \to \alpha_n \to a$, where a is an atom and $n \geq 0$. The algorithm is given by the following.

- If $n = 0$, i.e. $\varphi \equiv a$, then

$$\texttt{tree}(\varphi) = \begin{array}{c} | \\ a \end{array}$$

- If $n \geq 1$, then

$$\texttt{tree}(\varphi) = \begin{array}{ccc} & | & \\ & a & \\ \texttt{t}(\alpha_1) & \cdots & \texttt{t}(\alpha_n) \end{array}$$

where

$$\texttt{t}(b) = \begin{array}{c} b \\ | \end{array}$$

and finally for $k \geq 1$ and $m_1, \ldots, m_k \geq 0$ take
$\texttt{t}((\beta_{11} \to \ldots \to \beta_{1m_1} \to b_1) \to \ldots \to (\beta_{k1} \to \ldots \to \beta_{km_k} \to b_k) \to b) =$

$$\begin{array}{ccccccc} & & & b & & & \\ & b_1 & & \cdots & & b_k & \\ \texttt{t}(\beta_{11}) & \cdots & \texttt{t}(\beta_{1m_1}) & \cdots & \texttt{t}(\beta_{k1}) & \cdots & \texttt{t}(\beta_{km_k}) \end{array}$$

Note that one can easily define an inverse algorithm, which given a formula tree `FT` computes the unique type φ such that $\texttt{tree}(\varphi) = \texttt{FT}$.

DEFINITION 2.2
The indexed counterpart $\varphi_{\mathtt{i}}$ of a type φ is obtained by successively indexing all occurrences of type variables in φ.

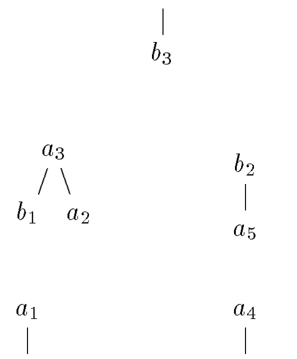
EXAMPLE 2.3
Consider the formula

$$\theta \equiv ((a \to b) \to a \to a) \to ((a \to a) \to b) \to b.$$

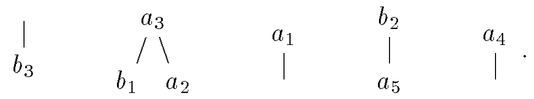
The formula tree of its indexed counterpart

$$\theta_{\mathtt{i}} \equiv ((a_1 \to b_1) \to a_2 \to a_3) \to ((a_4 \to a_5) \to b_2) \to b_3$$

is $\texttt{tree}(\theta_{\mathtt{i}}) =$



with primitive parts



3 Proof trees

DEFINITION 3.1
Let $\varphi_{\mathtt{i}}$ be the indexed counterpart of a type φ. A proof tree built from $\texttt{tree}(\varphi_{\mathtt{i}})$ is obtained by joining primitive parts of $\texttt{tree}(\varphi_{\mathtt{i}})$ identifying (different) occurrences of the same type variable. We call it a valid proof tree if

- all leafs in the proof tree are of form (p3);
- whenever a primitive part p_i occurs beneath some primitive part p_j in the formula tree $\texttt{tree}(\varphi_{\mathtt{i}})$, then above every occurrence of p_i in the proof tree there is at least one occurrence of p_j.

Note that the formula tree $\varphi_{\mathtt{i}}$ defines some kind of hierarchy over its primitive parts, which has to be respected by every valid proof tree built from $\mathtt{tree}(\varphi_{\mathtt{i}})$. In particular, this means that the root of every valid proof tree built from $\mathtt{tree}(\varphi_{\mathtt{i}})$ is the root of $\mathtt{tree}(\varphi_{\mathtt{i}})$.

EXAMPLE 3.2
The following is a valid proof tree built from the primitive parts of $\mathtt{tree}(\theta_{\mathtt{i}})$ from example 2.3.

$$\begin{array}{c} | \\ b_3 = b_2 \\ | \\ a_5 = a_3 \end{array}$$

$$\begin{array}{c} b_1 = b_2 \\ | \\ a_5 = a_1 \\ | \end{array} \qquad\qquad \begin{array}{c} a_2 = a_4 \\ | \end{array}$$

with restrictions $a_1 = a_3 = a_5$, $a_2 = a_4$ and $b_1 = b_2 = b_3$. Note that we represented the identification of different occurrences of the type variables in this proof tree by equations, such as $b_3 = b_2$.

4 Proof trees and (long) normal inhabitants

In this section we establish the relation between the set of normal inhabitants of a type φ and the set of valid proof trees constructed from $\mathtt{tree}(\varphi_{\mathtt{i}})$. More precisely, we analyse the existing relationship with a special subclass of normal inhabitants of φ, called long normal inhabitants. But, as we will see, this is in fact no restriction at all. A β-normal inhabitant M of a type φ is called a long normal inhabitant of φ iff every variable-occurrence z in M is followed by the longest sequence of arguments allowed by its type, i.e. iff each component with form $(zP_1 \ldots P_n)$, $(n \geq 0)$ that is not in a function position has atomic type. The (finite) set of all terms obtained by η-reducing a λ-term M is called the η-family of M and denoted by $\{M\}_\eta$. It has been shown (cf. [2], [9]) that the η-families of the long normal inhabitants of φ partition the set of normal inhabitants of φ into non-overlapping finite subsets, each η-family containing just one long member. Furthermore, Ben-Yelles (cf. [2], [9]) showed that every normal inhabitant of a type φ can be η-expanded to one unique (up to α-conversion) long normal inhabitant of φ. A simple expansion-algorithm can be found in [9].

We begin this section defining an algorithm which given a long inhabitant M of a type φ, with indexed counterpart $\varphi_{\mathtt{i}}$, computes a valid proof tree $\mathtt{PT}(M) = \mathtt{pt}(M, \varphi_{\mathtt{i}}, \emptyset)$ of $\mathtt{tree}(\varphi_{\mathtt{i}})$. For $\varphi_{\mathtt{i}} = \alpha_1 \to \ldots \to \alpha_n \to a_s$ and a set of type-assignments (basis) Γ, $\mathtt{pt}(M, \varphi_{\mathtt{i}}, \Gamma)$ is defined by the following.

- For $M = \lambda x_1 \ldots x_n.y$ and $y : a_t \in \Gamma \cup \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}$, let

$$\mathtt{pt}(M, \varphi_{\mathtt{i}}, \Gamma) = \begin{array}{c} | \\ a_s = a_t \\ | \end{array} .$$

- For $M = \lambda x_1 \ldots x_n.yM_1 \ldots M_m$ with $m \geq 1$, let $\mathtt{pt}(M, \varphi_{\mathtt{i}}, \Gamma) =$

$$\begin{array}{ccc} & | & \\ & a_s = a_t & \\ \mathtt{PT}_1 & \cdots & \mathtt{PT}_{\mathtt{m}} \end{array}$$

  where $\Gamma' = \Gamma \cup \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}$, $y : \beta_1 \to \ldots \to \beta_m \to a_t \in \Gamma'$ and for $1 \leq j \leq m$, $\mathtt{PT}_j = \mathtt{pt}(M_j, \beta_j, \Gamma')$.

PROPOSITION 4.1
If M is a long normal inhabitant of a type φ, with indexed counterpart $\varphi_{\mathtt{i}}$, then $\mathtt{PT}(M)$ is a valid proof tree built from $\mathtt{tree}(\varphi_{\mathtt{i}})$.

PROOF. By induction on the length of $\varphi_{\mathtt{i}}$, using the definition of the algorithm **tree** to construct the formula tree of $\varphi_{\mathtt{i}}$. ∎

COROLLARY 4.2
If a formula φ has a normal inhabitant M, then there is at least one valid proof tree built from $\mathtt{tree}(\varphi_{\mathtt{i}})$, where $\varphi_{\mathtt{i}}$ is the indexed counterpart of φ.

PROOF. It is sufficient to η-expand M to the unique long normal form M^+ in its η-family (using Ben-Yelles' algorithm). By proposition 4.1, $\mathtt{PT}(M^+)$ is a valid proof tree built from $\mathtt{tree}(\varphi_{\mathtt{i}})$. ∎

Now, we will define an *inverse* algorithm, which given the formula tree of the indexed counterpart of some type φ, $\mathtt{t}_{\mathtt{i}} = \mathtt{tree}(\varphi_{\mathtt{i}})$, and any valid proof tree $\mathtt{PT}$ built from it, computes a set $\mathtt{Terms}(\mathtt{PT}) = \mathtt{Terms}(\mathtt{t}_{\mathtt{i}}, 0, \mathtt{PT}, \emptyset)$ of long closed normal inhabitants of φ. In the following let L represent a set assigning variable names to primitive parts of $\mathtt{t}_{\mathtt{i}}$. Then, $\mathtt{Terms}(\mathtt{t}_{\mathtt{i}}, k, \mathtt{PT}, L)$ is defined by the following.

Let $\begin{array}{c} | \\ a_s = a_t \end{array}$ be the root of $\mathtt{PT}$ and consider the corresponding occurrence of a_s in $\mathtt{t}_{\mathtt{i}}$ of the form

$$\begin{array}{ccc} & | & \\ & a_s & \\ p_1 & \cdots & p_n \end{array}$$

with $n \geq 0$. Now, let $L' = L \cup \{(p_i, x_i^k) \,|\, 1 \leq i \leq n\}$.

- If $\mathtt{PT}$ is of the form $\begin{array}{c} | \\ a_s = a_t \\ | \end{array}$, then take the corresponding primitive part $p = \begin{array}{c} a_t \\ | \end{array}$ and let

$$\mathtt{Terms}(\mathtt{t_i}, k, \mathtt{PT}, L) = \{\lambda x_1^k \ldots \lambda x_n^k . x \mid (p, x) \in L'\}.$$

- Otherwise, $\mathtt{PT}$ is of the form

$$\begin{array}{ccc} & | & \\ & a_s = a_t & \\ \mathtt{PT}_1 & \cdots & \mathtt{PT}_m \end{array}$$

with $m \geq 1$, and such that for $i = 1, \ldots, m$ the root of $\mathtt{PT}_i$ is $\begin{array}{c} | \\ b_{s_i} = b_{t_i} \end{array}$. Consider the corresponding primitive part p of the form

$$\begin{array}{ccc} & | & \\ & a_t & \\ b_{s_1} & \cdots & b_{s_m} \end{array}$$

and let $\mathtt{Terms}(\mathtt{t_i}, k, \mathtt{PT}, L) =$

$\{\lambda x_1^k \ldots \lambda x_n^k . x M_1 \ldots M_m \mid (p, x) \in L', M_j \in \mathtt{Terms}(\mathtt{t_i}, k+1, \mathtt{PT}_j, L'), 1 \leq j \leq m\}.$

Proposition 4.3
Let $\mathtt{PT}$ be a valid proof tree built from the formula tree of the indexed counterpart of some type φ. Then every member of $\mathtt{Terms(PT)}$ is a closed long normal inhabitant of φ. Furthermore, the two algorithms are complementary in the sense that for every closed long normal inhabitant M of φ there is $M \in \mathtt{Terms}(\mathtt{PT}(M))$.

Proof. In order to prove the first claim, define a basis (set of type-assignments) from L by $\Gamma_L = \{x : \varphi_p \mid (x, p) \in L\}$, where φ_p denotes the subtype of φ which corresponds to the subtree beneath the primitive part p in $\varphi_{\mathtt{i}}$. Then, prove by induction on the structure of $\mathtt{PT}$ that for every $M \in \mathtt{Terms}(\mathtt{t_i}, k, \mathtt{PT}, L)$ there is $\Gamma_L \vdash M : \tau$, where τ is the subtype of φ which can be reconstructed from the occurrence of the first variable in the root of $\mathtt{PT}$, and that M is a β-nf which is long with respect to (Γ_L, τ). Then the result follows from the fact that the algorithms starts with an empty set L. The second part of the proof is by induction on the structure of M. ∎

Example 4.4
The algorithm $\mathtt{Terms}$ computes exactly one normal λ-term

$$M = \lambda xy.y(\lambda t.x(\lambda z.y(\lambda w.z))t)$$

corresponding to the valid proof tree in example 3.2. The η-family of M is

$$\{M, \lambda xy.y(x(\lambda z.y(\lambda w.z)))\}.$$

From corollary 4.2 and proposition 4.3 we conclude the following.

Corollary 4.5
A type φ is inhabitated if and only iff there is at least one valid proof tree constructed from $\mathtt{tree}(\varphi_{\mathtt{i}})$. ∎

Resuming, every normal inhabitant of a type φ corresponds to one valid proof tree built from $\mathtt{tree}(\varphi_{\mathtt{i}})$, every valid proof tree built from $\mathtt{tree}(\varphi_{\mathtt{i}})$ corresponds to a finite set of normal inhabitants of φ, and distinct valid proof trees correspond to distinct and disjoint sets of normal inhabitants.

5 Principal proof trees

Definition 5.1
Let φ be a type with indexed counterpart $\varphi_{\mathtt{i}}$. A valid proof tree $\mathtt{PT}$ built from $\mathtt{tree}(\varphi_{\mathtt{i}})$ is a principal proof tree iff

- $\mathtt{PT}$ uses every primitive part of form (p2) in $\mathtt{tree}(\varphi_{\mathtt{i}})$;
- $\mathtt{PT}$ identifies all occurrences of the same type variable in $\varphi_{\mathtt{i}}$.

The following result is the correspondent, for proof trees, to Proposition 17 in [3].

Proposition 5.2
A type φ has long principal normal inhabitants iff there is at least one principal proof tree $\mathtt{PPT}$ constructed from $\mathtt{tree}(\varphi_{\mathtt{i}})$. Furthermore, every term in $\mathtt{Terms(PPT)}$ is a principal long normal inhabitant of φ and whenever M is a long principal normal inhabitant of φ, then $\mathtt{PT} = \mathtt{pt}(M, \varphi_{\mathtt{i}}, \emptyset)$ is a principal proof tree.

Proof. The proof of this result follows the lines of the proof of Proposition 17 in [3]. ∎

Example 5.3
We conclude that the term $M = \lambda xy.y(\lambda t.x(\lambda z.y(\lambda w.z))t)$ (example 4.4) is a long normal, but no principal inhabitant of θ (example 2.3). In order to obtain a principal proof tree of θ it is necessary to further identify a_2 or a_4 with at least one of a_1, a_3 or a_5. This can be done by substituting in the proof tree of example 3.2 the (right) subtree

$$\begin{array}{c} \ldots = a_4 \\ | \end{array} \qquad \text{by} \qquad \begin{array}{ccc} & \ldots = a_3 & \\ / & & \backslash \\ b_1 = b_2 & & a_2 = a_4 \\ | & & | \\ a_5 = a_1 & & \\ | & & \end{array}$$

obtaining thus the principal proof tree `PPT` =

$$\begin{array}{c} | \\ b_3 = b_2 \\ | \\ a_5 = a_3 \end{array}$$

$$\begin{array}{ccccc} b_1 = b_2 & & & a_2 = a_3 & \\ | & & & / \quad \backslash & \\ a_5 = a_1 & & b_1 = b_2 & & a_2 = a_4 \\ | & & | & & | \\ & & a_5 = a_1 & & \\ & & | & & \end{array}$$

with restrictions $a_1 = a_2 = a_3 = a_4 = a_5$ and $b_1 = b_2 = b_3$ and using every primitive part of form (p2) in $\mathtt{tree}(\theta_\mathtt{i})$. Consequently, the (unique) member of the corresponding set of terms

$$\mathtt{Terms(PPT)} = \{\lambda xy.y(\lambda t.x(\lambda z.y(\lambda w.z))(x(\lambda u.y(\lambda v.u))t))\}$$

is a principal long normal inhabitant of θ.

In the previous example, note that the occurrence b_3, which corresponds to the tail of θ, had to be identified with two other occurrences of b. Consequently, in any principal proof tree there must be at least one other node, besides the root of the tree, identifying occurrences of b. In fact, in the principal proof tree `PPT` from the example there are two such nodes, both of the form $b_1 = b_2$. Now it is easy to see that we obtain another (more complex) valid principal proof tree when we substitute the subtree in either one of these nodes by the whole principal proof tree. We conclude the following for the general case.

PROPOSITION 5.4
Whenever the number of occurrences of the tail of a type φ is greater than or equal to three then the number of principal normal inhabitants of τ is either zero or infinite. ∎

6 Another example

Consider the type

$$\alpha = ((((a \to a) \to a) \to a) \to a) \to a \to a$$

which has an infinite number of (long) normal inhabitants, such as $\lambda xy.y$, $\lambda xy.x(\lambda z.y)$, $\lambda xy.x(\lambda z.z(\lambda u.y))$ or $\lambda xy.x(\lambda z.z(\lambda u.u))$, neither of which is principal. But does α have any principal normal inhabitant at all?

Using the algorithm defined in [3] and [4], we can answer this question after generating and testing a "finite" set of type-schemes with depth $\leq |\alpha|^4$, i.e. with depth $\leq 7^4 = 2401$. To understand the real complexity of this task, note that the four terms above have respectively depth 0, 1, 2 and 2.

On the other hand, analysing the formula tree of the indexed counterpart of α

$$\alpha_{\mathtt{i}} = ((((a_1 \to a_2) \to a_3) \to a_4) \to a_5) \to a_6 \to a_7$$

which is $\mathtt{tree}(\alpha_{\mathtt{i}})$ =

$$\begin{array}{ccc} & | & \\ & a_7 & \\ & & \\ a_5 & & a_6 \\ | & & | \\ a_4 & & \\ & & \\ a_3 & & \\ | & & \\ a_2 & & \\ & & \\ a_1 & & \\ | & & \end{array}$$

with primitive parts

$$p_1 = \begin{array}{c} | \\ a_7 \end{array} \qquad p_2 = \begin{array}{c} a_5 \\ | \\ a_4 \end{array} \qquad p_3 = \begin{array}{c} a_3 \\ | \\ a_2 \end{array} \qquad p_4 = \begin{array}{c} a_1 \\ | \end{array} \qquad p_5 = \begin{array}{c} a_6 \\ | \end{array},$$

we conclude that any valid proof tree built from $\mathtt{tree}(\alpha_{\mathtt{i}})$ = uses either p_4 or p_5 and will thus not identify a_1 or a_6 with the remaining occurrences of a in α. Consequently, none of the infinite number of valid proof trees for $\mathtt{tree}(\alpha_{\mathtt{i}})$ is a principal one and we conclude that α has in fact no principal normal inhabitants at all.

It is easy to see that every valid proof tree built from $\mathtt{tree}(\alpha_{\mathtt{i}})$ is either $\begin{array}{c} p_1 \\ p_5 \end{array}$, i.e. is $\begin{array}{c} | \\ a_7 = a_6 \\ | \end{array}$, or else of the form

$$\begin{array}{c} p_1 \\ p_2 \\ p_{i_1} \\ \vdots \\ p_{i_k} \\ p_f \end{array}$$

where $k \geq 0$, $i_1, \ldots, i_k \in \{2, 3\}$ and $f \in \{4, 5\}$. Via algorithm `Terms` we thus get a concrete idea of the structure of the set of normal inhabitants of α.

Acknowledgements

The work presented in this paper has been partially supported by funds granted to *LIACC* through *Programa do Financiamento Plurianual, Fundação para a Ciência e Tecnologia* and *Programa PRAXIS.*

References

[1] H. Barendregt. Lambda calculi with types. In Abramsky, Gabbay, and Maibaum, editors, *Background: Computational Structures*, volume 2 of *Handbook of Logic in Computer Science*, pages 117–309. Oxford Science Publications, 1992.

[2] C.-B. Ben-Yelles. *Type-assignment in the lambda-calculus; syntax and semantics.* PhD thesis, Mathematics Dept., University of Wales Swansea, UK, 1979.

[3] S. Broda and L. Damas. Counting a type's principal inhabitants. Proceedings of the 4th International Conference on Typed Lambda Calculi and Applications, *TLCA'99*, *Lecture Notes in Computer Science* 1581, pág. 69-82, Springer Verlag, 1999.

[4] S. Broda and L. Damas. Counting a type's (principal) inhabitants. To appear in *Fundamenta Informaticae.*

[5] A. Church, A formulation of the simple theory of types. *Journal of Symbolic Logic*, **5**, 56–68, 1940.

[6] H. B. Curry. Functionality in combinatory logic. *Proc. Nat. Acad. Science USA*, **20**, 584–590, 1934.

[7] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *Journal of Symbolic Logic*, **57**, 795–807, 1992.

[8] G. Gentzen. Untersuchungen über das logische Schliessen. *Math. Zeitschrift*, **39**, 176–210, 1935.

[9] J. R. Hindley. *Basic Simple Type Theory.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1997.

[10] S. Hirokawa. Note: Infiniteness of proof(α) is polynomial-space complete. *Theoretical Computer Science*, 206:331–339, 1998.

[11] W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[12] R. Statman. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science*, 9:67–72, 1979.

Approximate Belief Revision - A Preliminary Report

SAMIR CHOPRA, *CUNY Graduate Center and New York University, New York, USA. E-mail: schopra@gc.cuny.edu*

ROHIT PARIKH, *Brooklyn College of CUNY and CUNY Graduate Center, New York, USA. E-mail: ripbc@cunyvm.cuny.edu*

RENATA WASSERMANN, *Department of Computer Science, University of São Paulo, Brazil. E-mail: renata@ime.usp.br*

Abstract

The standard theory for belief revision [1] provides an elegant and powerful framework for reasoning about how a rational agent should change its beliefs when confronted with new information. However, the agents considered are extremely idealized. Some recent models attempt to tackle the problem of plausible belief revision by adding structure to the belief bases and using nonstandard inference operations. One of the key ideas is that not all of an agent's beliefs are relevant for an operation of belief change.

In this paper we incorporate the insights pertaining to local change and relevance sensitivity with the use of *approximate inference relations* [13]. These approximate inference relations offer us partial solutions at any stage of the revision process. The quality of the approximations improves as we allow for more and more resources to be used. We are provided with upper and lower bounds to what would be obtained with the use of classical inference.

Keywords: Belief Change, Approximate Reasoning

1 Introduction

Belief revision, the process of transforming a belief state on receipt of new information, is an integral part of any attempt to provide plausible architectures for automated agents. Conventional AGM-style revision [1] provides an elegant and powerful framework for reasoning about how a rational agent should change its beliefs when confronted with new information, but it tells us very little about *how* that agent could really perform such belief changes. Moreover, the rational agent described is a highly idealized one, a perfect reasoner with unbounded memory, logical ability, no inconsistent beliefs and no time constraints.

In the AGM paradigm, the belief state of an agent is represented by a *belief set*, i.e., a logically closed, consistent theory. Such an approach suffers from problems associated with representational and computational intractability. To alleviate these problems, *belief base* revision offers open, finite representations. This alternative has been extensively studied and AGM-like operations have been defined for belief bases. Although the use of belief bases solves the problem of representing a belief state, belief bases are typically quite large and the belief change operations make use of computationally expensive consistency checks.

One way to attack the problem is to try to reduce the *size* of the set to be explored. Recent models such as [12], [8], [4], [16], attempt to tackle the problem of plausible belief revision by using nonstandard inference operations such as local change operators and offering structuring relations on belief bases via relevance sensitivity. In these frameworks for local reasoning and belief change, the key idea is that not all of an agent's beliefs are relevant for an operation of belief change. As an example, suppose that I wake up and see that the sun is shining in Amsterdam. This contradicts my previous beliefs about the weather, like the one that it always rains in Holland, and leads to revision. However, the revision process does not have to take my beliefs about mathematics into account. These beliefs are completely irrelevant for the change taking place and should not play any role in it.

In this paper we incorporate the insights pertaining to local change and relevance sensitivity with the use of *approximate inference relations* [13]. These approximate inference relations offer us *partial solutions* at any stage of the revision process. The quality of the approximations improves as we allow for more and more resources to be used. The process of belief revision is rendered an approximate one, with the understanding that at any given stage, we are provided with upper and lower bounds to what would be obtained with the use of classical inference.

The outline of this paper is as follows: we first briefly summarize the concepts of local change and approximate inference. We then argue for heuristics based on relevance to guide the approximations. We show how approximate inference can be integrated with the local change paradigm. We close the paper with a discussion about the gains of using the two different approximations proposed in [13]. Due to the lack of space, all proofs are left for the full version of the paper.

Notation: L is a finite propositional language with the usual logical connectives $(\neg, \vee, \wedge, \rightarrow, \leftrightarrow)$. The constants $\top$, $\bot$ are in L; $\alpha, \beta, \gamma \ldots$ denote arbitrary formulae; $p, q, r \ldots$ denote propositional atoms; $A, B \ldots$ denote sets of formulae. For any formula α, $L(\alpha)$ is the set of propositional atoms in α. Cn stands for classical consequence, C stands for arbitrary inference operations.

2 Local Change

In this session, we briefly describe the paradigm for local change proposed in [8]. The main idea is that if we can isolate the part of a belief base which is relevant for a given change, we can apply the belief change operations to this subset of the base and keep the rest as it was.

In [8], given two sets of formulas A and B, the relevant formulas in B for A are defined to be those that contribute to proving or disproving any formula of A. This is formalized using the notion of *kernel sets* - minimal subsets implying a given sentence:

Definition 2.1 ([7])
Let C be an inference operation on L. A set X is an element of $B \perp\!\!\!\perp_C \alpha$ if and only if X is a minimal subset of B that implies α according to C. The elements of $B \perp\!\!\!\perp_C \alpha$ are called α-kernels.

We write $\perp\!\!\!\perp$ for $\perp\!\!\!\perp_{Cn}$, where Cn is a classical consequence operation.

Definition 2.2 ([8])
The A-compartment of B, where A and B are sets of sentences, is defined as:

$c(A,B) = \bigcup_{\alpha\in A}(\bigcup((B \perp\!\!\!\perp \alpha) \cup (B \perp\!\!\!\perp \neg\alpha) \setminus (B \perp\!\!\!\perp \bot))$

We call c the compartmentalization function for the sets A and B.

DEFINITION 2.3 ([8])
Let C be an inference operation on L and let c be as in Definition 2.2. Then the A-localization of C is the inference operation C_A such that: $C_A(B) = C(c(A,B))$.
A set B is *A-locally consistent* if and only if $\bot \notin C_A(B)$.

Local versions of the operations of contraction, revision, consolidation and semi-revision can be constructed and characterized by means of postulates. We reproduce below the characterization of *local kernel contraction* [8], which can be used for deriving the others. In kernel contraction, if we remove from the belief base at least one element of each α-kernel, we obtain a belief base that does not imply α. In the case of the local operation, we want a belief base that does not A-locally imply α, where A is a set of sentences. We consider only those kernel sets that A-locally imply α and use an *incision function* to select the formulas from the kernel sets to be removed.

DEFINITION 2.4 ([7])
An *incision function* is any function $\sigma : \mathcal{P}(\mathcal{P}(L)) \rightarrow \mathcal{P}(L)$ such that: 1. $\sigma(S) \subseteq \bigcup S$, and 2. If $\emptyset \neq X \in S$, then $X \cap \sigma(S) \neq \emptyset$.

DEFINITION 2.5 ([7])
Let C be an inference operation on L and σ an incision function. The local kernel contraction of B determined by C and σ is the operation $\dot{-}_{C,\sigma}$ such that: $B\dot{-}_{C,\sigma}\alpha = B \setminus \sigma(B \perp\!\!\!\perp_C \alpha)$

The following theorem characterizes the operation of local kernel contraction and is a generalization of the result obtained in [7]. The intended interpretation is that C is a local inference operator. However, the formal result is more general:

THEOREM 2.6 ([8])
Let C satisfy monotony and compactness. An operation $\dot{-}$ is a local kernel contraction determined by C and some incision function iff for all B:

- If $\alpha \notin C(\emptyset)$, then $\alpha \notin C(B\dot{-}\alpha)$ (success)
- $B\dot{-}\alpha \subseteq B$ (inclusion)
- If $x \in B \setminus B\dot{-}\alpha$, then there is some $B' \subseteq B$ such that $\alpha \notin C(B')$ and $\alpha \in C(B' \cup \{x\})$ (core-retainment)
- If for all $B' \subseteq B$, $\alpha \in C(B')$ iff $\beta \in C(B')$, then $B\dot{-}\alpha = B\dot{-}\beta$ (uniformity)

Sets of postulates and representation results are given for operations of partial meet contraction, kernel revision and partial meet revision. Besides compactness and monotony, two other properties are needed for the characterization of revision:

- **consistency** $\bot \notin C(\emptyset)$.
- **α-local non-contravention** if $\neg\alpha \in C(B \cup \{\alpha\})$, then $\neg\alpha \in C(B)$.

The construction of the compartments required for the local change inference operations is computationally expensive. Local change operations are computationally as costly as the original versions, although intuitively more appropriate. However, the representation results proven in [8] do not rely on the way the compartments are defined, but only on properties of the local inference operation obtained, namely, monotony, compactness, consistency, and α-local non-contravention.

We now turn to more efficient ways in which to isolate a meaningful part of a belief base for a change operation.

3 Approximate entailment

In this section, we summarize the main results concerning approximate inferences.

In [13], Schaerf and Cadoli define two approximations of classical entailment: $\models_S^1$ which is complete but not sound and $\models_S^3$ which is sound and incomplete. These approximations are carried out over a set of atoms $S \subseteq L$ which determines their closeness to classical entailment. In the trivial extreme of approximate entailment, i.e., when $S = L$, classical entailment is obtained. At the other extreme, when $S = \emptyset$, $\models_S^1$ holds for any two formulas and $\models_S^3$ corresponds to Levesque's logic for explicit beliefs [11], which bears a connection to Anderson and Belnap's relevance logic.

In an S_1 assignment, if $x \in S$, then $x, \neg x$ are given opposite truth values; if $x \notin S$, then $x, \neg x$ both get the value 0. In an S_3 assignment, if $x \in S$, then $x, \neg x$ get opposite truth values, while if $x \notin S$, $x, \neg x$ do not both get 0, but may both get 1. The belief base B is assumed to be in clausal form.[1]

The following examples illustrate the use of approximate entailment. Since $\models_S^3$ is sound but incomplete, it can be used to approximate $\models$, i.e., if for some S we have that $B \models_S^3 \alpha$, then $B \models \alpha$. On the other hand, $\models_S^1$ is unsound but complete, and can be used for approximating $\not\models$, i.e., if for some S we have that $B \not\models_S^1 \alpha$, then $B \not\models \alpha$.

Example 3.1 ([13])
We want to check whether $B \models \alpha$, where $\alpha = \neg\texttt{cow} \vee \texttt{has-molar-teeth}$ and

$$B = \{\neg\texttt{cow} \vee \texttt{grass-eater}, \neg\texttt{dog} \vee \texttt{carnivore}, \neg\texttt{grass-eater} \vee \neg\texttt{has-canine-teeth}, \neg\texttt{carnivore} \vee \texttt{mammal}, \neg\texttt{mammal} \vee \texttt{has-canine-teeth} \vee \texttt{has-molar-teeth}, \neg\texttt{grass-eater} \vee \texttt{mammal}, \neg\texttt{mammal} \vee \texttt{vertebrate}, \neg\texttt{vertebrate} \vee \texttt{animal}\}.$$

For $S = \{\texttt{grass-eater}, \texttt{mammal}, \texttt{has-canine-teeth}\}$, we have that $B \models_S^3 \alpha$, hence $B \models \alpha$.

Example 3.2 ([13])
We want to check whether $B \not\models \beta$, where $\beta = \neg\texttt{child} \vee \texttt{pensioner}$ and

$$B = \{\neg\texttt{person} \vee \texttt{child} \vee \texttt{youngster} \vee \texttt{adult} \vee \texttt{senior}, \neg\texttt{pensioner} \vee \texttt{senior}, \neg\texttt{youngster} \vee \texttt{student} \vee \texttt{worker}, \neg\texttt{senior} \vee \texttt{pensioner} \vee \texttt{worker}, \neg\texttt{adult} \vee \texttt{student} \vee \texttt{worker} \vee \texttt{unemployed}, \neg\texttt{pensioner} \vee \neg\texttt{student}, \neg\texttt{student} \vee \texttt{child} \vee \texttt{youngster} \vee \texttt{adult}, \neg\texttt{pensioner} \vee \neg\texttt{worker}\}.$$

For $S = \{\texttt{child}, \texttt{worker}, \texttt{pensioner}\}$, we have that $B \not\models_S^1 \beta$, and hence $B \not\models \beta$.

Note that in both examples above, S is a small part of the language. The approximation of classical inference is made via a simplification of the belief base B as follows:

Lemma 3.3 ([13])
Let *simplify-3*(B, S) be the result of deleting all clauses of B which contain an atom outside S. Then B is S_3-satisfiable iff *simplify-3*(B, S) is classically satisfiable.

[1] All results can be extended for negation normal form cf. [2], but further generalization implies missing the good complexity upper bounds, as shown in [3]. Besides the increase in complexity, the standard translation of formulas into NNF makes use of De Morgan's law and does not preserve truth values under Schaerf and Cadoli's non-standard semantics.

LEMMA 3.4 ([13])
Let *simplify-1*(B, S) be the result of deleting all literals of B which mention atoms outside S. B is S_1-satisfiable if and only if *simplify-1*(B, S) is classically satisfiable.

THEOREM 3.5 ([13])
There exists an algorithm for deciding if $B \models_S^3 \alpha$ and deciding $B \models_S^1 \alpha$ which runs in $O(|B|.|\alpha|.2^{|S|})$ time.

The result above depends on a poly-time satisfiability algorithm for belief bases and formulas in clausal form alone. This result has been extended in [2] for formulas in negation normal form, but is not extendable to formulas in arbitrary forms [3].

4 Using Relevance to Guide the Approximations

The main problem with the use of approximate entailment, as noted in [13] and in [15], is that there is no general strategy or heuristic for choosing the right set S for the approximation. In this section, we argue for the use of a notion of *relevance* in order to guide the choice of the context set S.

It has been shown in [5] and [16] that one way in which relevance in a database can be defined is by looking at the syntactical structure of the formulas. In these studies, two formulas were considered relevant to each other if they shared an atom; the definition was generalized to the case when formulas did not share an atom with each other but did so with other formulas that in turn shared atoms. This enabled the definition of a notion of degrees or levels of relevance. In the present paper, we will use a notion of relevance between propositional variables: two atoms are relevant to each other if and only if they occur in the same formula in the belief base.[2]

DEFINITION 4.1
Given a belief base B, two atoms p, q are directly relevant, $R(p, q, B)$, iff there is $\alpha \in B$ such that $p, q \in L(\alpha)$. Two atoms p, q are k-relevant w.r.t B if $\exists p_0, p_1, \ldots p_{k+1} \in L$ such that: i) $p_0 = p$; ii) $p_{k+1} = q$; iii) $R(p_i, p_{i+1}, B)$, $i = 0, \ldots k$.

We first look at some necessary conditions that the context set S must satisfy. As shown in [14][3], if we want to use S_1 interpretations to approximate $B \models \alpha$, S must contain at least $L(\alpha)$ and at least one propositional variable occurring in each of the clauses of B. For S_3 interpretations, no such requirement is needed.

The following strategy may then be used in the construction of the set S. If we are considering S_1 entailments, we start with S being such that $L(\alpha) \subseteq S$ and for every formula $\gamma \in B$, $L(\gamma) \cap S \neq \emptyset$. For S_3 entailment, lemma 5.1 from [13] says that $L(\alpha)$ can be left out of S.

Construction strategy for S:

1. For S_1 entailment, start with S containing $L(\alpha)$ and at least one atom from each formula in B; for S_3, start with $S = \{p | p$ directly relevant to q and $q \in L(\alpha)\}$.
2. Test entailments.
3. If the result is not satisfactory and resources are available, add to S the atoms relevant to elements in S and go to 2.

---

[2] This is used in RABIT [6] and cited in [16].

[3] Theorems 2 and 3.

4. Return approximations.

If we look back to Example 3.1, we see that we start with S being the atoms directly relevant to `cow` and `has-molar-teeth`, i.e., $S=$ {`grass-eater`, `mammal`, `has-canine-teeth`}. Applying this technique to example 3.2, we start with $S = L(\alpha)$. Since $L(\alpha)$ does not intersect all clauses in B, we have a choice of adding either `worker`, `student` or both `youngster` and `unemployed`, or both `youngster` and `adult` to S. If we start with 1-relevant, S is the entire language L except for `unemployed`. Clearly, such a result is not satisfactory since our strategy has not provided us with a useful heuristic for minimizing the size of the set S. In session 6, we suggest an improvement to this strategy.

Example 4.2
Take $B = \{p, p \to q, q \to r, r \to s, s \to t\}, \alpha = t$. Now, $S = \{p, q, r, s\}$, is constructed by collecting all atoms up to 3-degrees of relevance and the degree of relevance which we need to consider is a function of the length of proof required to derive α from B.

Note, however, that belief bases of the form above are likely to be extremely rare in practice and therefore, we may expect that in most cases, S will be constructed fairly quickly i.e., with small k. In any case, the fact that it takes a larger S to find a long proof is in line with our intuition that long proofs are only found if the agent has enough resources and can, therefore, reason with a larger context set S.

Proposition 4.3
Using the above strategy, S_1 and S_3 entailment converge, i.e., after a finite number of iterations, we have that either $B \not\models^1_S \alpha$, hence $B \not\models \alpha$ or $B \models^3_S \alpha$ hence $B \models \alpha$.

It is not hard to see that in most cases, our strategy will return a set S which is larger that what is needed. As noted in [15], if we have domain information about the particular problem we are trying to solve, specific heuristics can be used to limit the context set S. The particular notion of relevance that we use in this paper is to be used in the general case, when no domain information is available. In the case of particular applications, usually there is a better notion of relevance that can be used (see for example [17]). If B is in clausal form, we can use dependence links as used in logic programming [10].

5 Approximating Belief Change Operators

We now show that the notion of approximate entailment described in the previous section can be used for implementing local change efficiently. As we have noted above in section 2, if an inference operation satisfies monotony, compactness, consistency, and α-local non-contravention, it can be used together with the axiomatizations given in [8]. We now show that approximate entailment satisfies these requirements.

Definition 5.1
$C^i_S(B) = \{\alpha | B \models^i_S \alpha\}$, where S is a set of propositional variables and $i \in \{1, 3\}$.

Proposition 5.2
Let S be a fixed set of propositional variables. Then C^3_S satisfies monotony, compactness, consistency, and α-local non-contravention for every formula α.

PROPOSITION 5.3
Let S be a fixed set of propositional variables. Then C_S^1 satisfies monotony, compactness, consistency, and α-local non-contravention iff $L(\alpha) \subseteq S$.

As an example of how C_S^1 can fail to satisfy the conditions above when $L(\alpha) \not\subseteq S$, take $\alpha = p$, $B = \emptyset$, and $p \notin S$. Then any S_1 interpretation assigns both α and $\neg\alpha$ the value 0. We have that $\neg\alpha \in C_S^1(B \cup \{\alpha\}) = Cn(\perp)$, but $\neg\alpha \notin C_S^i(B) = Cn(\emptyset)$.

The propositions above imply that both inference operations can be used to define local operations of belief change. This allows us to combine the computational efficiency of Schaerf and Cadoli's method for approximate entailment with the logical characterization of belief change operations given in [8]. The idea is that these approximate inferences give us an approximation of the revised belief base and the larger the set S grows, the closer we get to the classical definition. At any point in the revision process, we can stop to check our results and depending upon resource availability, we can choose to carry on the revision process further or stop. To this end, we define the following contraction operations:

DEFINITION 5.4
Let C_S^i be an inference operation on L and σ an incision function. The approximate local kernel contraction of B determined by C_S^i and σ is the operation $\dot{-}_{C_S^i,\sigma}$ such that: $B \dot{-}_{C_S^i,\sigma} \alpha = B \setminus \sigma(B \perp\!\!\!\perp_{C_S^i} \alpha)$

As we have seen, since the operations C_S^i satisfy the required properties, approximate kernel contraction is characterized by theorem 2.6.

Note that in keeping with the approximated inference operation used above, we will obtain two kinds of contraction operations. Operations based on S_1 entailment will model more radical contraction operations (leading to a greater loss of beliefs) and those based on S_3 will model cautious operations (fewer beliefs than warranted are dropped). The choice of which contraction operation to use will be a context-sensitive one. At any stage of the approximated revision process, we know that contraction based on classical consequence lies between contraction based on approximated consequence relations, and that adding more propositional variables to the context S provides a better approximation. If $S = L$, then $B \dot{-}_{C_S^i,\sigma} \alpha = B \dot{-}_{Cn,\sigma} \alpha$. Usually, the approximations will converge for a proper subset S of L.

DEFINITION 5.5
Let $\mathcal{X}$ and $\mathcal{Y}$ be two families of sets. We say that $\mathcal{X} \sqsubseteq \mathcal{Y}$ if and only if for all $Y \in \mathcal{Y}$ there is $X \in \mathcal{X}$ such that $X \subseteq Y$.

PROPOSITION 5.6
$(B \perp\!\!\!\perp_{C_S^1} \alpha) \sqsubseteq (B \perp\!\!\!\perp_{Cn} \alpha) \sqsubseteq (B \perp\!\!\!\perp_{C_S^3} \alpha)$.

It is not difficult to see that since C_S^1 is complete but not sound, there will be at least as many C_S^1 kernels as there are classical ones. However, some of the sets which are kernels according to C_S^1 may not be subsets of any classical kernel. A similar relation holds for classical and C_S^3 kernels. An example of these inclusion relations is provided in Section 5.1 below.

In the model that we have presented above, belief revision is viewed as an approximated process, arrived at by a combination of the revision schemes presented above. Given that the approximate inference operations satisfy the conditions required for the general results obtained in [8], we note that similar results will obtain for the operations of partial meet contraction, partial meet and kernel revision.

5.1 Example

We now present an example that demonstrates the use of the relevance based heuristic for constructing a context-set S and the application of approximated inference to local kernel contraction. The following represents our beliefs about a young student, Hans.

$B =$ {`student` ,¬`student` ∨ `young`,¬`young` ∨ ¬`pensioner`, `worker`, ¬`worker` ∨ ¬`pensioner`, `blue-eyes`, `likes-dancing`, `six-feet-tall`}.

We now find out that Hans is a young genius who made millions of dollars in the stock market and is now happily retired and drawing a pension. As a result, we want to contract B by ¬`pensioner`. The tables below show how B gets simplified as S grows.

B	$S =$ {pensioner}	$S =$ {pensioner,young, worker}	$S =$ {pensioner,young, worker,student}
student	⊥	⊥	student
¬student ∨ young	⊥	young	¬student ∨ young
¬young ∨ ¬pensioner	¬pensioner	¬young ∨ ¬pensioner	¬young ∨ ¬pensioner
worker	⊥	worker	worker
¬worker ∨ ¬pensioner	¬pensioner	¬worker ∨ ¬pensioner	¬worker ∨ ¬pensioner
blue-eyes	⊥	⊥	⊥
likes-dancing	⊥	⊥	⊥
six-feet-tall	⊥	⊥	⊥

TABLE 1. Simplify-1 applied to B with different values of S.

Table 1 shows the result of simplifying the belief base B according to S_1 using different context sets. The second column of table 1 shows the result of *simplify-1*(B, {`pensioner`}). Since all formulas which are reduced to inconsistency are minimal sets S_1-implying anything, this gives us the following set of kernels:
$B \perp\!\!\!\perp_{C_S^1}$ ¬`pensioner`={{`student`}, {¬`student` ∨ `young`}, {¬`young` ∨ ¬`pensioner`}, {`worker`}, {¬`worker` ∨ ¬`pensioner`}, {`blue-eyes`}, {`likes-dancing`}, {`six-feet-tall`}}.

As S grows, some S_1-kernels are eliminated. The third column, where $S=$ {`pensioner`, `young`, `worker`}, gives: $B \perp\!\!\!\perp_{C_S^1}$ ¬`pensioner`={{`student`}, {¬`student` ∨ `young`, ¬`young` ∨ ¬`pensioner`},{`worker`, ¬`worker` ∨ ¬`pensioner`}, {`blue-eyes`},{`likes-dancing`},{`six-feet-tall`}}.

B	$S =$ {pensioner}	$S =$ {pensioner, young, worker}	$S =$ {pensioner, young, worker, student}
student	-	-	student
¬student ∨ young	-	-	¬student ∨ young
¬young ∨ ¬pensioner	-	¬young ∨ ¬pensioner	¬young ∨ ¬pensioner
worker	-	worker	worker
¬worker ∨ ¬pensioner	-	¬worker ∨ ¬pensioner	¬worker ∨ ¬pensioner
blue-eyes	-	-	-
likes-dancing	-	-	-
six-feet-tall	-	-	-

TABLE 2. Simplify-3 applied to B with different values of S.

Table 2 shows the result of simplifying B according to S_3.

This example serves as an illustration of proposition 5.6. Using an appropriate incision function would provide us with similar upper and lower bounds for the contracted belief base.

6 Alternative Heuristics

As we have seen in Section 4, for S_1, we have to start the approximation with a set S which intersects all formulas of the belief base in order to guarantee convergence. There are many possible such sets and it is not trivial how to select one. For a special sort of belief base, this is not necessary:

Definition 6.1
A belief base B is connected iff any two atoms $p, q \in L(B)$ are k-relevant for some k.

Proposition 6.2
If B is connected, then our strategy for S_1 approximation converges to the classical result for the initial context set $S = L(\alpha)$.

We illustrate the necessity of the connectedness condition via the example in Section 5.1. It is easy to see that the initial base is not connected. The last three formulas (`blue-eyes`, `likes-dancing`, `six-feet-tall`) are completely disconnected from the rest. They are treated in the appropriate way by S_3 approximations, i.e., they are ignored. On the other hand, *simplify-1*(B, S) renders them inconsistent whenever S does not mention their atoms. This means that these irrelevant formulas will always generate inconsistent kernels which are singletons and therefore will be needlessly deleted by a contraction operation. We can choose to pre-process the belief base eliminating irrelevant formulas as suggested in [4] and [16]: we only revise the relevant component and keep the rest of the belief base unchanged.

We now present a slightly different strategy which uses $\models^1_S$ in a more sophisticated way. Recall from Definition 4.1 that $R(p, q, B)$ iff there is a clause in B which has literals based on both p, q. Now define R_i as the set of atoms which are i relevant to atoms in $L(\alpha)$ according to Definition 4.1. Then for any k we choose, divide all atoms into three classes: $V = R_k$, $T = R_{k+1} \setminus R_k$, and $U = L - R_{k+1}$. V are the atoms most relevant to α, T are the atoms of medium relevance and U contains those which, we hope, are not relevant to the truth of α. Now eliminate all literals based on atoms from T. The remaining clauses either contain literals only from R_k or else literals not from R_{k+1}, and these two varieties never occur together in a clause. Let B_1 be the first (relatively small) set of clauses and B_2 the second. Assuming that B_2 is not inconsistent, we get, $B_1 \not\models \alpha$ iff $B \not\models^1_S \alpha$ and of course if $B \not\models^1_S \alpha$, then $B \not\models \alpha$. This allows us to approximate $B \models \alpha$ from the negative direction. We postpone details and proofs to the full paper.

7 Conclusion

In this paper, we have presented an application of the techniques of approximate entailment to belief revision. We have shown how approximate belief change fits the local change paradigm given in [8]. This provides us a clear logical axiomatization of the approximate operations. It also provides us an efficient method for implementing operations of belief change as performed by resource-bounded agents. In addition, we have provided heuristics for the construction of the context set S, without which any implementation of approximated reasoning systems would be very difficult. We note that domain specific knowledge will play a key role in any heuristic for constructing the context set. In future work, we plan to implement such a system and further investigate its properties.

Acknowledgments

The research reported here was funded by NSF.

References

[1] Carlos Alchourrón, Peter Gärdenfors and David Makinson. On the logic of theory change. In *Journal of Symbolic Logic*, Volume 50, pages 510-530, 1985.

[2] Marco Cadoli and Marco Schaerf. Approximate inference in default logic and circumscription. In *Fundamenta Informaticae* , Vol. 23, 1995, 123-143.

[3] Marco Cadoli and Marco Schaerf. The Complexity of Entailment in Propositional Multivalued Logics, *Annals of Mathematics and Artificial Intelligence*, Vol. 18 (1), 1996, 29-50.

[4] Samir Chopra and Rohit Parikh. An inconsistency tolerant model for belief representation and belief revision. In *Proceedings of IJCAI 99*, Morgan Kaufmann.

[5] Samir Chopra, Konstantinos Georgatos, and Rohit Parikh. Non-monotonic inference on belief sequences (preliminary report). In Phokion Kolaitis and George Koletsos, editors, *Proceedings of the Second Pan-Hellenic Symposium on Logic*, 1999.

[6] Kevin Gary. RABIT – A spreading activation approach to real-time commonsense reasoning. Master's thesis, Arizona State University, Tempe, Arizona, 1993.

[7] Sven Ove Hansson. Kernel contraction. *Journal of Symbolic Logic*, 59:845–859, 1994.

[8] Sven Ove Hansson and Renata Wassermann. Local change: A preliminary report. In *Fourth Symposium on Logical Formalizations of Commonsense Reasoning, London*, 1998.

[9] Andreas Herzig. How to change factual beliefs using laws and independence information. In D. Gabbay, R. Kruse, A. Nonnengart, and H. Ohlbach, editors, *Qualitative and Quantitative Practical Reasoning - ECSQARU-FAPR'97*, number 1244 in LNCS. Springer, 1997.

[10] Robert Kowalski. *Logic for Problem Solving*. North Holland, 1979.

[11] Hector Levesque A logic of explicit and implicit belief. In *Proceedings of AAAI-84*.

[12] Rohit Parikh. Beliefs, belief revision, and splitting languages (presented at ITALLIC-96) In *Logic, Language, and Computation, Volume 2*, eds. Lawrence S. Moss, Jonathan Ginzburg, and Maarten de Rijke, CSLI Lecture Notes No. 96, CSLI Publications, 1999, pp. 266–268.

[13] Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74(2):249–310, 1995.

[14] Annette ten Teije and Frank van Harmelen. Computing approximate diagnoses by using approximate entailment. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 256–265, Boston, November 1996.

[15] Annette ten Teije and Frank van Harmelen. Exploiting domain knowledge for approximate diagnosis. In M. Pollack, editor, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 454–459, Nagoya, Japan, August 1997.

[16] Renata Wassermann. On structured belief bases. In Hans Rott and Mary-Anne Williams, editors, *Frontiers in Belief Revision*. Kluwer, 2000. to appear.

[17] Renata Wassermann. Local Diagnosis. In Proceedings of the 8th International Workshop on Non-Monotonic Reasoning, Breckenridge, Colorado, 2000.

A Model Theoretic Approach to Translations Between Logics

MARCELO ESTEBAN CONIGLIO, *Group of Theoretical and Applied Logic, Department of Philosophy, State University of Campinas, P.O. Box 6133, 13081-970, Campinas, SP, Brazil.*
E-mail: coniglio@cle.unicamp.br

WALTER ALEXANDRE CARNIELLI, *Group of Theoretical and Applied Logic, Department of Philosophy and Center for Logic, Epistemology and the History of Science, State University of Campinas, P.O. Box 6133, 13081-970, Campinas, SP, Brazil.*
E-mail: carniell@cle.unicamp.br

Abstract

We use the language and concepts of model theory to describe the relationship between logics through translations. Abstract logics are conceived as two-sorted first-order structures, and translations between logics as morphisms of structures. This leads to a formal framework for studying abstract logics and translations between them, appropriate to give precise content to the slogan: *if T is a translation between logics* L_1 *and* L_2*, then the meaning of* L_1 *is preserved in* L_2.

Keywords: abstract logics, combinations of logics, translations between logics, model theory

1 Basic motivation

The concept of translation between logics (introduced in [7], see also [5] for comments) is a very broad one, and several properties of abstract logics, such as relative consistency and the meaning of a logical system in terms of others (via *possible-translations semantics*, cf. [4]), can be stated in terms of translations. We are using here the notion of *abstract logic* in the sense of the seminal paper [3] by Brown and Suszko.

However, there is no theory capable of expressing the general properties of abstract logics and explaining the full meaning of translations between them. In this paper we discuss this problem, and offer a formal framework where logics and their translations can be treated and their properties expressed. This section recalls some basic definitions and results from (many-sorted) model theory. We assume the reader is familiar with the notion of a *two-sorted first-order structure* as a 5-uple

$$\mathrm{A} = \langle A_1, A_2, \mathcal{P}, \mathcal{F}, \mathcal{C}\rangle.$$

The (first-order, two-sorted) language $\mathbb{L}$ associated with a two-sorted structure and the notion of satisfiability in a two-sorted structure are defined as usual. The sorts of $\mathbb{L}$ will be denoted by $\mathbf{A_1}$ and $\mathbf{A_2}$. We also assume familiarity with related notions such as: *restrictions* $\mathrm{A}_{|\langle B_1, B_2\rangle}$ for $\emptyset \neq B_i \subseteq A_i$ $(i = 1, 2)$, and *expansions*.

DEFINITION 1.1
Let $A = \langle A_1, A_2, \mathcal{P}, \mathcal{F}, \mathcal{C}\rangle$ and $B = \langle B_1, B_2, \mathcal{P}', \mathcal{F}', \mathcal{C}'\rangle$ be two-sorted structures for the same language $\mathbb{L}$. A *morphism of structures* $h : A \longrightarrow B$ is a pair $h = \langle F_1, F_2\rangle$ such that:

(i) $F_i : A_i \longrightarrow B_i$ for $i = 1, 2$;
(ii) if P is a $(n + m)$-ary predicate symbol of $\mathbb{L}$, then

$$(\vec{a}; \vec{b}) \in P^A \quad \text{implies that} \quad (F_1(a_1), ..., F_1(a_n); F_2(b_1), ..., F_2(b_m)) \in P^B$$

for all $\vec{a} = (a_1, ..., a_n) \in A_1^n$, $\vec{b} = (b_1, ..., b_m) \in A_2^m$;
(iii) if c (of sort $\mathbf{A_1}$) and d (of sort $\mathbf{A_2}$) are symbols for constants, then

$$F_1(c^A) = c^B \quad \text{and} \quad F_2(d^A) = d^B;$$

(iv) if $f : \mathbf{A_1}^n \times \mathbf{A_2}^m \longrightarrow \mathbf{A_1}$ is a symbol of function of $\mathbb{L}$, then

$$F_1(f^A(\vec{a}; \vec{b})) = f^B(F_1(a_1), ..., F_1(a_n); F_2(b_1), ..., F_2(b_m))$$

for all $\vec{a} = (a_1, ..., a_n) \in A_1^n$ and $\vec{b} = (b_1, ..., b_m) \in A_2^m$. An analogous condition is required if f has codomain of sort $\mathbf{A_2}$.

A morphism $h = \langle F_1, F_2\rangle$ is an *isomorphism* if F_1 and F_2 are both bijective, and "implies that" is substituted by "iff" in condition (ii) above. We say that A is *embeddable* in B, denoted by $A \subseteq B$, if there exists a morphism $h : A \longrightarrow B$ such that h is an isomorphism between A and $B_{|\langle F_1(A_1), F_2(A_2)\rangle}$.

PROPOSITION 1.2
If $\langle F_1, F_2\rangle : A \longrightarrow B$ is an isomorphism, then

$$A \models \varphi[a_1, ..., a_n; b_1, ..., b_m] \quad \text{iff} \quad B \models \varphi[F_1(a_1), ..., F_1(a_n); F_2(b_1), ..., F_2(b_m)]$$

for all formulas $\varphi(x_1, ..., x_n; y_1, ..., y_m)$ with at most $x_1, ..., x_n$ and $y_1, ..., y_m$ as free variables (of sort $\mathbf{A_1}$ and $\mathbf{A_2}$, respectively), for all $(a_1, ..., a_n) \in A_1^n$ and for all $(b_1, ..., b_m)$ in A_2^m. In particular, $A \models \sigma$ iff $B \models \sigma$ for all $\mathbb{L}$-sentences σ.

2 Logics as structures

In this section we show how model-theoretic concepts can express deep properties of abstract logics and translations between them.

DEFINITION 2.1
An *abstract logic* L is a two-sorted structure of the form

$$L = \langle X, Z, \in, \vdash, \cup, \{\cdot\}, \emptyset\rangle$$

where $Z \subseteq \mathcal{P}(X) = \{\Gamma : \Gamma \subseteq X\}$, $\in\ \subseteq X \times Z$ is the *belonging* relation, $\vdash\ \subseteq Z \times X$ is the *consequence* relation, $\cup : Z \times Z \longrightarrow Z$ is the *join* operation and $\{\cdot\} : X \longrightarrow Z$, $\{\cdot\}(x) = \{x\}$ for all $x \in X$. The relation $\in$ and the functions $\cup$ and $\{\cdot\}$ have their usual set-theoretic meaning, and $\emptyset$ is the empty set. Additionally, Z must be a Boolean algebra satisfying the following:

(i) $\emptyset \in Z$ and $X \in Z$;
(ii) $\{a\} \in Z$ for all $a \in X$;
(iii) if $\Gamma, \Delta \in Z$ then $\Gamma \cup \Delta \in Z$;
(iv) if $\Gamma \in Z$, then $X - \Gamma \in Z$;
(v) if $\Gamma \in Z$, then $\{a \in X \ : \ \Gamma \vdash a\} \in Z$.

Throughout the rest of this paper, $\mathbb{L}$ will denote the language

$$\{\mathbf{A_1}, \mathbf{A_2}, \in, \vdash, \cup, \{\cdot\}, \emptyset\}$$

appropriate to "talk about" abstract logics. Note that $\mathbb{L}$ contains the constant symbol $\emptyset$ (of sort $\mathbf{A_2}$) for the empty set. A structure of the type of abstract logics will be denoted by

$$\mathrm{A} = \langle X, Z, \in_{\mathrm{A}}, \vdash_{\mathrm{A}}, \cup_{\mathrm{A}}, \{\cdot\}_{\mathrm{A}}, \emptyset_{\mathrm{A}} \rangle,$$

omitting the subscripts in the set-theoretic operations if A is an abstract logic.

We will see that translations between logics are special cases of morphisms between structures.

Definition 2.2
Let $\mathrm{L}_i = \langle X_i, Z_i, \in, \vdash_i, \cup, \{\cdot\}, \emptyset \rangle$ be abstract logics ($i = 1, 2$). A *translation from* L_1 *to* L_2 is a morphism $h : \mathrm{L}_1 \longrightarrow \mathrm{L}_2$, $h = \langle T, IT \rangle$ such that

$$IT(\Gamma) = \{T(a) \ : \ a \in \Gamma\} \ (= T(\Gamma)) \qquad \text{for all } \Gamma \in Z_1.$$

An isomorphic translation between L_1 and L_2 is called a *strongly conservative translation from* L_1 *to* L_2. A translation h is *conservative* if h is a morphism satisfying "iff" instead of "implies that" in condition (ii) of 1.1.

It is worth observing that, if $h : \mathrm{L}_1 \longrightarrow \mathrm{L}_2$ is a translation, then

$$\Gamma \vdash_1 a \quad \text{implies} \quad T(\Gamma) \vdash_2 T(a),$$

and, if h is conservative, the following holds:

$$\Gamma \vdash_1 a \quad \text{iff} \quad T(\Gamma) \vdash_2 T(a),$$

that is, we recapture the usual definitions of translation and conservative translation introduced in [7]. It is also to be noted that the parameter IT is irrelevant in our definition of translation. On the other hand, the image of a logic under a conservative translation $T : \mathrm{L}_1 \longrightarrow \mathrm{L}_2$ is also a logic (T being conservative ensures that $T(X_1)$ satisfies condition 2.1(v)). T is a strongly conservative translation iff T is a bijective conservative translation. This kind of translation is very important, because it determines that the logics involved are the same (from the point of view of the language $\mathbb{L}$). In other words, the logics related by translations of this kind are indiscernible by sentences in the language $\mathbb{L}$ of logics.

PROPOSITION 2.3
Let $T : \mathrm{L}_1 \longrightarrow \mathrm{L}_2$ be a strongly conservative translation. Then for any formula $\varphi(x_1, ..., x_n; Y_1, ..., Y_m)$, every $(a_1, ..., a_n) \in X_1^n$ and $(\Gamma_1, ..., \Gamma_m)$ in Z_1^m,

$$\mathrm{L}_1 \models \varphi[a_1, ..., a_n; \Gamma_1, ..., \Gamma_m] \quad \text{iff} \quad \mathrm{L}_2 \models \varphi[T(a_1), ..., T(a_n); T(\Gamma_1), ..., T(\Gamma_m)],$$

where φ has at most $x_1, ..., x_n$ (of sort $\mathbf{A_1}$) and $Y_1, ..., Y_m$ (of sort $\mathbf{A_2}$) as free variables. In particular, for each sentence σ in the language $\mathbb{L}$,

$$\mathrm{L}_1 \models \sigma \quad \text{iff} \quad \mathrm{L}_2 \models \sigma.$$

PROOF. An immediate consequence of proposition 1.2. ∎

The definition of abstract logic does not impose any conditions on the consequence relation $\vdash$. If we set $C(\Gamma) = \{a \in X \ : \ \Gamma \vdash a\}$ then $C : Z \longrightarrow Z$ is the closure operator associated with L. Tarski's well-known axioms for a closure operator $C : \mathcal{P}(X) \longrightarrow \mathcal{P}(X)$ are the following (cf. [7]):

[C1] $A \subseteq C(A)$ for all $A \subseteq X$;
[C2] $A \subseteq B$ implies $C(A) \subseteq C(B)$ for all $A, B \subseteq X$;
[C3] $C(C(A)) \subseteq C(A)$ for all $A \subseteq X$.

In the language $\mathbb{L}$ this can be expressed as follows:

[A1] $(\forall Y)(\forall x)(x \in Y \rightarrow Y \vdash x)$;
[A2] $(\forall Y_1)(\forall Y_2)(Y_1 \subseteq Y_2 \rightarrow (\forall x)(Y_1 \vdash x \ \rightarrow \ Y_2 \vdash x))$;
[A3] $(\forall Y_1)(\forall x)((\exists Y_2)[(\forall y)(y \in Y_2 \ \leftrightarrow \ Y_1 \vdash y) \ \wedge \ Y_2 \vdash x] \ \rightarrow \ Y_1 \vdash x)$,

where $Y_1 \subseteq Y_2$ represents the formula $(\forall x)(x \in Y_1 \rightarrow x \in Y_2)$. As a consequence of Proposition 2.3, if T is a strongly conservative translation from L_1 to L_2, then L_1 satisfies [A1]-[A3] iff L_2 satisfies [A1]-[A3]. That is, L_1 is a logic (in the sense of [7]) iff L_2 is a logic in that sense (provided that $Z_i = \mathcal{P}(X_i)$, $i = 1, 2$).
In general, properties of logics will be expressed by means of sentences in $\mathbb{L}$, that is, theories $\mathbb{T}$ over $\mathbb{L}$. For example, the fact that Z contains every closed theory can be expressed as

$$(\forall Y_1)(\exists Y_2)(\forall x)[(x \in Y_2) \leftrightarrow (Y_1 \vdash x)].$$

As another example, a theory $\mathbb{T}$ may contain the sentence

$$(\forall Y_1)(\forall Y_2)(\forall x)(\forall y)[(Y_1 \vdash x) \ \wedge \ (Y_2 \cup \{x\} \vdash y) \ \rightarrow \ (Y_1 \cup Y_2 \vdash y)]$$

expressing the usual cut rule, which we call **CUT** property. Therefore, L_1 has **CUT** iff L_2 has **CUT**, provided that there exists a strongly conservative translation between them.

3 Ultraproducts of Logics

The main result, to be proved in this section, is that the ultraproduct of abstract logics is an abstract logic. We assume that the reader is familiar with the usual concept of filter, ultrafilter and ultraproduct.

THEOREM 3.1
Let $\mathrm{F} = \{\mathrm{A}_i \,:\, i \in I\}$ be a family of two-sorted structures over the same language $\mathbb{L}_1$, and let A be the ultraproduct of F over an ultrafilter $D \subseteq \mathcal{P}(I)$. Then

$$\mathrm{A} \models \sigma \quad \text{iff} \quad \{i \in I \,:\, \mathrm{A}_i \models \sigma\} \in D$$

for every $\mathbb{L}_1$-sentence σ.

If we conceive the elements of D as "large" subsets of I, the Fundamental Theorem of Ultraproducts 3.1 says that the ultraproduct A satisfies a sentence σ iff σ is satisfied by a "large" number of structures A_i.

We now prove the important fact that the class of abstract logics is closed under ultraproducts.

THEOREM 3.2 (Ultraproducts of Abstract Logics)
Let $\mathrm{F} = \{\mathrm{L}_i \,:\, i \in I\}$ be a family of abstract logics, where

$$\mathrm{L}_i = \langle X_i, Z_i, \in, \vdash_i, \cup, \{\cdot\}, \emptyset\rangle \qquad (i \in I).$$

Then the ultraproduct L of F over an ultrafilter $D \subseteq \mathcal{P}(I)$ is an abstract logic.

PROOF. The ultraproduct L' of F over D is given by

$$\mathrm{L}' = \langle X_D, Y_D, \in_D, \vdash_D, \cup_D, \{\cdot\}_D, \emptyset_D\rangle,$$

where $X_D = \Pi_D\{X_i \,:\, i \in I\}$, $Y_D = \Pi_D\{Z_i \,:\, i \in I\}$,

$$f_D \in_D \Gamma_D \quad \text{iff} \quad \{i \in I \,:\, f_i \in \Gamma_i\} \in D,$$

$$\Gamma_D \vdash_D f_D \quad \text{iff} \quad \{i \in I \,:\, \Gamma_i \vdash_i f_i\} \in D,$$

$$\Gamma_D \cup_D \Delta_D = \langle \Gamma_i \cup \Delta_i \,:\, i \in I\rangle_D = \{\Theta \,:\, \{i \in I \,:\, \Theta_i = \Gamma_i \cup \Delta_i\} \in D\},$$

$$\{f_D\}_D = \langle \{f_i\} \,:\, i \in I\rangle_D = \{\Gamma \,:\, \{i \in I \,:\, \Gamma_i = \{f_i\}\} \in D\},$$

$$\emptyset_D = \{\Gamma \,:\, \{i \in I \,:\, \Gamma_i = \emptyset\} \in D\}.$$

Consider the 5-uple $\mathrm{L} = \langle X_D, Z_D, \in, \vdash^D, \cup, \{\cdot\}, \emptyset\rangle$ defined as follows:

$Z_D \subseteq \mathcal{P}(X_D)$, where $S \in Z_D$ iff there exists $\Gamma \in \prod_{i \in I} Z_i$ such that

$S = \{f_D \in X_D \,:\, \{i \in I \,:\, f_i \in \Gamma_i\} \in D\} = S(\Gamma)$;

if $S = S(\Gamma)$, then $S \vdash^D f_D$ iff $\{i \in I \,:\, \Gamma_i \vdash_i f_i\} \in D$.

We assume that $\emptyset$, $\in$, $\cup$ and $\{\cdot\}$ are set-theoretic. We must prove that Z_D and $\vdash^D$ are well defined. Our argument is based on the following facts, whose proof we omit.

Fact a: If $f_D = g_D$ and $\Gamma \in \prod_{i \in I} Z_i$, then

$$\{i \in I \,:\, f_i \in \Gamma_i\} \in D \quad \text{iff} \quad \{i \in I \,:\, g_i \in \Gamma_i\} \in D.$$

Fact b: If $S(\Gamma) = S = S(\Delta)$, that is, if

$$\{f_D \in X_D \ : \ \{i \in I \ : \ f_i \in \Gamma_i\} \in D\} = \{f_D \in X_D \ : \ \{i \in I \ : \ f_i \in \Delta_i\} \in D\},$$

then $\Gamma_D = \Delta_D$ in Y_D.

Let $S = S(\Gamma) = S(\Delta) \in Z_D$ and $f_D \in X_D$. Then $\Gamma_D = \Delta_D$, by **Fact b**. Therefore

$\{i \in I \ : \ \Gamma_i \vdash_i f_i\} \in D$ iff $\Gamma_D \vdash_D f_D$
iff $\Delta_D \vdash_D f_D$ iff $\{i \in I \ : \ \Delta_i \vdash_i f_i\} \in D$.

This shows that $\vdash^D$ is a well defined relation over $Z_D \times X_D$, thus L is a two-sorted structure. We will prove that L is an abstract logic isomorphic to L$'$.

Let's define a map $S : Y_D \longrightarrow Z_D$, given by

$$S(\Gamma_D) = \{f_D \in X_D \ : \ \{i \in I \ : \ f_i \in \Gamma_i\} \in D\}.$$

The map S is well defined: if $\Gamma_D = \Delta_D$, then

$$R = \{i \in I \ : \ \Gamma_i = \Delta_i\} \in D.$$

On the other hand, suppose that $P = \{i \in I \ : \ f_i \in \Gamma_i\} \in D$. Then $R \cap P \in D$, that is,

$$\{i \in : \ f_i \in \Gamma_i \ \wedge \ \Gamma_i = \Delta_i\} \in D.$$

But clearly $R \cap P \subseteq \{i \in I \ : \ f_i \in \Delta_i\} = T$, thus $T \in D$. Analogously, $T \in D$ implies that $P \in D$. This proves that S is well defined. Moreover, since $S(\Gamma_D) = S(\Gamma)$ then S is injective, by **Fact b**, and S is clearly surjective. Thus, S is a bijective map. Consider $h = \langle id, S\rangle$, where $id : X_D \longrightarrow X_D$ is the identity map. We now prove that h is an isomorphism between L$'$ and L. In fact:

$f_D \in_D \Gamma_D$ iff $\{i \in I \ : \ f_i \in \Gamma_i\} \in D$ iff $id(f_D) \in S(\Gamma_D)$;
$\Gamma_D \vdash_D f_D$ iff $\{i \in I \ : \ \Gamma_i \vdash_i f_i\} \in D$ iff $S(\Gamma_D) \vdash^D id(f_D)$.

On the other hand,

$$S(\Gamma_D \cup_D \Delta_D) = \{f_D \ : \ \{i \in I \ : \ f_i \in \Gamma_i \cup \Delta_i\} \in D\}.$$

Let $f_D \in S(\Gamma_D \cup_D \Delta_D)$; then $R = \{i \in I \ : \ f_i \in \Gamma_i \cup \Delta_i\} \in D$. Let

$$P = \{i \in I \ : \ f_i \notin \Gamma_i\} \quad \text{and} \quad Q = \{i \in I \ : \ f_i \notin \Delta_i\}.$$

If $P \in D$ and $Q \in D$ then $P \cap Q \in D$, that is, $\{i \in I \ : \ f_i \notin \Gamma_i \cup \Delta_i\} \in D$, a contradiction (because $R \in D$). Thus $P \notin D$ or $Q \notin D$, whence

$$\{i \in I \ : \ f_i \in \Gamma_i\} \in D \quad \text{or} \quad \{i \in I \ : \ f_i \in \Delta_i\} \in D,$$

because D is an ultrafilter. This means that $f_D \in S(\Gamma_D) \cup S(\Delta_D)$. Conversely, if $f_D \in S(\Gamma_D)$, then $R' = \{i \in I \ : \ f_i \in \Gamma_i\} \in D$. But $R' \subseteq R$, thus $R \in D$, whence

$f_D \in S(\Gamma_D \cup_D \Delta_D)$; analogously, $f_D \in S(\Delta_D)$ implies that $f_D \in S(\Gamma_D \cup_D \Delta_D)$. That is, $S(\Gamma_D \cup_D \Delta_D) = S(\Gamma_D) \cup S(\Delta_D)$. Finally, observe that

$$\begin{aligned} S(\{f_D\}_D) &= \{g_D \ : \ \{i \in I \ : \ g_i \in \{f_i\}\} \in D\} \\ &= \{g_D \ : \ \{i \in I \ : \ g_i = f_i\} \in D\} = \{f_D\} = \{id(f_D)\}. \end{aligned}$$

Since $S(\emptyset_D) = \emptyset$, we infer that h is an isomorphism. Moreover, the proof above shows that Z_D satisfies:

$\emptyset \in Z_D$;
$\{f_D\} \in Z_D$ for every $f_D \in X_D$;
$\mathcal{S}, \mathcal{R} \in Z_D$ implies that $\mathcal{S} \cup \mathcal{R} \in Z_D$.

Since $X_i \in Z_i$ for all $i \in I$ then

$$\{i \in I \ : \ \mathrm{L}_i \models (\exists Y)(\forall x)(x \in Y \leftrightarrow (x = x))\} = I \in D,$$

therefore $\mathrm{L} \models (\exists Y)(\forall x)(x \in Y \leftrightarrow (x = x))$, by Theorem 3.1 and Proposition 1.2. That is, $X_D \in Z_D$. Since each Z_i satisfies item (iv) of Definition 2.1 then

$$\{i \in I \ : \ \mathrm{L}_i \models (\forall Y_1)(\exists Y_2)(\forall x)(x \in Y_2 \leftrightarrow x \notin Y_1)\} = I \in D,$$

therefore $\mathrm{L} \models (\forall Y_1)(\exists Y_2)(\forall x)(x \in Y_2 \leftrightarrow x \notin Y_1)$, by Theorem 3.1 and Proposition 1.2. That is, Z_D satisfies (iv). Finally, since each Z_i satisfies 2.1(v), then

$$\{i \in I \ : \ \mathrm{L}_i \models (\forall Y_1)(\exists Y_2)(\forall x)[(x \in Y_2) \leftrightarrow (Y_1 \vdash x)]\} = I \in D,$$

therefore $\mathrm{L} \models (\forall Y_1)(\exists Y_2)(\forall x)[(x \in Y_2) \leftrightarrow (Y_1 \vdash x)]$, by Theorem 3.1 and Proposition 1.2. That is, Z_D satisfies (v). Thus L is an abstract logic, and this concludes our proof. ∎

Corollary 3.3
If L is the ultraproduct of the family $\{\mathrm{L}_i\}_{i \in I}$ over D, then

$$\mathrm{L} \models \sigma \quad \text{iff} \quad \{i \in I \ : \ \mathrm{L}_i \models \sigma\} \in D$$

for every $\mathbb{L}$-sentence σ.

Proof. Immediate from Theorem 3.1. ∎

This means that the ultraproduct logic L satisfies a property σ (expressed in the language $\mathbb{L}$) iff the property σ is verified by a subset of logics L_i that is "large" (in some sense).

4 Some additional results from model theory

In this section we present some additional results from model theory which can be applied to our general approach to logics as structures. All new results (which appear as corollaries) are immediate, and the proofs are omitted.

DEFINITION 4.1
Let φ be a formula of a two-sorted language $\mathbb{L}'$.
(a) φ is a Π^0_1-*formula* if $\varphi = (\forall x_1 \cdots \forall x_n)\psi$, where each x_i is a variable (of sort $\mathbf{A_1}$ or sort $\mathbf{A_2}$) and ψ has no quantifiers.
(b) φ is a Π^0_2-*formula* if $\varphi = (\forall x_1 \cdots \forall x_n)(\exists y_1 \cdots \exists y_m)\psi$, where each x_i and y_j is a variable (of sort $\mathbf{A_1}$ or sort $\mathbf{A_2}$) and ψ has no quantifiers.
(c) φ is a *positive formula* if φ has no $\rightarrow$ and $\neg$ symbols.

PROPOSITION 4.2
Let $\mathbb{T}$ be a closed theory. If $\mathbb{T}$ is axiomatizable by Π^0_1-formulas, then $\mathbb{T}$ is preserved by submodels.

COROLLARY 4.3
Let L be a logic. If L satisfies a set $\mathbb{T}$ of Π^0_1-formulas, then every sublogic L$'$ of L also satisfies $\mathbb{T}$.

DEFINITION 4.4
A *chain of structures* is a sequence of $\mathbb{L}'$-structures $\mathcal{S} = \{\mathrm{A}_\gamma \ : \ \gamma < \alpha\}$ such that A_γ is a submodel of A_β (if $\gamma < \beta < \alpha$). The *union* A *of* $\mathcal{S}$ is the structure given by $A = \bigcup_{\gamma<\alpha} A_\gamma$, $B = \bigcup_{\gamma<\alpha} B_\gamma$, and the relations and functions are the union of the respective relations and functions of the structures in $\mathcal{S}$. The constants are unchanged.

PROPOSITION 4.5
Any set of Π^0_2-formulas is preserved by union of chains. □

COROLLARY 4.6
The union of a chain of logics is a logic. □

COROLLARY 4.7
If $\mathcal{S}$ is a chain of logics, each of them satisfying a set $\mathbb{T}$ of Π^0_2-formulas, then the union of $\mathcal{S}$ also satisfies $\mathbb{T}$. □

PROPOSITION 4.8
Let $\mathbb{T}$ be a consistent closed theory. Then, $\mathbb{T}$ is preserved by homomorphisms (that is: $\mathrm{A} \models \mathbb{T}$ implies that $\mathrm{B}_{|\langle F_1(A_1), F_2(A_2)\rangle} \models \mathbb{T}$ for every $\langle F_1, F_2\rangle : \mathrm{A} \longrightarrow \mathrm{B}$) iff $\mathbb{T}$ has a positive set of axioms. □

The following corollary establishes an important property of translations: the contents of positive sentences are preserved under translations.

COROLLARY 4.9
Given a translation $T : \mathrm{L} \longrightarrow \mathrm{L}'$, then

$$\mathrm{L} \models \varphi[a_1, ..., a_n; \Gamma_1, ..., \Gamma_m]$$

implies that

$$\mathrm{L}' \models \varphi[T(a_1), ..., T(a_n); T(\Gamma_1), ..., T(\Gamma_m)]$$

for every positive formula φ, every $(a_1, ..., a_n) \in X^n$ and every $(\Gamma_1, ..., \Gamma_m) \in Z^m$. Therefore, if a logic L satisfies a set $\mathbb{T}$ of positive properties, then, for every conservative translation $T : \mathrm{L} \longrightarrow \mathrm{L}'$, the image of L by T also satisfies $\mathbb{T}$.

5 Examples: first-order properties of abstract logics

As a direct application we show how general properties of abstract logics (or universal logics, cf. [2]) can be treated by our approach. From now on the expression $Y_1 \vdash Y_2$ will denote the formula $(\forall x)(x \in Y_2 \rightarrow Y_1 \vdash x)$.

Definition 5.1
Following [2] we define the following properties of abstract logics:

$[A]$ $(\forall x)(\{x\} \vdash x)$ (Auto-deductivity);

$[M\infty]$ $(\forall Y_1)(\forall Y_2)(\forall x)[((Y_1 \vdash x) \wedge (Y_1 \subseteq Y_2)) \rightarrow Y_2 \vdash x]$
(Infinite monotonicity);

$[Y\infty]$ $(\forall Y_1)(\forall Y_2)(\forall x)[((Y_1 \vdash Y_2) \wedge (Y_2 \vdash x)) \rightarrow Y_1 \vdash x]$
(Infinite syllogism);

$[AMY] \equiv_{def} [A] \wedge [M\infty] \wedge [Y\infty]$;

$[A\infty]$ $(\forall Y)(\forall x)(x \in Y \rightarrow Y \vdash x)$
(Infinite auto-deductivity);

$[H\infty]$ $(\forall Y_1)(\forall Y_2)(\forall Y_3)(\forall x)[((Y_1 \vdash Y_2) \wedge (Y_2 \cup Y_3 \vdash x)) \rightarrow Y_1 \cup Y_3 \vdash x]$
(Infinite Herz's law);

$[N]$ $(\forall Y_1)(\forall x)[(Y_1 \vdash x) \leftrightarrow (\forall Y_2)((Y_2 \vdash Y_1) \rightarrow (Y_2 \vdash x))]$
(Normal law).

Proposition 5.2
(i) $[A\infty]$ and $[Y\infty]$ implies $[M\infty]$.
(ii) $[A\infty]$ implies that $[Y\infty]$ is equivalent to $[H\infty]$.
(iii) $[N]$ is equivalent to $[AMY]$.

Proof. By adapting the proofs of these results found in [2] to our first-order formalism. ∎

6 Fibring propositional calculi

Finally, we will show that the process of fibring propositional calculi introduced by Sernadas et al. (cf. [8]) can also be expressed in our framework.

Definition 6.1
Let $C = \{C_k\}_{k \in \mathbb{N}}$ be a (propositional-based) signature (cf. [8]). That is, each C_k is a set of connectives of arity k. Connectives in C_0 are *propositional symbols*. Let $\mathbb{L}_C$ be the language obtained from $\mathbb{L}$ and C, considering each $f \in C_k$ as a symbol of function $f : \mathbf{A_1}^k \longrightarrow \mathbf{A_1}$. In particular, each $p \in C_0$ is a symbol of constant of sort $\mathbf{A_1}$. A *Hilbert calculus specification* is a pair $\langle C, R \rangle$ where C is a (propositional-based) signature and R is a set of sentences in the language $\mathbb{L}_C$ of the form

$$(\forall)(\{\xi_1, ..., \xi_n\} \vdash \xi) \quad \text{and} \quad (\forall)(\emptyset \vdash \xi),$$

where $(\forall)\Phi$ denotes the universal closure of the formula Φ, and $\{\xi_1, ..., \xi_n\}$ denotes the term $(\cdots(\{\xi_1\} \cup \cdots) \cup \{\xi_n\})$. Given a specification $E = \langle C, R\rangle$ we obtain the logic $\mathrm{L}_E = \langle X, Z, \in, \vdash, \cup, \{\cdot\}, \emptyset, C\rangle$ over $\mathbb{L}_C$ defined as follows:

- X is the free algebra generated by C;
- $Z = \mathcal{P}(X)$;
- $\vdash = \bigcap\{\vdash' \subseteq Z \times X \mid \langle X, Z, \in, \vdash', \cup, \{\cdot\}, \emptyset, C\rangle \models R \cup \{[A\infty], [Y\infty]\}\}$.

Observe that, by proposition 5.2, $\mathrm{L}_E \models [M\infty]$. Given $\Gamma \cup \{\alpha\} \in Z$ we say that α *is provable from* Γ *in* L_E if $\mathrm{L}_E \models (Y_1 \vdash x_1)[\Gamma, \alpha]$.

The *unconstrained fibring* of two logics L_1 and L_2 over $\langle C^1, R_1\rangle$ and $\langle C^2, R_2\rangle$ is defined naturally as $\mathrm{L} = \mathrm{L}_1 \vee \mathrm{L}_2$, the logic obtained from the specification $\langle C, R\rangle$, where $C = C^1 \sqcup C^2$ is the (disjoint) union of C^1 and C^2 and $R = R_1 \cup R_2$. It is easy to see that $\mathrm{L}_1 \vee \mathrm{L}_2$ is the least logic L over $\mathbb{L}_C$ satisfying $\mathrm{L}_i \subseteq \mathrm{L}_{|\mathbb{L}_{C^i}}$ for $i = 1, 2$, that is: if $\mathrm{L}_i \subseteq \mathrm{L}'_{|\mathbb{L}_{C^i}}$ $(i = 1, 2)$ then $\mathrm{L} \subseteq \mathrm{L}'$.

The *constrained fibring* of L_1 and L_2, obtained by sharing connectives (cf. [8]), is obtained analogously. The generalisation to arbitrary families of logics is straightforward.

7 Concluding remarks

We have introduced a formal framework in which abstract logics and their translations can be analysed and (hopefully) better understood. Our definitions and main results show that:

(1) most properties of abstract logics can be expressed and proved in our formalism;

(2) classical results of model theory can be used to establish important (meta)properties of translations between abstract logics, helping to assign a philosophical meaning to the concept of translation;

(3) interesting results from the point of view of combinations of abstract logics can also be obtained. In particular, we have shown that the category of abstract logics is closed under ultraproducts and that the process of fibring is also expressable in the proposed formalism.

References

[1] J. L. Bell, A. B. Slomson, **Models and Ultraproducts: an introduction**. North Holland, 1971.

[2] J. Y. Béziau, *Recherches sur la Logique Universelle (Excessivité, Négation, Séquents)*. Doctoral Thesis, Paris 7 (1994).

[3] D. J. Brown, R. Suszko, *Abstract Logics*. Dissertationes Mathematicae **102** (1973), pp. 9–41.

[4] W. A. Carnielli, *Possible-Translations Semantics for Paraconsistent Logics*. In **Frontiers of Paraconsistent Logic**, pp. 159172 . D. Batens, C. Mortensen, G. Priest and J. P. Van Bendegem, eds. Logic and Computation Series, 2000.

[5] W. A. Carnielli, I. M. L D'Ottaviano, *Translations between Logical Systems: a Manifesto*. Logique et Analyse **157** (1997), pp. 67–81.

[6] C. C. Chang, H. J. Keisler, **Model Theory**. North Holland, 1990.

[7] J. J. da Silva, I. M. L. D'Ottaviano, A. M. Sette, *Translations between Logics*. In **Models, Algebras and Proofs**, pp. 435–448. X. Caicedo and C.H. Montenegro, eds. Marcel Dekker, 1999.

[8] A. Zanardo, A. Sernadas, C. Sernadas, *Fibring: Completeness Preservation*. To appear in *The Journal of Symbolic Logic*.

Elementary number theory in weak fragments of arithmetic (abstract)

PAOLA D'AQUINO, *Dipartimento di Matematica, Seconda Università degli Studi di Napoli, Napoli, Italy. E-mail: paola.daquino@unina2.it*

Abstract

I will consider some classical results about primes in non standard models of $I\Delta_0$ and its extensions. In particular, I will study the residue fields of models of $I\Delta_0 + \Omega_1$ (where Ω_1 is the axiom guaranteeing that the function $(x, y) \mapsto x^{[\log y]}$ is total). I will prove that the residue fields of such models have a unique extension of finite degree n, for each $n \in \mathbf{N}$, and that the Galois group of such extensions are cyclic.

I will then extend these results to quotient fields of models of $I\Delta_0 + \Omega_1$.

References

[1] D'Aquino, P. and Macintyre, A., Non-standard finite fields over $I\Delta_0 + \Omega_1$, in *Israel Journal of Mathematics* 117 (2000), pp. 311–333.

[2] D'Aquino, P., Quotient fields of models of $I\Delta_0 + \Omega_1$, submitted.

Reasoning about Dynamic Databases: Tableaux for Single-Step-Update Models

CHRISTIAN G. FERMÜLLER, *Institut für Computersprachen, Technische Universität, Karlsplatz 13/E185.2, A-1040 Vienna, Austria. E-mail: chrisf@logic.at*

Abstract

We study the dynamic behaviour of step-wise evolving databases using concepts from modal and many-valued logic. We focus on modelling "single-step", i.e., atomic update operations, which change the truth value of at most one basic unit of information at a time. The resulting logics form a variant of standard modal logics. These "update logics" are generalized to a many-valued context to take into account possibly incomplete and inconsistent data. In particular, Belnap's well known four-valued logic is combined with the concept of distribution modalities and underlying Kripke structures. A special family of signed and prefixed analytic tableaux is introduced as an adequate tool for proof search in this context.

1 Introduction and summary

Various approaches employing modal logics for both the representation of knowledge and mechanized reasoning about data have been investigated. See, e.g., [3, 14, 6, 10, 9] for some recent work of relevance to database theory.

Here we introduce a model of dynamic databases that is particularly simple. The states of a database are identified with assignments of truth values to basic propositions. Some states are considered as results of updating other states of the database. In other words, a binary *update relation* is defined over the set of possible states. This clearly amounts to defining usual Kripke stuctures. Standard modal logics arise if we augment classical propositional logic over the signature of basic propositions with the modalities $\Box$ and $\Diamond$ (interpreted as "in all updated states" and "in some updated state", respectively).

Literature on the so-called "update problem" usually aims at formalizing changes in databases triggered by *arbitrary complex* changes in the environment to which the database refers. Here however, we want to model only *atomic* or *"single-step" updates.* More exactly, each update operation is assumed to change the truth value of at most one basic proposition at a time. In general, single-step updates reflect this type of adaption to a changing environment only via *sequences* of such atomic update operations. However, we think that considering single-step updates leads to a *more realistic* model of the actual computational behaviour of dynamic databases at a less abstract level. Certainly the evolution of any database proceeds in basic steps, each of which corresponds to some well defined atomic action that can be performed on single data. We aim at a (conceptually as well as technically) *simple* logical model of this aspect of dynamic databases.

The subtle computational constraint on the update relation — namely: to consider only atomic update operations — seems to have dramatic effects: the set of formulae valid in all corresponding Kripke models is not closed under substitution. In response to this fact we propose to use a *two sorted* propositional language that allows to distinguish between "atoms" (basic propositions of the database) and genuine propositional variables; and consequently between "concrete" and "schematic" statements about data. We define a corresponding semantics and provide complete and sound *tableau calculi* for the resulting logics.

Moreover, we generalize the just described model of atomic dynamic databases to scenarios allowing for incomplete and inconsistent information about (atomic as well as complex) propositions. Replacing classical logic by Belnap's well-known four-valued logic [4] opens the space for new types of modal operators over corresponding update models. Some examples of such *distribution modalities* expressing properties of updates will be investigated.

A variant of the tableaux for finite-valued logics with distribution modalities introduced in [7] turns out to be adequate for formalizing reasoning in corresponding logics.

A main motivation for presenting these ideas at this particular workshop is that all four areas of research adressed by it's name — Logic, Language, Information and Computation — are in an obvious and essential way involved in the proposed approach to reasoning about dynamic databases.

We emphasize that the concepts and results presented here should be considered only as a first step in exploring the scope and limits of many-valued Kripke structures and distribution modalities in the context of reasoning about dynamic databases. Accordingly, we conclude with a list of future topics of research.

2 Atomic databases and update models

Our first object of investigation is argueably the most simple logical model of a database: it refers to a fixed set of basic propositions (called *atoms*) and presumes that the only information explictly contained in the database is which of those atomic propostions hold and which do not hold. More formally, by a *(classical) state (of a database)* we mean a total function of type $\mathsf{atoms} \mapsto \{\mathbf{t}, \mathbf{f}\}$, where atoms is a non-empty, finite set of propositional atoms.[1] Obviously we can evaluate arbitrary propostional formulae over the signature atoms with respect to a state α as usual:

- $v_\alpha(p) = \alpha(p)$, for $p \in \mathsf{atoms}$
- $v_\alpha(\top) = \mathbf{t}$
- $v_\alpha(\neg A) = \mathbf{t}$ iff $v_\alpha(A) = \mathbf{f}$
- $v_\alpha(A \circ B) = \widetilde{\circ}(v_\alpha(A), v_\alpha(B))$

where $\circ \in \{\wedge, \vee, \supset\}$ and $\widetilde{\circ}$ is the classical truth function associated with the binary connective $\circ$.

State α' is called a *single-step-update* of a state α if the following condition is satisfied:

(ss) $\alpha(p) = \alpha'(p)$ for all *except at most one* $p \in \mathsf{atoms}$.

[1] Throughout this paper we will consider the set of atoms to be fixed for any given database and may therefore surpress reference to it in other definitions.

States of a database can be viewed as worlds of a Kripke model, where a world β is accessible from a world α iff β is a single-step-update of α.

DEFINITION 2.1
A *single-step-update model* is a pair $\mathcal{M} = (\Sigma, U)$ where

- Σ is a set of states of a database over (a fixed set) atoms, i.e., a set of functions of type $\mathsf{atoms} \mapsto \{\mathbf{t}, \mathbf{f}\}$, and
- U is a binary relation over S, subject to the restriction that $\forall \alpha, \alpha' \in \Sigma\colon \alpha U \alpha'$ implies that α' is a single step update of α.

We augment the propostional language by the modal operators $\Box$ and $\Diamond$, with the intended meaning "in all updates" and "in some update", respectively. More exactly—referring to states α of a single-step-update model $\mathcal{M} = (\Sigma, U)$—we extend the definition of v_α as follows:

- $v_\alpha^{\mathcal{M}}(\Box A) = \mathbf{t}$ iff $\forall \beta \in \Sigma\colon \alpha U \beta$ implies $v_\beta^{\mathcal{M}}(A) = \mathbf{t}$, and
- $v_\alpha^{\mathcal{M}}(\Diamond A) = \mathbf{t}$ iff $\exists \beta \in \Sigma\colon \alpha U \beta$ and $v_\beta^{\mathcal{M}}(A) = \mathbf{t}$.

This simple logical formalism—close to familiar model structures—allows to express statements that refer not only to the current state of a database but also to states that are the result of an update.

EXAMPLE 2.2
The formula $A \supset \Box A$ may be paraphrased as "If the statement A is currently supported by the database, then all possible updates will still support A". Similarly $\Diamond A \wedge \Diamond \neg A$ expresses that A is a statement that will be evaluated differently in different updates of the database. Likewise we can express that there is no possible update by "$\neg\Diamond\top$". The statement that for every update (of the current state) a further update is possible is expressed by "$\Box\Diamond\top$" etc.

By requiring the update relation to fulfill simple properties we can adapt our model to databases which obey certain dynamic constraints. E.g., requiring the update relation U to be symmetric corresponds to modelling databases for which every update is reversable. Similary in many applications it will be useful to require U to be reflexive or serial.

DEFINITION 2.3
A single-step-update model is called *update* B-, D-, T-, or TB-*model* if its update relation is symmetric, serial, reflexive, or symmetric and reflexive, respectively. The class of *all* single-step-update models is called *update* K-*models*.

3 Concrete versus schematic statements

It might seem as if—so far—we have only described just another interpretation of normal modal logic. However, in insisting that the truth value of *only one* atom can be changed in one update operation we ensured that, e.g., the formula

$$F_1 = (p \wedge q) \supset \Box(p \vee q)$$

is evaluated true in all states of all single-step-update models, if p and q are different atoms. In contrast, substituting in F_1 p for q results in a formula which is false at all states which update p from $\mathbf{t}$ to $\mathbf{f}$. In other words, the set of formulae that are true at all points of a single-step-update model is not closed under substitution.

There is a simple way of recovering closure under substitution:

DEFINITION 3.1
A formula A is *schematically valid* in a single-step-update model $\mathcal{M} = (\Sigma, U)$ if for all substitution instances A' of A and all $\alpha \in \Sigma$: $v_\alpha^M(A') = \mathbf{t}$.

The set of formulae that are schematically valid in all update $\mathcal{L}$-models is called *update-$\mathcal{L}$* (for $\mathcal{L} \in \{\mathsf{K}, \mathsf{B}, \mathsf{D}, \mathsf{T}, \mathsf{TB}\}$).

By definition these update-$\mathcal{L}$ are closed under substitution. It is easy to see that they are closed under modus ponens and the necessity rule, as well. Indeed we have the following result:

THEOREM 3.2
For $\mathcal{L} \in \{\mathsf{K}, \mathsf{B}, \mathsf{D}, \mathsf{T}, \mathsf{TB}\}$, update-$\mathcal{L}$ and (ordinary) $\mathcal{L}$ coincide.

REMARK 3.3
This result can be viewed as a strengthening of completeness for the standard normal logics $\mathcal{L}$. It says that for every non-valid formula F there is a counter-model of an instance of F that obeys the single-step-update restriction **(ss)**. Indeed, the proof of the theorem consists in an explicit construction of such a substitution instance and its corresponding update counter-model.

We are interested in reasoning about changing databases both, at the level of schematic statements and by evaluating statements referring to concrete atoms of a database. Theorem 3.2 tells us that we remain within standard normal modal logics as long as only schematic statements are considered.[2] In order to be able to refer to the schematic as well as the concrete level simultaneously we define the languange $\mathcal{UL}$ over a two-sorted signature:

- An *atomic formula* of $\mathcal{UL}$ is either an element $p \in \mathsf{atoms}$ or a propositional variable[3] or $\top$.
- *Complex formulae* of $\mathcal{UL}$ are built up as usual from the atomic formulae using the connectives $\neg, \wedge, \vee, \supset$ and the modalities $\Box$, $\Diamond$.

A formula of our extended language is called *concrete* if it does not contain propositional variables. Otherwise, it is called *schematic*. For concrete formulae F $v_\alpha^{\mathcal{M}}(F)$ is defined as in Section 2. An arbitrary (possibly schematic) formula F is called *valid* in $\mathcal{M}$ if for all concrete formulae F', that arise by substituting the propositional variables of F with concrete formulae we have $v_\alpha^{\mathcal{M}}(F') = \mathbf{t}$ for all states α of $\mathcal{M}$.

Notation. We use lower case letters for atoms. Different letters always denote different atoms. Propositional variables are denoted by upper case letters from the end of the alphabet.

EXAMPLE 3.4
The concrete formula $(a \wedge \neg b) \supset \Diamond(b \supset a)$ is valid in all update models. However the schematic formula $(X \wedge \neg b) \supset \Diamond(b \supset X)$ is not valid in most update models.

[2] By a schematic statement we mean one in which the propositional variables range over all possible (non-modal and modal) statements over the signature atoms.

[3] The set of propositional variables and atoms are disjoint.

The concrete formula $\phi = \Diamond(a \wedge b) \wedge \Diamond(a \wedge \neg b) \wedge \Diamond(\neg a \wedge b) \wedge \Diamond(\neg a \wedge \neg b)$ can never evaluate to $\mathbf{t}$. In other words $\neg\phi$ is valid in all update models. In contrast, it is easy to find counter models for $\Box(X \vee Y) \vee \Box(X \vee \neg Y) \vee \Box(\neg X \vee Y) \vee \Box(\neg X \vee \neg Y)$.

4 Prefixed tableaux adapted to update models

We have so far defined adequate syntax and semantics of a language that allows to express various statements with respect to changing databases (of a particularly simple type). To substantiate the claim that this formalism provides a basis for *reasoning* we have to define sound and complete calculi. Often one is content with Hilbert style axiomatizations, however analytic (Gentzen style) calculi are certainly much more adequate if we aim at *automatic* deduction. Fortunately, Fitting's analytic *prefixed tableaux* [8] for standard normal logics turn out to be easily adaptable to our scenario. (See also [12] for an overview and history of related methods.)

We assume familiarity with tableaux but review some relevant terminology.

A *prefix* is a finite sequence of natural numbers intended to denote a world of a Kripke structure. A *prefixed formula* is a pair consisting of a prefix σ and formula F written as $\sigma :: F$. *Prefixed tableaux* are downward rooted trees of prefixed formulae, generated by appending new prefixed formula to a branch of a tree according to three types of rules.

Non-modal rules:
The rules for negation and disjunction are as follows:

$$(\neg\neg)\ \frac{\sigma :: \neg\neg F}{\sigma :: F} \qquad (\vee)\ \frac{\sigma :: F \vee G}{\sigma :: F \ \mid\ \sigma :: G} \qquad (\neg\vee)\ \frac{\sigma :: \neg(F \vee G)}{\begin{array}{c}\sigma :: \neg F\\ \sigma :: \neg G\end{array}}$$

The rules for conjunction and implication are similar.

Modal rules:
The rules for analyzing the modality $\Box$ in the basic modal logic K are

$$(\mathrm{K})\ \frac{\sigma :: \Box F}{\sigma.x :: F} \qquad (\pi)\ \frac{\sigma :: \neg\Box F}{\sigma.n :: \neg F}$$

where n is such that the prefix $\sigma.n$ is new to the current branch. The K-rules for $\Diamond$ are dual.
For serial, reflexive, and symmetric models we have to add the following rules, respectively:

$$(\mathrm{D})\ \frac{\sigma :: \Box F}{\sigma :: \Diamond F} \qquad (\mathrm{T})\ \frac{\sigma :: \Box F}{\sigma :: F} \qquad (\mathrm{B})\ \frac{\sigma.x :: \Box F}{\sigma :: F}$$

Closure rules:
The closure rules for standard modal logics are

$$\frac{\begin{array}{c}\sigma :: \neg F\\ \sigma :: F\end{array}}{\mathtt{closed}} \qquad \frac{\sigma :: \neg\top}{\mathtt{closed}}$$

A tableau is *closed* if all its branches are closed. It is well known (see, e.g., [8, 12]) that the calculus sketched above is sound and complete for the respective modal logics. I.e., there exists a closed tableau constructed according to the above rules starting with root $0 :: \neg F$ if and only if F is valid in the corresponding modal logic K, B, D, T, or TB.

To accomodate for the difference between atoms and propositional variables in the language $\mathcal{UL}$ as well as for the single-step-update condition in update models it suffices to extend the standard tableau calculi by additional closure rules.

Single-Step closure rules:

$$\frac{\begin{array}{c}\sigma :: a\\ \sigma.n :: \neg a\\ \sigma :: b\\ \sigma.n :: \neg b\end{array}}{\texttt{closed}} \qquad \frac{\begin{array}{c}\sigma :: \neg a\\ \sigma.n :: a\\ \sigma :: b\\ \sigma.n :: \neg b\end{array}}{\texttt{closed}} \qquad \frac{\begin{array}{c}\sigma :: a\\ \sigma.n :: \neg a\\ \sigma :: \neg b\\ \sigma.n :: b\end{array}}{\texttt{closed}} \qquad \frac{\begin{array}{c}\sigma :: \neg a\\ \sigma.n :: a\\ \sigma :: \neg b\\ \sigma.n :: b\end{array}}{\texttt{closed}}$$

where a and b are different atoms.

A tableau constructed according to the above rules is called *update $\mathcal{L}$-tableau*, where $\mathcal{L}$ refers on of the logics K, B, D, T, or TB, and the corresponding modal rules.

THEOREM 4.1
For $\mathcal{L} \in \{\mathsf{K}, \mathsf{B}, \mathsf{D}, \mathsf{T}, \mathsf{TB}\}$: an $\mathcal{UL}$-formula F is valid in all update $\mathcal{L}$-models iff there exists a closed *update* $\mathcal{L}$-tableau with root $0 :: \neg F$.

5 Incomplete and inconsistent data

Our model of the dynamic behaviour of a database is yet to simple to capture phenomena like possibly incomplete and inconsistent data. However, we claim that the basic formalism of single-step update models and corresponding tableaux is easily adapted to such scenarios.

Belnap's four-valued logic [4] has repeatedly been suggested as a tool for reasoning about (possibly) inconsistent and incomplete information. The main intuition in this context is that a database may not only contain information implying that a statement is *false* or *true*, but such information may also be absent or inconsistent. (I.e., the database might assign none or both classical truth values **t**, **f** to an atom.) The four possible states of knowledge are represented by the four truth values **f** (*false*), **u** (*undetermined*), $\bot$(*inconsistent*), and **t** (*true*), respectively. This intended interpretation induces the following truth functions for the connectives $\neg$, $\wedge$, and $\vee$:

	$\neg$
f	**t**
u	**u**
$\bot$	$\bot$
t	**f**

$\wedge$	**f**	**u**	$\bot$	**t**
f	**f**	**f**	**f**	**f**
u	**f**	**u**	**f**	**u**
$\bot$	**f**	**f**	$\bot$	$\bot$
t	**f**	**u**	$\bot$	**t**

$\vee$	**f**	**u**	$\bot$	**t**
f	**f**	**u**	$\bot$	**t**
u	**u**	**u**	**t**	**t**
$\bot$	$\bot$	**t**	$\bot$	**t**
t	**t**	**t**	**t**	**t**

For the definition of other connectives (in particular forms of implication) and the choice of designated truth values we refer to the extensive investigations of Avron and Arieli (see, e.g., [2, 1]). Here we follow a remark in [7] and argue that Kripke structures based on four-valued "Belnap-worlds" (instead of two-valued worlds) are adequate for modelling changing databases.

The many-valued context allows to extend the classical universal and existential modalities to the more general concept of *distribution modalities*, introduced in [7]. Let $\mathcal{V}$ be the set of truth values and, correspondingly, a world of a (many-valued)

Kripke-model an assignment α : $\mathsf{atoms} \mapsto \mathcal{V}$. Then any function $\widetilde{\mu}$ of type $2^{\mathcal{V}} \mapsto \mathcal{V}$ induces a truth function of a distribution modality μ by:

$$v_\alpha^{\mathcal{M}}(\mu F) = \widetilde{\mu}(\{v_\beta^{\mathcal{M}}(F) \mid \beta \in W : \alpha R \beta\}).$$

Here W is the set of worlds of the model $\mathcal{M}$ and R its accessibility relation. $\{v_\beta^{\mathcal{M}}(F) \mid \beta \in W : \alpha R \beta\}$ is called *distribution* of F in $\mathcal{M}$ at α. This definition directly transfers to update models based on many-valued assignments. In particular, we call the update models in which the states of the database consist in assignments of type $\mathsf{atoms} \mapsto \{\mathbf{t}, \mathbf{u}, \mathbf{f}, \bot\}$ *Belnap update structures.* This context allows us to define modalities like

- $\mathit{det}(F)$ with the intended meaning: "No update renders information on F incomplete or inconsistent", and
- $\mathit{unif}(F)$ with the intended meaning: "F is evaluated uniformly in all updates".

Since $\mathit{det}(F)$ is intended to express a meta-linguistic (and therefore classical) property of F within the object language itself, it always evaluates to $\mathbf{t}$ or $\mathbf{f}$. More exactly, its semantics is fixed by:

$$\widetilde{\mathit{det}}(W) = \begin{cases} \mathbf{t} & \text{if } W = \emptyset, \{\mathbf{f}\}, \{\mathbf{t}\}, \text{or } \{\mathbf{t}, \mathbf{f}\} \\ \mathbf{f} & \text{otherwise} \end{cases}$$

On the other hand "uniform evaluation" admittedly is an ambigous concept. Certainly, we want $\mathit{unif}(F)$ to be *true* if set of all truth values to which F is evaluated in some update is $\{\mathbf{f}\}$ or $\{\mathbf{t}\}$. Likewise, it is clear that $\mathit{unif}(F)$ is *false* if the distribution contains $\mathbf{t}$ *and* $\mathbf{f}$ (i.e., if there is an update evaluating the formula to $\mathbf{t}$, but also another update that evaluates it to $\mathbf{f}$.) But we want $\mathit{unif}(F)$ to be *undetermined* if the distribution of F contains $\mathbf{u}$. One way to round off and formalize these intuitions is to define the truth function for unif as follows:

$W \subseteq \mathcal{V}$	$\widetilde{\mathit{unif}}(W)$
$\emptyset, \{\mathbf{f}\}, \{\mathbf{t}\}$	$\mathbf{t}$
$\{\mathbf{u}\}, \{\mathbf{u}, \mathbf{f}\}, \{\mathbf{u}, \mathbf{t}\}$	$\mathbf{u}$
$\{\mathbf{f}, \mathbf{t}\}, \{\mathbf{f}, \mathbf{u}, \mathbf{t}\}, \{\mathbf{f}, \bot, \mathbf{t}\}, \{\mathbf{f}, \mathbf{u}, \bot, \mathbf{t}\}$	$\mathbf{f}$
$\{\bot\}, \{\mathbf{f}, \bot\}, \{\bot, \mathbf{t}\}, \{\mathbf{u}, \bot\}, \{\mathbf{f}, \mathbf{u}, \bot\}, \{\mathbf{u}, \bot\, \mathbf{t}\}$	$\bot$

Of course, det and unif are just two simple examples. Observe that there are 4^{2^4} possible distribution modalities definable over Belnap update structures. All of them refer to properties of the "truth status" of statements with respect to the class of possible stepwise evolutions of the database. We claim that at least some of those properties are of interest in the context of modelling the logical structure of dynamic databases.[4] Moreover, we remind the reader that Belnap update models come in different variants according to the constraining properties of the update relation.

To define a particular *Belnap update logic* with respect to a class of Belnap update models we therefore have to fix three independent parameters (in addition to the set of atoms and propositional variables):

(1) a set of designated truth values $\mathcal{D}$; usually $\{\mathbf{t}\}$ or $\{\mathbf{t}, \bot\}$

[4] Space limits do not allow us to investigate more distribution modalities from this point of view, here. We plan to do so in a longer version of this paper.

(2) a set of $\{\mu_1, \ldots, \mu_n\}$ of distribution modalities (with associtated truth functions $\widetilde{\mu_1}, \ldots, \widetilde{\mu_n}$ and four-valued connectives (specified by their truth tables)

(3) properties like symmetry or reflexivity, which we want the update relation to observe.

We call a formula F *valid* in such a logic if $v_\alpha^{\mathcal{M}}(F) \in \mathcal{D}$ for all states α of all corresponding single-step update models $\mathcal{M}$.

6 Prefixed signed tableaux for Belnap update models

It is well known that appropriate analytic calculi for all (truth functional) finite valued logics can be defined using *signed* versions of tableaux (see, e.g., [13]). These can be extended to finite valued modal logics by combining prefixes (denoting worlds) and signs (denoting truth values) as was shown in [7]. We describe a somewhat simplified variant of the latter calculi, adapted for update structures, for the special case of Belnap update models with serial (but otherwise general) update relation.[5]

A prefixed signed formula is a triple consisting of a finite sequence of natural numbers σ (called prefix), a truth value v, and a formula F, written as $\sigma\colon [v]\colon F$.

REMARK 6.1
In classical logic the prefixed signed formulae $\sigma\colon [\mathbf{t}]\colon F$ and $\sigma\colon [\mathbf{f}]\colon F$ are just notational variants of the prefixed formulae $\sigma :: F$ and $\sigma :: \neg F$, respectively. For many-valued logics truth value signs are not only an elegant way to make semantic information explicit but are, in general, *needed* to obtain complete tableau calculi.

Again, a (prefixed signed) tableau is a downward rooted tree of prefixed signed formulae, constructed using the following rules.

Non-modal rules: can directly be read off from the truth tables of connectives. We refer to [13, 15] for general methods and results about constructing optimal rules.

Closure rules: The standard closure rule is

$$\frac{\begin{array}{c}\sigma\colon [v]\colon F\\ \sigma\colon [w]\colon F\end{array}}{\texttt{closed}}$$

where v and w are different truth values.

A modal operator μ induces an additional closure rules if a formula μF never evaluates to a particular truth value. E.g., the modality *det* triggers the following two closure rules:

$$\frac{\sigma\colon [\mathbf{u}]\colon det(F)}{\texttt{closed}} \qquad \frac{\sigma\colon [\bot]\colon det(F)}{\texttt{closed}}$$

Modal rules:

A general method for constructing modal rules from associated truth functions is described in [7]. We represent the two remaining cases of *det*-modalized formulae:

$$\frac{\sigma\colon [\mathbf{t}]\colon det(F)}{\begin{array}{c}\sigma.n\colon [\mathbf{t}]\colon F\\ \sigma.m\colon [\mathbf{f}]\colon F\end{array}} \qquad \frac{\sigma\colon [\mathbf{f}]\colon det(F)}{\sigma.n\colon [\mathbf{u}]\colon F \;\Big|\; \sigma.n\colon [\bot]\colon F}$$

[5] Other properties of the update relation result in simple technical variations of some modal rules. The corresponding calculi are omitted here for space reasons.

where $\sigma.n$ and $\sigma.m$ are different (but not necessarily new on the branch).

Single-step closure rule:

$$\frac{\begin{array}{c}\sigma : [u_1] : a \\ \sigma.n : [v_1] : a \\ \sigma : [u_2] : b \\ \sigma.n : [v_2] : b\end{array}}{\texttt{closed}}$$

where a and b are different atoms and $v_i \neq u_i$ for $i \in \{1, 2\}$.

THEOREM 6.2
A formula F is valid in a Belnap update logic if and only if for all non-designated truth values v there exists a corresponding update tableau with root $0 : [v] : F$ that is closed.

It turns out that, for all mentioned variants of update logics, a systematic, exhaustive and terminating tableau construction procedure can be defined. This implies the *decidability* of these logics.

7 Open ends

We want to emphasize that the framework of update models, distribution modalities and corresponding tableaux is a flexible general tool rather than just another special non-classical formalism. To this aim we conclude with an (incomplete) list of topics for future research in this context.

Different underlying many-valued logics. Update models can be defined over all kinds of truth functional logics as mechanism for "local" evaluation. As an interesting example we mention the bi-lattice based logics suggested by M.L. Ginsberg [11] for modelling default reasoning. Also dynamic *fuzzy* databases can be modelled by building on an appropriate fuzzy logic (e.g., some finite-valued Łukasiewicz logic).

Other useful distribution modalities. As explained above, every function of type $2^{\mathcal{V}} \mapsto \mathcal{V}$ induces a distribution modality. A systematic investigation of expressibility, complexitiy of corresponding rules and functional dependency between different sets of modalities is still lacking.

Modelling global update constraints. Many kinds of general integrity constraints can easily be modelled by imposing further restrictions on the update relation between states. As a simple example consider the condition—for Belnap update models—that updates can only *increase knowledge* about data. Technically this corresponds to requiring $\alpha(a) \leq_k \beta(a)$ if $\alpha U \beta$, where $\leq_k$ is the partial "knowledge order" defined by $\mathbf{u} \leq_k \mathbf{t} \leq_k \bot$ and $\mathbf{u} \leq_k \mathbf{f} \leq_k \bot$.

Other types of update operations. Obviously single-step-updates as defined by condition **(ss)** (Section 2) are only a special case.[6] One might, e.g., study multiple update relations that are indexed by the "new information" that triggers the update. This information is often represented by a boolean combination of atoms and thus naturally induces a corresponding algebra of update relations.

[6] For a thorough computational analysis of different related concepts see [5].

First-order reasoning. Both, update models and corresponding tableau, are readily generalized to the first-order level. This move, of course, vastly improves the expressibility of the corresponding logics. Their strength and limits should also be explored.

References

[1] Ofer Arieli and Arnon Avron. The logical role of the four-valued bilattice. In *13th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 118–126, Indianapolis (USA), 1998. IEEE Computer Society.

[2] Ofer Arieli and Arnon Avron. The value of the four values. *Artificial Intelligence*, 102(1):97–141, 1998.

[3] Philippe Balbiani. A modal logic for data analysis. In Wojciech Penczek and Andrzej Szalas, editors, *Proc. Mathematical Foundations of Computer Science, 21st MFCS*, volume 1115 of *Lecture Notes in Computer Science*, pages 167–179, Cracow, Poland, 1996. Springer-Verlag.

[4] Nuel D. Belnap. A useful four-valued logic. In J.M. Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 8–37. D. Reidel Publishing Co., 1977.

[5] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.

[6] Dieter Fensel, Rix Groenboom, and G.R.Renardel de Lavalette. Modal change logic (MCL): Specifying the reasoning of knowledge-based systems. *Data Knowl. Eng.*, 25(1-2):243–269, 1998.

[7] Christian G. Fermüller and Herbert Langsteiner. Tableaux for finite-valued logics with arbitrary distribution modalities. In Harrie de Swart, editor, *Proc. Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'98)*, volume 1397 of *Lecture Notes in Computer Science*, pages 156–171, Oisterwijk (Netherlands), 1998. Springer-Verlag.

[8] Melvin C. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* D. Reidel Publishing Co., Dordrecht, 1983.

[9] Melvin C. Fitting. Modality and databases. In Roy Dyckhoff, editor, *Proc. Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2000)*, Lecture Notes in Computer Science, St. Andrews, Scotland, 2000. Springer-Verlag. to appear.

[10] Nir Friedman and Joseph Y. Halpern. Modeling belief in dynamic systems. II: Revision and update. *J. Artif. Intell. Res. (JAIR)*, 10:17–167, 1999.

[11] Matthew L. Ginsberg. Multi-valued logics. *Computational Intelligence*, 4(3), 1988.

[12] Rajeev Goreé. Tableau methods for modal and temporal logics. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*, pages 297–396. Kluwer Academic Publishers, 1999.

[13] Reiner Hähnle. *Automated Deduction in Multiple-valued Logics.* Clarendon Press, Oxford, 1993.

[14] Mark Ryan and Pierre-Yves Schobbens. Counterfactuals and updates as inverse modalities. *J. of Logic Lang. Inf.*, 6(2):123–146, 1997.

[15] Gernot Salzer. Optimal axiomatizations for multiple-valued operators and quantifiers based on semilattices. In Michael McRobbie and John Slaney, editors, *Proc. 13th Conference on Automated Deduction, New Brunswick/NJ, USA*, volume 1104 of *Lecture Notes in Computer Science*, pages 688–702. Springer-Verlag, 1996.

Infinite SLaKE-Tableaux

MARCELO FINGER, *Department of Computer Science, University of São Paulo, Rua do Matão 1010, 05508-900, São Paulo, Brazil.*
E-mail: mfinger@ime.usp.br

Abstract

In this work, we explore an algorithm for the Correspondence Theory of Substructural Categorial Logic, known as a SLaKE-tableau. Such algorithm allows us to compute a first-order formula equivalent to a sequent in substructural categorial logic when such sequent generates a finite SLaKE-tableau.

Here we study the cases when such SLaKE-tableau is infinite. We show that it is decidable whether a sequent will generate a finite tableau, presenting a graph-based algorithm for its detection. Our main result is a proof that sequents generating infinite SLaKE-tableaux are *not* equivalent to a first-order formula.

Keywords: Correspondence Theory, KE-Tableaux, SLaKE-Tableaux, Substructural Logics.

1 Introduction

Since Kripke has proposed a possible world semantics for modal logics [5], it has been noted that the presence of certain modal axioms impose specific restrictions on the binary accessibility relation of Kripke frames. Any modal axiom can be translated to a second-order formula and at the heart of *modal correspondence theory* lies the identification of which axioms correspond to some first-order restriction over Kripke frames [10].

In her PhD thesis [6], Natasha Kurtonina has shown that a fragment of substructural logics, known as Categorial Logics [7, 1], can be treated as a modal logic with a Kripke-style semantics based on a *ternary accessibility relation*. Kurtonina showed, using several methods, that some sequents are equivalent to first-order restrictions over ternary frames, while others have no such first-order equivalence.

In a recent work [4], we have proposed an algorithmic method for Categorial Correspondence Theory. This method is based on *SLaKE-tableaux*, and we have shown that a categorial sequent that generates a finite SLaKE-tableau is equivalent to a first-order formula that is obtainable by algorithmic methods.

In this work we extend such result, showing that:

- there is a graph-based algorithm that decides whether a categorial sequent generates an infinite SLaKE-tableau; and
- if a categorial sequent generates an infinite SLaKE-tableau, it is not equivalent to a first-order formula.

Our work is inserted in the tradition of algorithmic correspondence theory for Modal Logics. The most traditional result in this area is the *Sahlqvist-van Benthem Algorithm* that finds a first-order formula corresponding to monomodal formula in the so-called *Sahlqvist* fragment of modal logic [8, 10]. That algorithm works only for that restricted modal fragment, for the problem of finding whether a modal formula has a first-order equivalent is undecidable, a result due to Chagrova. We do not suffer from this restriction because our fragment involves ternary relations (and not binary, as in the monomodal case); also, the semantics of substructural logics further restricts the format of the second-order equivalents of substructural logics. As a result the problem of finding a first-order equivalent is decidable for the fragment dealt with in this paper.

The paper develops as follows. Section 2 presents Substructural Categorial Logics and its semantics based on a ternary accessibility relation. Section 3 introduces SLaKE-tableaux. We concentrate on showing the decidability of finiteness for SLaKE-tableaux in Section 4. Finally, Section 5 shows that sequents associated with infinite tableaux have no first-order equivalent.

2 Substructural Categorial Logics

In this work we deal with the fragment of substructural categorial logic containing a countable set of propositional letters $\mathcal{P} = \{p_0, p_1, \dots\}$ and the usual categorial binary connectives: / (slash), \ (backslash) and $\bullet$ (product). The $\bullet$ connective is assumed to be left-associative, that is, $A \bullet B \bullet C \equiv (A \bullet B) \bullet C$. The /-connective is the right implication connective, such that from A/B and B (in this order), we infer A; similarly, the \-connective is the left implication, so from B and $B\backslash A$ (in this order) infer A.

A sequent is an expression of the form $\Gamma \vdash C$, where Γ is the antecedent and C is the single formula that is contained in the consequent. The antecedents of sequents is a comma-separated binary trees, also left associative, such that an antecedent of the form $A_1, A_2, A_3 \equiv ((A_1, A_2), A_3)$. A sequent calculus presentation for such a logic, known as the *non-associative Lambek Calculus*, has been defined in several places in the literature (e.g. in [7, 1]). But here, we concentrate purely on a modal-like semantic presentation based on *ternary frames*.

We define a *ternary frame* as a pair $\mathrm{F} = (W, R)$, where W is a non-empty set of *possible worlds* and R is a ternary *accessibility relation*. A *ternary model* $\mathcal{M} = (W, R, V)$ consists of ternary frame plus a *valuation* $V : \mathcal{P} \to 2^W$, sending propositional letters to sets of worlds. The semantic interpretation of categorial formulas in the $\{/, \backslash, \bullet\}$-fragment over ternary frames is given below

$$\begin{array}{lll}
\mathrm{F}, V, a \models p & \text{iff} & a \in V(p) \\
\mathrm{F}, V, a \models A/B & \text{iff} & \forall b \forall c (Rcab \wedge \mathrm{F}, V, b \models B \Rightarrow \mathrm{F}, V, c \models A) \\
\mathrm{F}, V, a \models B\backslash A & \text{iff} & \forall b \forall c (Rcba \wedge \mathrm{F}, V, b \models B \Rightarrow \mathrm{F}, V, c \models A) \\
\mathrm{F}, V, a \models B \bullet A & \text{iff} & \exists b \exists c (Rabc \wedge \mathrm{F}, V, b \models A \wedge \mathrm{F}, V, c \models B)
\end{array}$$

We also use the abbreviations:

$$\begin{array}{l}
Rab(cd) =_{def} \exists x (Rabx \wedge Rxcd) \\
Ra(bc)d =_{def} \exists y (Rayd \wedge Rybc)
\end{array}$$

Such abbreviation associates to the left; that is, $Ra(bcd)e =_{def} Ra((bc)d)e$, etc. As usual, we write $\mathrm{F}, a \models A$ when $\mathrm{F}, V, a \models A$ for any valuation V; if a is omitted, this means that the condition holds for any possible world.

A sequent $A_1, \ldots A_n \vdash C$ is valid at a world $a \in W$ in ternary frame F (notation: $\mathrm{F}, a \models (A_1, \ldots A_n \vdash C)$) iff $\mathrm{F}, V, a \models A_1 \bullet \ldots \bullet A_n$ implies $\mathrm{F}, V, a \models C$. This is equivalent to saying that if for some $a_1, \ldots, a_n$, $Ra(a_1 \ldots a_{n-1})a_n$ and, for $1 \leq i \leq n$, $\mathrm{F}, V, a_i \models A_i$, this implies $\mathrm{F}, V, a \models C$.

Note that F can be seen as a first-order model for a language over R and $P_1, \ldots, P_n$, where each P_i is a predicate symbol corresponding to the propositional letter p_i. The notation $(\forall P)\varphi$ indicate the universal closure of all second-order variables in φ, and $(\exists P)\varphi$ the existential one. Following the modal logic tradition, the *standard translation* of a categorial formula A into second-order logic formula is $(\forall P)\forall a ST_a(A)$, where:

$$\begin{array}{lcl}
ST_a(p_i) & = & P_i(a) \\
ST_a(A \bullet B) & = & \exists bc(Rabc \wedge ST_b(A) \wedge ST_c(B)) \\
ST_a(B/A) & = & \forall bc(Rcab \wedge ST_b(A) \Rightarrow ST_c(B)) \\
ST_a(A\backslash B) & = & \forall bc(Rcba \wedge ST_b(A) \Rightarrow ST_c(B))
\end{array}$$

The second-order quantification is over the relevant predicate symbols and reflects all relevant valuation in a frame. So every categorial sequent $A \vdash C$ corresponds to a second-order formula such that $\mathrm{F} \models (A_1, \ldots, A_n \vdash C)$ iff

$$\mathrm{F} \models (\forall P)\forall a_1 \ldots a_n a \ (ST_{a_1}(A_1) \wedge \ldots \wedge ST_{a_n}(A_n) \wedge R(a_1 \ldots a_n)a \Rightarrow ST_a(C))$$

The goal of correspondence theory is to know when such a second-order formula defines a *first-order* frame property, that is, if there is a first-order formula ψ such that

$$\mathrm{F} \models (A_1, \ldots, A_n \vdash C) \textit{ iff } \mathrm{F} \models \psi$$

It is the computation of such a property, when it exists, that we investigate next by means of SLaKE-tableaux.

3 Substructural Labelled KE Tableaux

Substructural Labelled KE (SLaKE) tableaux are the main proof theoretical equipment we use in the generation of correspondence formulas for substructural sequents.

D'Agostino has shown in [3] that analytic tableaux, in the style proposed by Smullyan [9], cannot polynomially simulate truth tables and in some cases perform much worse than them. To avoid such problems in a principled way, KE-tableaux were introduced.

The use of KE tableaux for substructural logics have been proposed in [2], by attaching a label to the signed formula, as in $T\ A : a$. We use this idea, but without following its labelling discipline. Instead, we simply add a new label at each new node of the tableau. Formally, let $\mathcal{L}$ be a countable set of labels, let A be a categorial formula; then for every $a \in \mathcal{L}$, the formulas $T\ A : a$ and $F\ A : a$ are *signed labelled formulas.*

Each finite SLaKE-tableau is associated with a *correspondence formula.* The original sequent is associated with an initial tableau and with a second-order formula. There are several *linear expansion rules* and a single *branching rule* based on the Principle of Bivalence. Each of the tableau expansion rules is associated with an expansion of the correspondence formula of the form $\sharp_i := \varphi(R, B_1, \ldots, B_n, \sharp_{i+1})$, where R is the ternary accessibility relation, $B_1, \ldots, B_n$ are the formulas generated in the expansion, and $\sharp$ is the "substitution place" for next expansion and can be read simply as *truth*. The tableau rules for SLaKE-tableaux are illustrated in Figure 1.

Name	*Type*	*SLaKE Expansion*	*Formula Expansion*
$(T/)$	(linear-$\forall$)	$\dfrac{T\ A/B : a \qquad T\ B : b}{T\ A : c \text{ (new } c)}$	$\sharp_i := \forall c(Rcab \Rightarrow (ST_c(A) \wedge \sharp_{i+1}))$
$(F/)$	(linear-$\exists$)	$\dfrac{F\ A/B : a}{T\ B : b \text{ (new } b) \qquad F\ A : c \text{ (new } c)}$	$\sharp_i := \exists b \exists c(Rcab \wedge ST_b(B) \wedge \neg ST_c(A) \wedge \sharp_{i+1})$
$(T\backslash)$	(linear-$\forall$)	$\dfrac{T\ B\backslash A : a \qquad T\ B : b}{T\ A : c \text{ (new } c)}$	$\sharp_i := \forall c(Rcba \Rightarrow (ST_c(A) \wedge \sharp_{i+1}))$
$(F\backslash)$	(linear-$\exists$)	$\dfrac{F\ B\backslash A : a}{T\ B : b \text{ (new } b) \qquad F\ A : c \text{ (new } c)}$	$\sharp_i := \exists b \exists c(Rcba \wedge ST_b(B) \wedge \neg ST_c(A) \wedge \sharp_{i+1})$
$(T\bullet)$	(linear-$\exists$)	$\dfrac{T\ A \bullet B : a}{T\ A : b \text{ (new } b) \qquad T\ B : c \text{ (new } c)}$	$\sharp_i := \exists b \exists c(Rabc \wedge ST_b(A) \wedge ST_c(B) \wedge \sharp_{i+1})$
$(F\bullet)$	(linear-$\forall$)	$\dfrac{F\ A \bullet B : a \qquad T\ A : b}{F\ B : c \text{ (new } c)}$	$\sharp_i := \forall c(Rabc \Rightarrow (\neg ST_c(B) \wedge \sharp_{i+1}))$
(PB)	(branch)	$\dfrac{}{T\ A : x \qquad F\ A : x}$	$\sharp_i := \forall x((ST_x(A) \wedge \sharp^1) \vee (\neg ST_x(A) \wedge \sharp^2))$

FIG. 1. SLaKE rules

In each linear rule in Figure 1, the formulas above the horizontal line are the *premises* of the rule, and those below it are the *conclusions* of the rule. There are one-premised and two-premised rules, but each rule has exactly one premise that is a *compound* formula, which is called the *main premise*; other premises are called *auxiliary*. Two-premised rules are *universal* and one-premised rules are *existential*. If either of the conclusions of an existential rule is present on the current branch, it is not added again with a new label. Universal rules always generate a new conclusion.

The last rule in Figure 1 is the Principle of Bivalence (PB) branching rule. It is only applied for a formula A following the *branching heuristics*:

> PB is used for a formula A that serves as an auxiliary premise for a $\forall$-rule;
> PB is only applied if there is a unused $\forall$-main premise but no other linear expansion is possible.

The main premises that trigger the application of PB for A are: $F\ A \bullet B$, $T\ A\backslash B$ and $T\ B/A$. The corresponding $\forall$-rule will be applicable on the $T\ A$ branch. Only subformulas of the original sequent will be introduced by PB. It introduces two "substitution places" in the correspondence formula, $\sharp^1$ and $\sharp^2$, one for each new branch. Each active branch in a SLaKE tableau always has exactly one substitution place. Substitution places guarantee that each formula introduced in the correspondence formula will be in the scope the correct quantifiers.

A sequent of the form $A_1, \ldots, A_n \vdash C$ is transformed into the initial SLaKE-tableau:

$$\begin{array}{c} T\ A_1 : a_1 \\ \vdots \\ T\ A_n : a_n \\ F\ C : a \end{array}$$

Since the tableau is a refutation method, this induces the correspondence formula:

$$\neg\varphi = \neg\exists a a_1 \ldots a_n [ST_{a_1}(A_1) \wedge \ldots \wedge ST_{a_n}(A_n) \wedge \neg ST_a(C) \wedge Ra(a_1 \ldots a_{n-1})a_n \wedge \sharp]$$

A single premised sequent $A \vdash C$ generates the initial tableau containing $T\ A : a$ and $F\ C : a$, with the initial correspondence formula $\neg\varphi(\sharp_1) = \neg\exists a(ST_a(A) \wedge \neg ST_a(C) \wedge \sharp_1)$. We could extend the method for sequents with empty antecedents, but we do not pursue this topic here.

The aim of the SLaKE-tableau construction is *not* to close every tableau branch, but to expand each tableau branch until no more expansions are possible. Each expansion step will also give us a new version of the correspondence formula. If we can finitely expand all tableau branches, a valuation for the atomic formula is constructed, so that we obtain a first-order formula by substituting in the final formula the evaluated values. It is also possible that there will be some infinite branches (something that would not happen in simple propositional tableaux).

Formally, let the countable set of labels $\mathcal{L}$ be linearly ordered by $<_{\mathcal{L}}$. A *SLaKE-saturated set* Γ is a set of labelled signed formulas such that, with respect to the rules of Figure 1:

(a) If the premise of a $\exists$-rule is in Γ, each of its consequence is in Γ for some label.

(b) If the premises of a $\forall$-rule are in Γ, its consequence is in Γ with a label *bigger than those of its premises*, according to $<_{\mathcal{L}}$.

(c) For each compound formula in Γ that is a main premise of a $\forall$-rule, there must be in Γ either a an auxiliary premise or the opposite of it.

The expansion of a tableau aims at constructing branches that are SLaKE-saturated sets. Items (a) and (b) correspond to normal branch expansion. The restriction on the order of the consequence labels in (b) rises the possibility of having infinite SLaKE-saturated sets. Item (c) is the branching heuristics.

The expansion of a tableau is thus the stepwise construction of a *saturated counter-model* for the input sequent. Each step generates a second-order formula $\neg\varphi_{i+1}(\sharp_{i+1})$ that is equivalent to $\neg\varphi_i(\sharp_i)$. At the end of a finite expansion, a correspondence formula will be built from a suitable valuation, called the *canonical valuation*.

DEFINITION 3.1
Let $X \in \{T, F\}$; define $\bar{X}$ such that $\bar{T} = F$ and $\bar{F} = T$. For each SLaKE atomic formula $X\ p : x$ in the tableau, define:

$$O(X\ p : x) = \{y \mid \bar{X}\ p : y \text{ precedes X p:x}\}$$

For every atomic p, define the *canonical valuation* of $P(x)$ implicitly as:

$$\bigwedge_{Xp:x} ((P(x) \leftrightarrow \bigwedge_{y \in O(Tp:x)} x \neq y) \wedge (\neg P(x) \leftrightarrow \bigwedge_{y \in O(Fp:x)} x \neq y))$$

The canonical valuation is the *minimal condition* a countermodel must satisfy to falsify the input sequent:

- all the atomic conditions determined by the tableau must hold (see Lemma 5.1); and
- no formula can be both true and false at the same world.

To obtain a first-order formula we need to substitute the *canonical* valuation into $\neg\varphi$. If the tableau saturates, such a valuation/substitution satisfies all compound formulas. The first-order formula $\neg\psi$ thus obtained is obviously implied by $\neg\varphi$. We know that

FACT 3.2 ([4])
If a sequent produces a finite SLaKE-tableau, then it is equivalent to a first-order restriction over ternary frames. [1]

We want to show the converse of such statement, namely that infinite tableaux have no first-order equivalent. Let us first see some examples of SLaKE tableaux.

3.1 Finite SLaKE-Tableau

Consider the sequent $A/B \vdash B\backslash A$. Its associated SLaKE tableau is:

1.	$T\ p/q : a$		
2.	$F\ q\backslash p : a$		$\neg\varphi = \neg\exists a(ST_a(p/q) \wedge \neg ST_a(q\backslash p) \wedge \sharp_1)$
3.	$T\ q : b$	from 2	
4.	$F\ p : c$	from 2	$\sharp_1 := \exists b \exists c(Rcba \wedge Q(b) \wedge \neg P(c) \wedge \sharp_2)$
5.	$T\ p : d$	from 1,4	$\sharp_2 := \forall d(Rdab \Rightarrow P(d) \wedge \sharp_3)$

Initially, we expand line 2, and simultaneously, using the semantics of $q\backslash p$, we expand the correspondence formula substituting $\sharp_1$ into it. We then use lines 1 and 3 for another expansion, obtaining $\sharp_2$. At this point, the tableau is saturated and closed (remember our main goal is not to close a tableau, but to saturate every branch of it). Make $\sharp_3 := \top$. We have thus built the second-order formula:

$$\neg\varphi = \neg\exists a(ST_a(p/q) \wedge \neg ST_a(q\backslash p) \wedge \exists b \exists c(Rcba \wedge Q(b) \wedge \neg P(c) \wedge \forall d(Rdab \Rightarrow P(d))))$$

[1] This result has not been published yet, the the results in this paper are not dependent on it but are intended to complement it.

The formula $\neg\varphi$ is equivalent to the original sequent. We construct the following implicit canonical valuation for the atoms in $\neg\varphi$, by forcing that no world both contain a T- and an F-signed atom:

$$Q(b) \leftrightarrow \top \qquad\qquad (\neg P(c) \leftrightarrow \top) \wedge (P(d) \leftrightarrow d \neq c)$$

Saturation guarantees that the compound formulas in $\neg\varphi$ can be substituted by $\top$. By substituting each second-order variable in $\neg\varphi$ by their canonical valuation, we obtain the first-order formula

$$\neg\exists a\exists b\exists c(\top \wedge \top \wedge Rcba \wedge \exists b\exists c(Rcba \wedge \top \wedge \top \wedge \forall d(Rdab \Rightarrow d \neq c)))$$

which is equivalent to $\forall a\forall b\forall c(Rcba \Rightarrow Rcab)$, meaning that R must have the *commutativity* property for its second and third positions. It is easy to verify that any model satisfying such a property also satisfy the original sequent, and therefore it is *equivalent* to it.

Now let us see an example with branching. For that, consider Peirce's Axiom $(p\backslash q)\backslash p \vdash p$.

1.	$T\ (p\backslash q)\backslash p : a$		
2.	$F\ p : a$		
3(*i*).	$T\ p\backslash q : b$	3(*ii*).	$F\ p\backslash q : b$
4(*i*).	$T\ p : c$	4(*ii*).	$T\ p : e$
5(*i*).	$T\ q : d$	5(*ii*).	$F\ q : f$

$$\neg\varphi = \neg\exists a(ST_a((p\backslash q)\backslash p) \wedge \neg P(a) \wedge \sharp_1)$$
$$\sharp_1 := \forall b((ST_b(p\backslash q) \wedge \sharp_2^i) \vee (\neg ST_b(p\backslash q) \wedge \sharp_2^{ii}))$$
$$\sharp_2^i = \forall c(Rcba \Rightarrow (P(c) \wedge \sharp_3^i))$$
$$\sharp_3^i = \forall d(Rdcb \Rightarrow (Q(d) \wedge \sharp_4^i)) \qquad \sharp_2^{ii} = \exists e\exists f(Rfeb \wedge P(e) \wedge \neg Q(f) \wedge \sharp_3^{ii})$$

The branches are independently developed. The left branch will cause substitutions in $\sharp_2^i$ and the right branch will cause substitutions in $\sharp_2^{ii}$. If we were looking for a closed tableau, we could stop the development of the left branch at $4(i)$; however, our aim here is to obtain a *SLaKE-saturated* set at each branch, so we proceed to obtain $5(i)$. The correspondence formula obtained after both branches are saturated is:

$$\begin{array}{ll} \forall a\exists b[& \neg(ST_a((p\backslash q)\backslash p) \wedge \neg P(a) \wedge ST_b(p\backslash q) \wedge \forall c(Rcba \Rightarrow (P(c) \wedge \forall d(Rdcb \Rightarrow Q(d)))))] \\ & \wedge \\ \forall a\exists b[& \neg(ST_a((p\backslash q)\backslash p) \wedge \neg P(a) \wedge \neg ST_b(p\backslash q) \wedge \exists e\exists f(Rfeb \wedge P(e) \wedge \neg Q(f)))] \end{array}$$

The canonical valuation is developed such that no atom is both false and true at a world; to guarantee the right scoping of variables, a world is forced to be different from those opposite ones preceding it in the tableau. This results in the valuation:

$$(\neg P(a) \leftrightarrow \top) \wedge (P(c) \leftrightarrow c \neq a) \wedge (P(e) \leftrightarrow e \neq a)$$
$$(Q(d) \leftrightarrow \top) \wedge (\neg Q(f) \leftrightarrow \top)$$

The first-order correspondence formula thus obtained is:

$$\forall a\exists b(Raba \wedge \forall ef(Rfeb \Rightarrow e = a))$$

3.2 Infinite SLaKE-Tableau

It is not always the case that a tableau branch can be finitely saturated. In those cases we cannot apply the method above, so we do not get a first-order formula. For example, consider the tableau for $p/p \vdash p\backslash p$:

1.	$T\ p/p : a$	
2.	$F\ p\backslash p : a$	
3.	$T\ p : b$	from 2
4.	$F\ p : c$	from 2
5.	$T\ p : d$	from 1 and 3
6.	$T\ p : e$	from 1 and 5
7.	$T\ p : f$	from 1 and 6
	$\vdots$	

It is clear that the aim of constructing a SLaKE-saturated leads in this case to an infinite tableau (with a single branch). Another example is $(A\backslash A)\backslash A \vdash A$, which contains one finite and one infinite branch:

1. $T\ (p\backslash p)\backslash p : a$
2. $F\ p : a$

$3i.$	$T\ p\backslash p : b$	$3ii.$	$F\ p\backslash p : b$
$4i.$	$T\ p : c$	$4ii.$	$T\ p : e$
$5i.$	$T\ p : d$ from $3i$ and $4i$	$5i.$	$F\ p : f$
$6i.$	$T\ p : e$ from $3i$ and $5i$		
	$\vdots$		

It was shown in [6] that those sequents have no first-order correspondence.

4 Detecting Infinite Branches

We can detect infinite branches of SLaKE-tableaux in the categorial fragment in the following way. For each branch $\mathcal{B}$ in a tableau, define its *dependency graph* $\mathcal{G}_{\mathcal{B}} = (N, E)$, where a node in N is a set of signed unlabelled formulas used as joint premises in an expansion step in $\mathcal{B}$, that is:

$$N = \{\{X_1A_1, \ldots, X_nA_n\} |\ \ X_i \in \{T, F\} \text{ and } X_1\,A_1 : a_1, \ldots, X_n\,A_n : a_n \in \mathcal{B} \text{ are used as joint premises of the same derivation step in } \mathcal{B}\text{'s expansion } \}$$

According to the expansion rules in Figure 1, the nodes will be singletons or have two elements. There is a directed edge from $n_1 = \{X_1A_1, \ldots, X_nA_n\}$ to $n_2 \ni X\,A$ iff in $\mathcal{B}$'s construction there was an expansion rule with premises $X_1\,A_1 : a_1, \ldots, X_n\,A_n : a_n$ and consequence $X\,A : a$.

THEOREM 4.1
A branch $\mathcal{B}$ in a SLaKE tableau over the categorial fragment is infinite iff $\mathcal{G}_{\mathcal{B}}$ has a cycle.

A tableau is *non-deterministic* if at some point in its expansion two or more rules were applicable.

COROLLARY 4.2
If one expansion of a non-deterministic branch is infinite, then any other possible expansion will also be infinite.

COROLLARY 4.3
It is decidable whether the expansion of a finite set of SLaKE-formulas will lead to an infinite branch.

5 The Second-Orderness of Infinite SLaKE-Tableaux

The aim of this section is to show that if a SLaKE-tableau has an infinite branch, then its corresponding second-order formula $\neg\varphi$ cannot be equivalent to a first-order formula. For that we use an auxiliary construction.

Parallel to the construction of a SLaKE-saturated set, we build a R-configuration ρ. The set ρ is initially empty and is built during the tableau's expansion:

(a) If a $\exists$-rule generates an expansion of the form $\exists xy(R(x,y,z) \wedge \ldots)$, add $R(x,y,z)$ to ρ.

(b) If a $\forall$-rule generates an expansion of the form $\forall xy(R(x,y,z) \wedge \ldots \Rightarrow \ldots)$, add $R(x,y,z)$ to ρ.

The configuration ρ allows us to construct a model for the SLaKE-saturated set, which will be a countermodel to the input sequent.

Suppose we have a saturated branch $\mathcal{B}$, that may be finite or infinite, with associated SLaKE-saturated set Γ and R-configuration ρ. A *$\mathcal{B}$-model* is any model $\mathcal{M}_\mathcal{B} = (W, R, V_\mathcal{B})$ such that:

$$\begin{array}{l} W \supseteq \{x \mid x \text{ is a lable ocurring in } \Gamma\} \\ R \supseteq \{\langle x,y,z\rangle \mid Rxyz \in \rho\} \\ \text{If } F\ p : x \in \Gamma \text{ then } x \notin V_\mathcal{B}(p) \\ \text{If } T\ p : x \in \Gamma \text{ then } x \in V_\mathcal{B}(p) \end{array}$$

Let $X\ A : a \in \Gamma$, $X \in \{T, F\}$. We say that $\mathcal{M}_\mathcal{B}$ *satisfies* $X\ A : a$ if $\mathcal{M}_\mathcal{B}, a \models A$ iff $X = T$.

LEMMA 5.1
Let Γ be a SLaKE-saturated set corresponding to a branch $\mathcal{B}$, and let $\mathcal{M}_\mathcal{B}$ be a $\mathcal{B}$-model. For every $X\ A : x \in \Gamma$ then $\mathcal{M}_\mathcal{B}$ satisfies $X\ A : x$.

LEMMA 5.2
Consider a sequent $A_1, \ldots, A_n \vdash C$ whose tableau contains a saturated branch $\mathcal{B}$. Let $\neg\varphi$ be the sequent's corresponding second-order formula. Then, for any $\mathcal{B}$-model, $\mathcal{M}_\mathcal{B}$, $\mathcal{M}_\mathcal{B} \not\models \neg\varphi$.

Now let us consider a sequent $A_1, \ldots, A_n \vdash C$ whose saturated SLaKE-tableau contains an infinite branch $\mathcal{B}$ and whose second-order corresponding formula is $\neg\varphi$.

For each step i in the construction of the infinite branch $\mathcal{B}$, let $\mathcal{B}_i$ be the set of formulas contained in the tableau at step i. We define a theory τ_i inductively as:

$$\begin{array}{ll} W_i = \{x \mid x \text{ is a lable ocurring in } \mathcal{B}_i\} & \text{the world at } i \\ \alpha_i = \bigwedge\{x \neq y \mid x, y \in W_i; x \neq y\} & \text{unique names for worlds} \\ \rho_i = \bigwedge\{Rxyz \in \rho \mid x, y, x \in W_i\} & \rho \text{ restricted to } W_i \\ V_i = \bigwedge\{P(x) \mid T\ p : x \in \mathcal{B}_i\} \wedge \bigwedge\{\neg Q(x) \mid F\ q : x \in \mathcal{B}_i\} & V \text{ restricted to } W_i \\ \tau_i = \neg\varphi \wedge V_i \wedge \alpha_i \wedge \rho_i & \end{array}$$

LEMMA 5.3
At every step i there is a model for τ_i.

THEOREM 5.4
If a categorial sequent generates a SLaKE-tableau with an infinite branch, than it is not equivalent to a first-order formula.

PROOF. Assume that the tableau corresponding formula $\neg\varphi$ is equivalent to a first-order formula. By Corollary 4.2 any expansion of the input sequent will generate an infinite branch. For any such infinite branch $\mathcal{B}$, generate the infinite sequence $\{\tau_i\}_{i\in\mathbb{N}}$, where each τ_i is equivalent to a first-order formula. By Lemma 5.3, any finite subset of $\{\tau_i\}_{i\in\mathbb{N}}$ has a model (just take the model for the largest i). However, any model of $\{\tau_i\}_{i\in\mathbb{N}}$ is a $\mathcal{B}$-model and by Lemma 5.2 it cannot satisfy $\neg\varphi$, so $\{\tau_i\}_{i\in\mathbb{N}}$ has no models.

This contradicts the compactness theorem for first-order logic, so $\neg\varphi$ cannot be equivalent to a first-order formula. ∎

Acknowledgements

The author was partly supported by the Brazilian Research Council (CNPq), grant PQ 300597/95-5.

References

[1] Bob Carpenter. *Type-Logical Semantics.* MIT Press, 1997.

[2] M. D'Agostino and D. Gabbay. A Generalization of Analytic Deduction via Labelled Tableaux, part I: Basic Substructural Logics. *Journal of Automated Reasoning*, 13:243–281, 1994.

[3] Marcello D'Agostino. Are tableaux an improvement on truth-tables? — cut-free proofs and bivalence. *Journal of Logic, Language and Information*, 1:235–252, 1992.

[4] M. Finger. Algorithmic correspondence theory for substructural categorial logic. Submitted for publication, 2000.

[5] Saul Kripke. Semantical analysis of modal logic I. *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik.*, 9(67–96), 1963.

[6] Natasha Kurtonina. *Frames and Labels — A Modal Analysis of Categorial Inference.* PhD thesis, Research Institute for Language and Speech (OTS), Utrecht, and Institute of Logic, Language and Information (ILLC), Amsterdam, 1994.

[7] M. Moortgat. Categorial type logics. In J. Van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 93–178. Elsevier North-Holland/The MIT Press, 1997.

[8] H. Sahlqvist. Completeness and correspondence in the first- and second-order semantics for modal logic. In *Proceedings of the 3rd Scandinavian Logic Symposium*, 1975.

[9] Raymond M. Smullyan. *First-Order Logic.* Springer-Verlag, 1968.

[10] J. van Benthem. Correspondence theory. In *Handbook of Philosophical Logic*, volume II, pages 167–248. D. Reidel Publishing Company, 1984.

Declarative Semantics for Quantified Disjunctive Programs

DUŠAN GULLER, *Institute of Informatics, Comenius University, Mlynská dolina, 842 15 Bratislava, Slovakia.*
E-mail: guller@fmph.uniba.sk

Abstract

In the paper, we present model, fixpoint, and Herbrand semantics for quantified disjunctive programs with equality, sets of formulae in prenex conjunctive normal form. We shall proceed from the declarative semantics concerning only logical consequences of a given program P: $\mathcal{DS}(P) = \{C \mid C$ *is a formula in prenex conjunctive normal form,* $P \models C\}$. The model semantics is proposed by means of disjunctive models, a slight modification of clausal models from [5, 6]. We briefly introduce a class of disjunctive models and outline the relation between disjunctive and first-order models. We then recall the notion of a characteristic model for a theory, established in [6]. Such a model satisfies exactly the formulae being logical consequences of the given theory. The following characterisation theorem holds: *If a quantified disjunctive program P has a disjunctive model, then P has a characteristic disjunctive model* M*, which is constructible by a closure operator* $\mathcal{C}_P$. This model M will be taken as the basis for the model semantics of the program P: $\mathcal{MS}(P) = \{C \mid C$ *is a formula in prenex conjunctive normal form,* $\mathrm{M} \models^{\vee} C\}$. By the least fixpoint of the operator $\mathcal{C}_P$, we define the fixpoint semantics for P. In conclusion, we shall deal with a generalised statement of Herbrand's theorem, which applies to quantified disjunctive programs without adding Skolem and Herbrand function symbols [7]. This extended Herbrand's theorem gives us some characterisation of logical consequences of a quantified disjunctive program in terms of tautological following. By this statement, we establish the Herbrand semantics. As an immediate consequence of the characterisation theorem, we obtain the coincidence of the proposed semantics.

Keywords: logic programming, knowledge-based systems, model theory

1 Introduction

In disjunctive logic programming, the use of 'complete' Herbrand models leads to the minimal model semantics which yields an uncountable set of minimal models yet for trivial disjunctive programs, e.g. $P = \{p_0(f(x)) \vee p_1(f(x))\}$ (see the motivation example in [6]). We shall aim to overcome this difficulty and propose a model semantics for disjunctive programs on the basis of one model, instead of the class of minimal models. Simultaneously, we desire to preserve the equivalence to the above mentioned declarative semantics, based on logical following. This will be necessary for the first step of developing semantics for quantified disjunctive programs; namely, it provides us a clear logical basis for adding various types of negation-by-default. The demand of a one-model semantics implies an extension of the notion of a 'classical' and 'complete' Herbrand model to some kind of an 'incomplete' model. Take the simple program $P = \{p \vee \neg q\}$. We see that neither $P \not\models p$ nor $P \not\models \neg p$ nor $P \not\models \neg q$, $P \not\models q$, but $P \models p \vee \neg q$. In accordance with the intended equivalence to the declarative semantics, we expect that our proposed model, say M, will satisfy: $\mathrm{M} \not\models p$, $\mathrm{M} \not\models \neg p$, $\mathrm{M} \not\models \neg q$, $\mathrm{M} \not\models q$, but $\mathrm{M} \models p \vee \neg q$ - that is, it will be 'incomplete' in contrast to any 'complete'

Herbrand model, where e.g. either p or $\neg p$ must be true.

We now informally explain one way how to generalise 'complete' first-order models and develop a model semantics for quantified disjunctive programs viewed as sets of formulae in prenex conjunctive normal form. We introduce the notion of a disjunctive model:

Definition 1.1
A disjunctive model M for $\mathcal{L}$ is a pair $(\mathcal{U}_{\mathrm{M}}, \mathcal{B}_{\mathrm{M}})$ defined as follows:

- $\mathcal{U}_{\mathrm{M}} \neq \emptyset$ is the universum, domain of the model M. The universum may be an arbitrary non-empty set, in contrast to clausal models, where only a set of ground terms is considered.
- A function symbol $f \in \mathit{Func}_{\mathcal{L}}$ is interpreted as a function $f^{\mathrm{M}} : \mathcal{U}_{\mathrm{M}}^{ar(f)} \to \mathcal{U}_{\mathrm{M}}$.
- The base $\mathcal{B}_{\mathrm{M}}$ is a set of finite sets of triples of the form $(s, p, (u_1, \ldots, u_{ar(p)}))$ where $s \in \{0, 1\}$, $p \in \mathit{Pred}_{\mathcal{L}}$ and $u_i \in \mathcal{U}_{\mathrm{M}}$. Such a triple will be denoted as $\overline{p(u_1, \ldots, u_{ar(p)})}$ when $s = 0$, or $\overline{\neg p(u_1, \ldots, u_{ar(p)})}$ when $s = 1$. It is conceived as a 'semantical counterpart' of the ground literal $p(u_1, \ldots, u_{ar(p)})$ of the language $\mathcal{L} \cup \mathcal{U}_{\mathrm{M}}$ (new added constants). A finite set of triples will be denoted as $\overline{l_1} \overline{\vee} \cdots \overline{\vee} \overline{l_n}$ where $\overline{l_i}$'s are triples - 'counterparts' of ground literals l_i's. It is conceived as a 'semantical counterpart' of the ground literal disjunction $l_1 \vee \cdots \vee l_n$. Similarly, a finite union $\bigcup_{1 \le i \le n} \overline{D_i}$ where $\overline{D_i}$'s are finite sets of triples will be denoted as $\overline{\bigvee_{1 \le i \le n}} \overline{D_i}$.

  The base $\mathcal{B}_{\mathrm{M}}$ satisfies the following conditions:
  - $\nexists b \in \mathcal{B}_{\mathrm{M}}$ containing a pair of contradictory triples - i.e. triples of the form $\overline{l}$ and $\overline{\neg l}$.
  - $\emptyset \notin \mathcal{B}_{\mathrm{M}}$.
  - $\neg(\exists b \, \exists b' \in \mathcal{B}_{\mathrm{M}} \, b \subset b')$.
  - If $\overline{l} \in b$, $\overline{\neg l} \in b'$, and $b, b' \in \mathcal{B}_{\mathrm{M}}$, then either $\exists b'' \in \mathcal{B}_{\mathrm{M}} \, b'' \subseteq (b - \{\overline{l}\}) \cup (b' - \{\overline{\neg l}\})$ or $(b - \{\overline{l}\}) \cup (b' - \{\overline{\neg l}\})$ contains a pair of contradictory triples.

  The ideas and intuitions behind these conditions are the same as in the case of clausal models.
- At this point, we extend the definition of a literal disjunction being true in a clausal model to the case of formulae in prenex conjunctive normal form. In order to explain basic ideas behind, let us investigate the following simple cases:
  - Suppose that a quantified atom of the form $\forall x \, \exists y \, p(x, y)$ is true in a first-order model A, in symbols $\mathrm{A} \models \forall x \, \exists y \, p(x, y)$. This means that for every element u_x in the domain $\mathcal{U}_{\mathrm{A}}$, there exists an element u_y in $\mathcal{U}_{\mathrm{A}}$ such that $p^{\mathrm{A}}(u_x, u_y)$, where the binary relation p^{A} is the interpretation of the binary predicate symbol p in the model. By the axiom of choice, we can equivalently state that there exists a Skolem function $s : \mathcal{U}_{\mathrm{A}} \to \mathcal{U}_{\mathrm{A}}$ such that for every $u_x \in \mathcal{U}_{\mathrm{A}}$, $p^{\mathrm{A}}(u_x, s(u_x)$ holds. In other words, the Skolem function s gives us, for every element $u_x \in \mathcal{U}_{\mathrm{A}}$, some witness $s(u_x) \in \mathcal{U}_{\mathrm{A}}$ so that $p^{\mathrm{A}}(u_x, s(u_x)$ holds. Our first aim is to define when the formula $\forall x \, \exists y \, p(x, y)$ holds in a disjunctive model, say M. We shall use a weakened notion of a Skolem function. In place of the only witness, a weakened Skolem function s' provides us, for every element $u_x \in \mathcal{U}_{\mathrm{M}}$, a non-empty finite set $s'(u_x) \subseteq \mathcal{U}_{\mathrm{M}}$ of candidates aspiring to become a witness for y. Although we do not know a right witness, we know that at least one of the candidates for the witness succeeds. This implies that the finite set of triples

$ST = \overline{\bigvee_{c_y \in s'(u_x)} p(u_x, c_y)}$ should hold in M, i.e. ST should be subsumed by some $b \in \mathcal{B}_{\mathrm{M}}$, $b \subseteq ST$, or ST should contain a pair of contradictory triples. Hence, to find out if $\forall x\, \exists y\, p(x, y)$ holds in a disjunctive model M, we have to examine whether for every element $u_x \in \mathcal{U}_{\mathrm{M}}$, the finite set of triples $\overline{\bigvee_{c_y \in s'(u_x)} p(u_x, c_y)}$ is satisfied in M.

- ∘ Investigate the case when $\exists x\, \exists y\, p(x, y)$ should be true in a disjunctive model M. This time, the weakened Skolem function for x should be a constant s_x, similarly to first-order models, but that for y, s_y, should be a unary function. Since s_x gives us a non-empty finite set of distinct candidates for a witness substituted for x, instead of the only witness, the function s_y has to provide another set of candidates for a witness for y depending on each candidate in s_x; therefore, the function s_y must be a unary function, in contrast to the case of first-order models, where s_y would be a constant. Hence, in order to decide if $\mathrm{M} \models \exists x\, \exists y\, p(x, y)$, we have to check up whether the finite set of triples $\overline{\bigvee_{c_x \in s_x} \bigvee_{c_y \in s_y(c_x)} p(c_x, c_y)}$ is satisfied in M.
- ∘ In the end, consider the case $\mathrm{M} \models^{\vee} \exists x\, \forall y\, p(x, y)$ where M is a clausal model. This implies that there exists some weakened Skolem function for x, a constant s_x. We may rewrite the given formula on the equivalent disjunction $\bigvee_{c_x \in s_x} \forall y\, p(c_x, y)$ of the language $\mathcal{L} \cup \mathcal{U}_{\mathrm{M}}$ (new added constants). The formula $\forall y\, p(c_x, y)$ may be replaced by the equivalent conjunction $\bigwedge_{y \in \mathcal{U}_{\mathrm{M}}} p(c_x, y)$ of $\mathcal{L} \cup \mathcal{U}_{\mathrm{M}}$. This yields the formula $\bigvee_{c_x \in s_x} \bigwedge_{y \in \mathcal{U}_{\mathrm{M}}} p(c_x, y)$ of $\mathcal{L} \cup \mathcal{U}_{\mathrm{M}}$. But this formula is not still a disjunction of atoms (literals). However, we can make use of the distributive law and replace the last formula by the equivalent set of atomic (literal) disjunctions: $\{\bigvee_{c_x \in s_x} p(c_x, u(c_x)) \mid u : \mathcal{U}_{\mathrm{M}} \to \mathcal{U}_{\mathrm{M}}\ \textit{is a function}\}$. So, we may define $\mathrm{M} \models^{\vee} \exists x\, \forall y\, p(x, y)$ if and only if for every function $u : \mathcal{U}_{\mathrm{M}} \to \mathcal{U}_{\mathrm{M}}$, whose values are substituted for universally quantified variable y, the finite set of triples $\overline{\bigvee_{c_x \in s_x} p(c_x, u(c_x))}$ holds in M. Hence, we have introduced another kind of auxiliary functions to the definition, the distributive functions u's.
- ∘ Similarly, we shall also use index functions. Define when $\mathrm{M} \models^{\vee} \exists x\, p_1(x) \wedge p_2(x)$. There should be a weakened Skolem function for x, a constant s_x so that we may rewrite the given formula on the disjunction $\bigvee_{c_x \in s_x} p_1(c_x) \wedge p_2(c_x)$ of $\mathcal{L} \cup \mathcal{U}_{\mathrm{M}}$. We use again the distributive law and, in addition, index functions. The last formula can be replaced by the equivalent set of atomic (literal) disjunctions: $\{\bigvee_{c_x \in s_x} p_{idx(c_x)}(c_x) \mid idx : \mathcal{U}_{\mathrm{M}} \to \{1, 2\}\ \textit{is a function}\}$. Hence, we may define $\mathrm{M} \models^{\vee} \exists x\, p_1(x) \wedge p_2(x)$ if and only if for every index function $idx : \mathcal{U}_{\mathrm{M}} \to \{1, 2\}$, the finite set of triples $\overline{\bigvee_{c_x \in s_x} p_{idx(c_x)}(c_x)}$ holds in M.

So, we have explained the main ideas behind the generalisation. We further formalise these ideas in a general setting. Consider a prenex conjunctive normal form of $\mathcal{L}$:

$$PCNF = \forall x_1^1 \ldots \forall x_{m_1}^1\, \exists y^1 \ldots \forall x_1^n \ldots \forall x_{m_n}^n\, \exists y^n\, \forall x_1^{n+1} \ldots \forall x_{m_{n+1}}^{n+1} \bigwedge_{i \in I} D_i$$

where $x_1^0, \ldots, x_{m_0}^0$ are all the free variables in $PCNF$; $0 \le n$, $m_0, \ldots, m_{n+1}$; the x's, y's are pairwise different variables; and $\bigwedge_{i \in I} D_i$ is a conjunctive normal form. For the existentially quantified variables $y^1, \ldots, y^n$, we shall use weakened Skolem functions $s^1, \ldots, s^n$. The values of the function s^i will be substituted for the variable y^i in the resulting finite set of triples, say ST, and the function s^i will

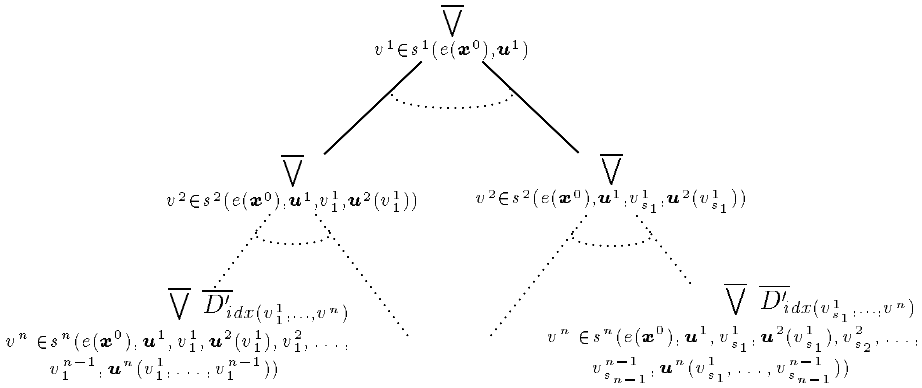


FIG. 1. The set of triples ST.

depend on the values substituted for the free variables $x^0_1, \dots, x^0_{m_0}$ and the bound variables standing before y^i in the prenex. For the universally quantified variables x^i_j's, we shall use distributive functions u^i_j's. The values of the function u^i_j will be substituted for the variable x^i_j in the resulting ST and the function u^i_j will have as arguments the values substituted for the existentially bound variables standing before x^i_j in the prenex. We shall also use index functions idx's which will index the literal disjunctions D's from the conjunctive normal form $\bigwedge_{i \in I} D_i$. The index functions will depend on the values substituted for all the existentially quantified variables $y^1, \dots, y^n$. So, we are in position to give the definition: The prenex conjunctive normal form $PCNF$ is true in M with respect to a variable assignment e in M, written as M $\models^{\vee}$ $PCNF[e]$, if and only if there exist weakened Skolem functions $s^1, \dots, s^n$, $s^i : \mathcal{U}_{\mathrm{M}}^{m_0+\dots+m_i+i-1} \to 2^{\mathcal{U}_{\mathrm{M}}}_{\mathcal{F}+}$, such that for every distributive functions $u^i_j : \mathcal{U}_{\mathrm{M}}^{i-1} \to \mathcal{U}_{\mathrm{M}}$ $(i = 1, \dots, n+1,\ j = 1, \dots, m_i)$ and for every index function $idx : \mathcal{U}^n_{\mathrm{M}} \to I$, the following finite set of triples

$$ST = \overline{\bigvee_{\substack{v^1 \in s^1(e(\boldsymbol{x}^0), \boldsymbol{u}^1) \\ v^2 \in s^2(e(\boldsymbol{x}^0), \boldsymbol{u}^1, v^1, \boldsymbol{u}^2(v^1)) \\ \dots \\ v^n \in s^n(e(\boldsymbol{x}^0), \boldsymbol{u}^1, v^1, \boldsymbol{u}^2(v^1), v^2, \dots, v^{n-1}, \boldsymbol{u}^n(v^1, \dots, v^{n-1}))}}} \ \overline{\begin{array}{r} D_{idx(v^1,\dots,v^n)}\{\boldsymbol{x}^0/e(\boldsymbol{x}^0),\ \boldsymbol{x}^1/\boldsymbol{u}^1,\ y^1/v^1,\ \boldsymbol{x}^2/\boldsymbol{u}^2(v^1),\ y^2/v^2, \dots, \\ \boldsymbol{x}^n/\boldsymbol{u}^n(v^1, \dots, v^{n-1}),\ y^n/v^n, \\ \boldsymbol{x}^{n+1}/\boldsymbol{u}^{n+1}(v^1, \dots, v^n)\} \end{array}}$$

either is subsumed by some $b \in \mathcal{B}_{\mathrm{M}}$, $b \subseteq ST$, or ST contains a pair of contradictory triples. The set of triples ST is outlined as a tree in Figure 1.

- A prenex conjunctive normal form of $\mathcal{L}$ $PCNF$ is true in M, denoted as M $\models^{\vee}$ $PCNF$, if and only if it is true in M with respect to every variable assignment e in M, in symbols $\forall e$ M $\models^{\vee}$ $PCNF[e]$.

- All the equality axioms in $Eq_{\mathcal{L}}$ must be true in M, i.e. $\mathrm{M} \models^{\vee} Eq_{\mathcal{L}}$ where

$$
\begin{aligned}
Eq_{\mathcal{L}} = &\{x = x\} \ \cup \\
&\{p(y_1, \ldots, y_{ar(p)}) \vee \neg p(x_1, \ldots, x_{ar(p)}) \vee x_1 \neq y_1 \vee \cdots \vee x_{ar(p)} \neq y_{ar(p)} \\
&\quad | \, p \in Pred_{\mathcal{L}}\} \ \cup \\
&\{f(x_1, \ldots, x_{ar(f)}) = f(y_1, \ldots, y_{ar(f)}) \vee x_1 \neq y_1 \vee \cdots \vee x_{ar(f)} \neq y_{ar(f)} \\
&\quad | \, f \in Func_{\mathcal{L}}, \, ar(f) > 0\}.
\end{aligned}
$$

  The x's and y's are pairwise different variables. We suppose that the equality symbol $=$ belongs to $Pred_{\mathcal{L}}$.
- An arbitrary formula F of $\mathcal{L}$ is true in M, written as $\mathrm{M} \models^{\vee} F$, if and only if some of its prenex conjunctive normal forms is true in M.

We next relate disjunctive models to first-order models. The following theorem holds:

THEOREM 1.2
Let T be a first-order theory of $\mathcal{L}$ and C a formula of $\mathcal{L}$.

i) For every disjunctive model M for $\mathcal{L}$, $\mathrm{M} \models^{\vee} T$ implies $\mathrm{M} \models^{\vee} C$ if and only if $T \models C$ (C is a logical consequence of T).
ii) T has a disjunctive model for $\mathcal{L}$ if and only if T has a first-order model for $\mathcal{L}$.

PROOF. Similar to the proof of Theorem 2.2 in [6]. ∎

2 Characterisation Theorem

We firstly recall the definitions of a characteristic model and the closure operator $\mathcal{C}_T$ from [6]. Secondly, we introduce the notions of modified skolemisation and an $\wedge$-expansion. Further, the characterisation theorem will be stated.

In the Introduction, we have claimed that one of our aims is to achieve the equivalence between the model and declarative semantics. Since the model semantics should be based on one model, this model has to satisfy the following important condition - namely, it has to be characteristic:

DEFINITION 2.1
Let T be a first-order theory of $\mathcal{L}$. A disjunctive model M for $\mathcal{L}$ is a characteristic disjunctive model of T if and only if for any formula C of $\mathcal{L}$,

$$\mathrm{M} \models^{\vee} C \Longleftrightarrow T \models C.$$

Note that this condition implies $\mathrm{M} \models^{\vee} T$.

DEFINITION 2.2
By $\mathcal{D}_{\mathcal{L}}$ we denote the set of all literal disjunctions of $\mathcal{L}$. Let T be a set of literal disjunctions of $\mathcal{L}$. The closure operator $\mathcal{C}_T$ is defined as follows: $\mathcal{C}_T : 2^{\mathcal{D}_{\mathcal{L}}} \longrightarrow 2^{\mathcal{D}_{\mathcal{L}}}$,

$\mathcal{C}_T(I) = T \cup \{(a_1 \vee \cdots \vee a_m \vee b_1 \vee \cdots \vee b_n)\theta' \mid$

- where $a_1 \vee \cdots \vee a_m \vee A$ and $b_1 \vee \cdots \vee b_n \vee B$ are some renamed permutations of literal disjunctions in I not sharing common variables (A, B are literal disjunctions);
- θ is an *mgu* for the set of literals $A \cup \neg B$, $dom(\theta) = vars(A, B)$;
- θ' is its regular extension to $vars(a_1 \vee \cdots \vee a_m \vee A,\ b_1 \vee \cdots \vee b_n \vee B)\}$.

It is evident that $\mathcal{C}_T$ is monotonic and continuous.

In the following definition, we describe modified skolemisation, a technique which allow us to avoid the addition of new Skolem function symbols to the language $\mathcal{L}$ of a disjunctive quantified program. The technique is based on the use of term-variables instead of new function symbols. Term-variables are specially denoted variables of the form $\tilde{y}^n_m(t_1, \ldots, t_n)$ where $n, m \geq 0$ and $t_1, \ldots, t_n$ are terms of $\mathcal{L}$. If the set of function symbols of $\mathcal{L}$ is countable, then so is the set of term-variables. In the following text, we shall distinguish between term-variables and common variables called just variables.

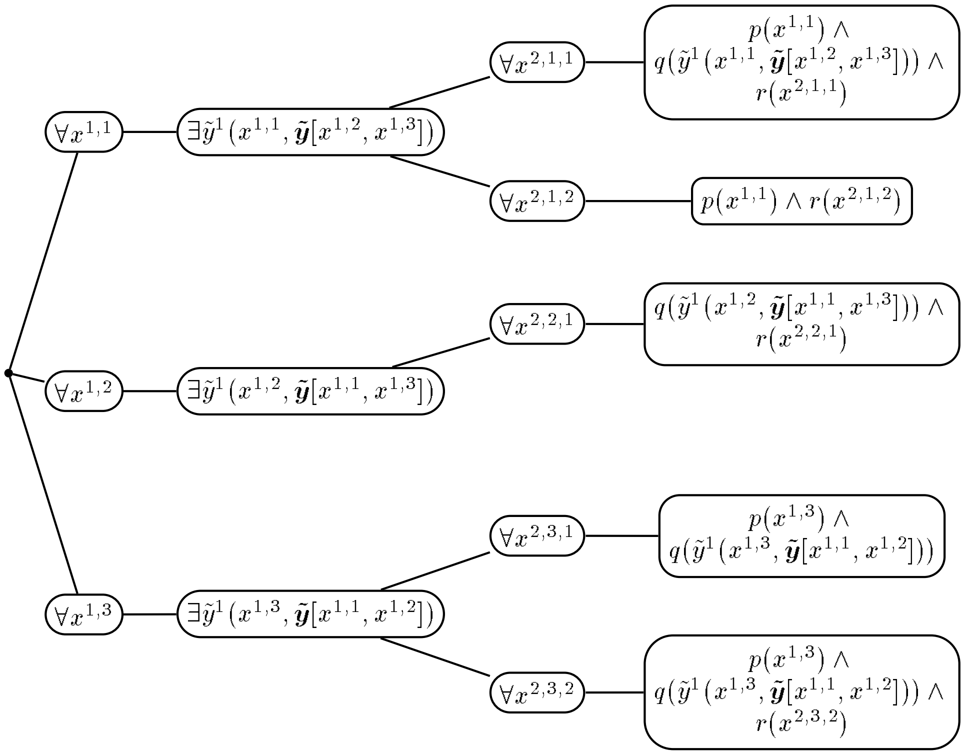
DEFINITION 2.3 (Modified Skolemisation)
Let $C = \forall x^1_1 \ldots \forall x^1_{m_1}\, \exists y^1 \ldots \forall x^n_1 \ldots \forall x^n_{m_n}\, \exists y^n\, \forall x^{n+1}_1 \ldots \forall x^{n+1}_{m_{n+1}}\, B$ be a prenex formula of $\mathcal{L}$ with free variables $x^0_1, \ldots, x^0_{m_0}$ where $0 \leq n, m_0, \ldots, m_{n+1}$; the x's, y's are pairwise different variables; and B is a quantifier-free formula. A skolemisation of C is the formula

$$C^S = B\{y^1/\tilde{y}^{m_0+m_1+1}_1(x^0_1, \ldots, x^1_{m_1}, v_1),\ y^2/\tilde{y}^{m_0+m_1+m_2+1}_2(x^0_1, \ldots, x^2_{m_2}, v_2),\\ \ldots, y^n/\tilde{y}^{m_0+m_1+\cdots+m_n+1}_n(x^0_1, \ldots, x^n_{m_n}, v_n)\}$$

where the $\tilde{y}(\ldots)$'s and v's are pairwise different term-variables and variables, respectively, all not occurring in C.
Let T be a set of prenex formulae. A skolemisation of T is the set $T^S = \{C^S \mid C \in T\}$ where the used term-variables $\tilde{y}(\ldots)$'s and variables v's are pairwise different and not occurring in T.

In order to formulate the generalised statement of Herbrand's theorem, we need the notion of an $\wedge$-expansion (see the definition in the full paper). Let C be a prenex conjunctive normal form without free variables. An $\wedge$-expansion $C^\wedge$ of C may be conceived as a tree consisting of inner nodes labeled by quantifiers with quantified variables and leaves by subconjunctions of C's matrix (a conjunctive normal form). Branches of the tree are formed by inner nodes labeled with renamed C's preneces; and after splitting, always before an existentially quantified variable, they do not share common variables. For existentially quantified variables in C, we shall use special term-variables. Each branch ends in a leaf labeled by a subconjunction of C's matrix renamed in the corresponding manner to this branch.

Consider the following example. Let $C = \forall x^1\, \exists y^1\, \forall x^2\, (p(x^1) \wedge q(y^1) \wedge r(x^2))$, then one of C's $\wedge$-expansions has the tree form given below:



Further, the tree is rewritten onto a prenex conjunctive normal form layer by layer. Moreover, we add to the matrix the equality axioms for the term-variables $\tilde{y}(\dots)$ used for renaming existentially quantified variables.

$$
\begin{gathered}
\forall x^{1,1}\,\forall x^{1,2}\,\forall x^{1,3}\,\exists \tilde{y}^1(x^{1,1}, \tilde{\boldsymbol{y}}[x^{1,2}, x^{1,3}])\,\exists \tilde{y}^1(x^{1,2}, \tilde{\boldsymbol{y}}[x^{1,1}, x^{1,3}])\,\exists \tilde{y}^1(x^{1,3}, \tilde{\boldsymbol{y}}[x^{1,1}, x^{1,2}]) \\
\forall x^{2,1,1}\,\forall x^{2,1,2}\,\forall x^{2,2,1}\,\forall x^{2,3,1}\,\forall x^{2,3,2} \\
\left(
\begin{array}{l}
\left(\begin{array}{c} p(x^{1,1}) \wedge \\ q(\tilde{y}^1(x^{1,1}, \tilde{\boldsymbol{y}}[x^{1,2}, x^{1,3}])) \wedge \\ r(x^{2,1,1}) \end{array}\right) \wedge p(x^{1,1}) \wedge r(x^{2,1,2}) \wedge \\[2ex]
\left(\begin{array}{c} q(\tilde{y}^1(x^{1,2}, \tilde{\boldsymbol{y}}[x^{1,1}, x^{1,3}])) \wedge \\ r(x^{2,2,1}) \end{array}\right) \wedge \\[2ex]
\left(\begin{array}{c} p(x^{1,3}) \wedge \\ q(\tilde{y}^1(x^{1,3}, \tilde{\boldsymbol{y}}[x^{1,1}, x^{1,2}])) \end{array}\right) \wedge \left(\begin{array}{c} p(x^{1,3}) \wedge \\ q(\tilde{y}^1(x^{1,3}, \tilde{\boldsymbol{y}}[x^{1,1}, x^{1,2}])) \wedge \\ r(x^{2,3,2}) \end{array}\right) \wedge \\[2ex]
\left(x^{1,1} \neq x^{1,2} \vee \tilde{y}^1(x^{1,1}, \tilde{\boldsymbol{y}}[x^{1,2}, x^{1,3}]) = \tilde{y}^1(x^{1,2}, \tilde{\boldsymbol{y}}[x^{1,1}, x^{1,3}])\right) \wedge \\[2ex]
\left(x^{1,1} \neq x^{1,3} \vee \tilde{y}^1(x^{1,1}, \tilde{\boldsymbol{y}}[x^{1,2}, x^{1,3}]) = \tilde{y}^1(x^{1,3}, \tilde{\boldsymbol{y}}[x^{1,1}, x^{1,2}])\right) \wedge \\[2ex]
\left(x^{1,2} \neq x^{1,3} \vee \tilde{y}^1(x^{1,2}, \tilde{\boldsymbol{y}}[x^{1,1}, x^{1,3}]) = \tilde{y}^1(x^{1,3}, \tilde{\boldsymbol{y}}[x^{1,1}, x^{1,2}])\right)
\end{array}
\right)
\end{gathered}
$$

Note that each existentially quantified term-variable $\tilde{y}^1(\ldots)$ contains all the universally quantified variables x's standing before it in the prenex. This fact will be used to ensure 'automatic' renaming these existentially quantified term-variables at instantiation of an $\wedge$-expansion (see definition of $\wedge$-expansion in the full paper) so that the instance will be a logical consequence of the given $\wedge$-expansion.

We are in position to state the characterisation theorem:

THEOREM 2.4 (**Characterisation Theorem**)
Let P be a quantified disjunctive program of $\mathcal{L}$.

i) If P has a clausal model of $\mathcal{L}$, then P has a characteristic clausal model of $\mathcal{L}$.

ii) (**Herbrand's Theorem**) Let C be a prenex formula of $\mathcal{L}$.

$$C = \forall x_1^1 \ldots \forall x_{m_1}^1 \, \exists y^1 \ldots \forall x_1^n \ldots \forall x_{m_n}^n \, \exists y^n \, \forall x_1^{n+1} \ldots \forall x_{m_{n+1}}^{n+1} \, B$$

where $x_1^0, \ldots, x_{m_0}^0$ are all the free variables in C, $0 \leq n$, $m_0, \ldots, m_{n+1}$; the x's, y's are pairwise different variables; and B is a quantifier-free formula.

$$P \models C \text{ if and only if there exist}$$

1) formulae $F_1, \ldots, F_k$ $(k \geq 0)$, some renamed and universally closed prenex conjunctive normal forms of $P \cup Eq_{\mathcal{L}}$ not sharing variables mutually; note that $P \models \bigwedge_{i=1,\ldots,k} F_i$;

2) a prenex conjunctive normal form F obtained from $F_1, \ldots, F_k$ by the concatenation of their preneces and the conjunction of their matrices:

$$F = \mathcal{P}renex_1 \ldots \mathcal{P}renex_k \bigwedge_{i=1,\ldots,k} CNF_i$$

where $\mathcal{P}renex_i$ is the prenex of F_i and CNF_i is the matrix of F_i; note that $\models \bigwedge_{i=1,\ldots,k} F_i \longrightarrow F$;

3) an $\wedge$-expansion $F^\wedge$ of F and its instance $IF^\wedge$ of the form:

$$IF^\wedge = \mathcal{P}renex \bigwedge_{i=1,\ldots,r} if_i^\wedge;$$

note that $\models F \longrightarrow F^\wedge$, $F^\wedge \models IF^\wedge$;

4) and tuples $T_1, \ldots, T_s$ $(s \geq 1)$ of terms of $\mathcal{L}$ such that

$$\begin{aligned} &\bigwedge_{i=1,\ldots,r} if_i^\wedge \models_{Taut} \\ &\bigvee_{(y^1,\ldots,y^n)\in\{T_1,\ldots,T_s\}} B\{x_1^0/z_1^0, \ldots, x_{m_0}^0/z_{m_0}^0,\, x_1^1/z_{m_0+1}^0, \ldots, x_{m_1}^1/z_{m_0+m_1}^0, \\ &\qquad x_1^2/z_1^1(y^1), \ldots, x_{m_2}^2/z_{m_2}^1(y^1), \ldots, \\ &\qquad x_1^{n+1}/z_1^n(y^1, \ldots, y^n), \ldots, x_{m_{n+1}}^{n+1}/z_{m_{n+1}}^n(y^1, \ldots, y^n)\} \end{aligned} \tag{1}$$

where $z_b^a(t_1^i, \ldots, t_a^i)$'s are specially denoted pairwise different common variables; $(t_1^i, \ldots, t_a^i)$ is the prefix of the tuple of terms $T_i = (t_1^i, \ldots, t_n^i)$; and the variable $z_b^a(t_1^i, \ldots, t_a^i)$ does not appear in the terms $t_1^i, \ldots, t_a^i$.

PROOF. In the full paper. ∎

3 Semantics

We now introduce the intended semantics. We recall the declarative one:

DEFINITION 3.1 (**Declarative Semantics**)
Let P be a quantified disjunctive program of $\mathcal{L}$.

$$\mathcal{DS}(P) = \{C \mid C \textit{ is a prenex conjunctive normal form of } \mathcal{L} \textit{ and } P \models C\}.$$

Using the characteristic disjunctive model M constructed in the proof of the characterisation theorem, we can propose the model semantics.

DEFINITION 3.2 (**Model Semantics**)
Let P be a quantified disjunctive program of $\mathcal{L}$.

- If there exists a disjunctive model of P for $\mathcal{L}$. (P is consistent.)
  $\mathcal{MS}(P) = \{C \mid C \textit{ is a prenex conjunctive normal form of } \mathcal{L} \textit{ and } \mathrm{M} \models^{\vee} C\}$ where $\mathrm{M} = (\mathcal{U}_{\mathrm{M}}, \mathcal{B}_{\mathrm{M}})$ is a characteristic disjunctive model of P for $\mathcal{L}$ defined as follows:
  - ∘ Let $\mathcal{A}_{\mathrm{M}} = \{f_n^m \mid m, n \in \omega\}$ where f_n^m's are new pairwise different m-ary function symbols not occurring in $\mathcal{L}$.
    Let P^S be a set of literal disjunctions of $\mathcal{L}$ obtained from a modified skolemisation of P by omitting all $\wedge$ connectives. Let S be the set of all new term-variables used in this skolemisation.
    We put $\mathcal{U}_{\mathrm{M}} = \mathit{Term}_{\mathit{Func}_{\mathcal{L}} \cup \mathcal{A}_{\mathrm{M}} \cup S}$ - i.e. $\mathcal{U}_{\mathrm{M}}$ is the set of all terms constructed by symbols in $\mathit{Func}_{\mathcal{L}} \cup \mathcal{A}_{\mathrm{M}} \cup S$. Since $\mathcal{A}_{\mathrm{M}}$ contains constants f_n^0's, $\mathcal{U}_{\mathrm{M}} \neq \emptyset$.
    A function symbol $f \in \mathit{Func}_{\mathcal{L}}$ is interpreted as the function $f^{\mathrm{M}} : \mathcal{U}_{\mathrm{M}}^{ar(f)} \to \mathcal{U}_{\mathrm{M}}$, $f^{\mathrm{M}}(u_1, \ldots, u_{ar(f)}) = f(u_1, \ldots, u_{ar(f)})$ where $u_1, \ldots, u_{ar(f)} \in \mathcal{U}_{\mathrm{M}}$.
  - ∘ $\mathcal{B}_{\mathrm{M}} = \{\overline{b} \mid$
    - · $\exists d \in \mathcal{C}_{P^S \cup Eq_{\mathcal{L} \cup S}} \uparrow \omega$ such that b, $vars(b) = \emptyset$, is an instance of d in the language $\mathcal{L} \cup \mathcal{A}_{\mathrm{M}}$;
    - · $\nexists b'$, $vars(b') = \emptyset$, being an instance of d' in the language $\mathcal{L} \cup \mathcal{A}_{\mathrm{M}}$ where $d' \in \mathcal{C}_{P^S \cup Eq_{\mathcal{L} \cup S}} \uparrow \omega$ so that $b' \sqsubset b$ (b is a proper superdisjunction of b');
    - · b is not a tautology$\}$.
- If there does not exist a disjunctive model of P for $\mathcal{L}$. (P is inconsistent.)
  $\mathcal{MS}(P) = \{C \mid C \textit{ is a prenex conjunctive normal form of } \mathcal{L}\}$.

By the closure operator $\mathcal{C}$ and its fixpoint $\mathcal{C}_{P^S \cup Eq_{\mathcal{L} \cup S}} \uparrow \omega$, used for constructing the characteristic model M, we can establish the fixpoint semantics.

DEFINITION 3.3 (**Fixpoint Semantics**)
Let P be a quantified disjunctive program of $\mathcal{L}$.
$\mathcal{FS}(P) = \{C \mid$

- *C is a prenex conjunctive normal form of $\mathcal{L}$:*
  $$C = \forall x_1^1 \ldots \forall x_{m_1}^1 \, \exists y^1 \ldots \forall x_1^n \ldots \forall x_{m_n}^n \, \exists y^n \, \forall x_1^{n+1} \ldots \forall x_{m_{n+1}}^{n+1} \bigwedge_{i \in I} D_i$$
  where $x_1^0, \ldots, x_{m_0}^0$ are all the free variables in C; $0 \le n, m_0, \ldots, m_{n+1}$; the x's, y's are pairwise different variables; and $\bigwedge_{i \in I} D_i$ is a conjunctive normal form;
- *there exist n-ary tuples $T_1, \ldots, T_s$, $s \ge 1$, of terms of $\mathcal{L}$;*
- *there exist $d_1, \ldots, d_{|I|^s}$ not sharing variables mutually and with C where each d_i is either a variant (of $\mathcal{L}$) of a literal disjunction in $\mathcal{C}_{P^S \cup Eq_{\mathcal{L} \cup S}} \uparrow \omega$ or a tautology of $\mathcal{L}$; P^S is a set of literal disjunctions of $\mathcal{L}$ obtained from a modified skolemisation of P by omitting all $\wedge$ connectives; S is the set of all new term-variables used in this skolemisation;*

- *for every literal disjunction of the form*

$$D = \bigvee_{(y^1,\ldots,y^n)\in\{T_1,\ldots,T_s\}} D_{i(y^1,\ldots,y^n)}\{x_1^0/z_1^0,\ldots,x_{m_0}^0/z_{m_0}^0,\; x_1^1/z_{m_0+1}^0,\ldots,x_{m_1}^1/z_{m_0+m_1}^0,$$
$$x_1^2/z_1^1(y^1),\ldots,x_{m_2}^2/z_{m_2}^1(y^1),\ldots,$$
$$x_1^{n+1}/z_1^n(y^1,\ldots,y^n),\ldots,x_{m_{n+1}}^{n+1}/z_{m_{n+1}}^n(y^1,\ldots,y^n)\}$$

[where $i(y^1,\ldots,y^n)\in I$; $z_b^a(t_1^i,\ldots,t_a^i)$'s are specially denoted pairwise different variables; $(t_1^i,\ldots,t_a^i)$ is the prefix of the tuple $T_i=(t_1^i,\ldots,t_n^i)$; and the variable $z_b^a(t_1^i,\ldots,t_a^i)$ does not appear in the terms $t_1^i,\ldots,t_a^i$], there exists an instance d_i' (of $\mathcal{L}$) of some d_i satisfying $d_i'\sqsubseteq D$ (D is a superdisjunction of d_i').

The generalised Herbrand's theorem gives us some characterisation of logical consequences of a quantified disjunctive program in terms of tautological following. We can introduce the Herbrand semantics.

Definition 3.4 (**Herbrand Semantics**)
Let P be a quantified disjunctive program of $\mathcal{L}$.
$\mathcal{HS}(P)=\{C\,|\,C$ *is a prenex conjunctive normal form of* $\mathcal{L}$ *and the conditions* 1)-4) *from Herbrand's theorem for the program* P *are satisfied*$\}$.

As a corollary of Theorem 2.4, we get the coincidence of the proposed semantics.

Corollary 3.5
Let P be a quantified disjunctive program of $\mathcal{L}$.

$$\mathcal{DS}(P)=\mathcal{MS}(P)=\mathcal{FS}(P)=\mathcal{HS}(P).$$

Proof. An immediate consequence of the characterisation theorem. ∎

4 Conclusion

In the paper, we have introduced a class of disjunctive models, a 'conservative' extension of first-order models. In terms of disjunctive models and their characterisation theorem, we have proposed the model, fixpoint, and Herbrand semantics for quantified disjunctive programs with equality, sets of formulae in prenex conjunctive normal form. In the end, the mutual coincidence of these semantics has been shown.

A Appendix

Definition A.1 ($\bigwedge$-expansion and Its Instance)
Let $C=\forall x_1^1\ldots\forall x_{m_1}^1\,\exists y^1\ldots\forall x_1^n\ldots\forall x_{m_n}^n\,\exists y^n\,\forall x_1^{n+1}\ldots\forall x_{m_{n+1}}^{n+1}\bigwedge_{i\in I}D_i$ be a prenex conjunctive normal form of $\mathcal{L}$ without free variables where $0\le n,\,m_1,\ldots,m_{n+1}$; the x's and y's are pairwise different variables; and $\bigwedge_{i\in I}D_i$ is a conjunctive normal form. For the sake of simplicity, denote $\forall x_1^{j,\alpha}\ldots\forall x_{m_j}^{j,\alpha}$ by $\forall \boldsymbol{x}^{j,\alpha}$ where α is some string of numbers. An $\wedge$-expansion $C^\wedge$ of C is the following prenex conjunctive normal form of $\mathcal{L}$:

$$
\begin{gathered}
C^{\wedge} = \quad \forall \boldsymbol{x}^{1,1} \ldots \forall \boldsymbol{x}^{1,k^1_1} \\
\exists \tilde{y}^1(\boldsymbol{x}^{1,1}, v^{1,1}) \ldots \exists \tilde{y}^1(\boldsymbol{x}^{1,k^1_1}, v^{1,k^1_1}) \\
\forall \boldsymbol{x}^{2,1,1} \ldots \forall \boldsymbol{x}^{2,1,k^2_1} \ldots \\
\forall \boldsymbol{x}^{2,k^1_1,1} \ldots \forall \boldsymbol{x}^{2,k^1_1,k^2_{k^1_1}} \\
\exists \tilde{y}^2(\boldsymbol{x}^{1,1}, \boldsymbol{x}^{2,1,1}, v^{2,1,1}) \ldots \\
\exists \tilde{y}^2(\boldsymbol{x}^{1,1}, \boldsymbol{x}^{2,1,k^2_1}, v^{2,1,k^2_1}) \ldots \\
\exists \tilde{y}^2(\boldsymbol{x}^{1,k^1_1}, \boldsymbol{x}^{2,k^1_1,1}, v^{2,k^1_1,1}) \ldots \\
\exists \tilde{y}^2(\boldsymbol{x}^{1,k^1_1}, \boldsymbol{x}^{2,k^1_1,k^2_{k^1_1}}, v^{2,k^1_1,k^2_{k^1_1}}) \\
\forall \boldsymbol{x}^{3,1,1,1} \ldots \forall \boldsymbol{x}^{3,k^1_1,k^2_{k^1_1},k^3_{s^3}} \\
\exists \tilde{y}^3(\boldsymbol{x}^{1,1}, \boldsymbol{x}^{2,1,1}, \boldsymbol{x}^{3,1,1,1}, v^{3,1,1,1}) \ldots \\
\exists \tilde{y}^3(\boldsymbol{x}^{1,k^1_1}, \boldsymbol{x}^{2,k^1_1,k^2_{k^1_1}}, \boldsymbol{x}^{3,k^1_1,k^2_{k^1_1},k^3_{s^3}}, v^{3,k^1_1,k^2_{k^1_1},k^3_{s^3}}) \\
\ldots \\
\forall \boldsymbol{x}^{n,\underbrace{1,\ldots,1}_{n}} \ldots \forall \boldsymbol{x}^{n,k^1_1,k^2_{k^1_1},k^3_{s^3},\ldots,k^n_{s^n}} \\
\exists \tilde{y}^n(\boldsymbol{x}^{1,1}, \ldots, \boldsymbol{x}^{n,\underbrace{1,\ldots,1}_{n}}, v^{n,\underbrace{1,\ldots,1}_{n}}) \ldots \\
\exists \tilde{y}^n(\boldsymbol{x}^{1,k^1_1}, \ldots, \boldsymbol{x}^{n,k^1_1,k^2_{k^1_1},k^3_{s^3},\ldots,k^n_{s^n}}, v^{n,k^1_1,k^2_{k^1_1},k^3_{s^3},\ldots,k^n_{s^n}}) \\
\forall \boldsymbol{x}^{n+1,\underbrace{1,\ldots,1}_{n+1}} \ldots \forall \boldsymbol{x}^{n+1,k^1_1,k^2_{k^1_1},k^3_{s^3},\ldots,k^{n+1}_{s^{n+1}}} \\
\left(
\begin{array}{ccc}
 & \displaystyle\bigwedge_{i \in I^{n+1,\underbrace{1,\ldots,1}_{n+1}}} & D_i \vartheta^{n+1,\underbrace{1,\ldots,1}_{n+1}} \wedge \cdots \wedge \\
 & \displaystyle\bigwedge_{i \in I^{n+1,k^1_1,k^2_{k^1_1},k^3_{s^3},\ldots,k^{n+1}_{s^{n+1}}}} & D_i \vartheta^{n+1,k^1_1,k^2_{k^1_1},k^3_{s^3},\ldots,k^{n+1}_{s^{n+1}}} \wedge \\
\displaystyle\bigwedge_{j=1}^{n} & \displaystyle\bigwedge_{\substack{l_i \le l'_i \\ 1 \le l_i \le k^i_{q^i} \\ 1 \le l'_i \le k^i_{q'^i} \\ i=1,\ldots,j \\ \exists i^*(1 \le i^* \le j \wedge \\ l_1 = l'_1 \wedge \cdots \wedge l_{i^*-1} = l'_{i^*-1} \wedge \\ l_{i^*} < l'_{i^*})}} & \begin{array}{c} \boldsymbol{x}^{i^*,l_1,\ldots,l_{i^*}} \neq \boldsymbol{x}^{i^*,l'_1,\ldots,l'_{i^*}} \vee \cdots \vee \\ \boldsymbol{x}^{j,l_1,\ldots,l_{i^*},\ldots,l_j} \neq \boldsymbol{x}^{j,l'_1,\ldots,l'_{i^*},\ldots,l'_j} \vee \\ \tilde{y}^j(\boldsymbol{x}^{1,l_1}, \ldots, \boldsymbol{x}^{j,l_1,\ldots,l_j}, v^{j,l_1,\ldots,l_j}) \\ = \\ \tilde{y}^j(\boldsymbol{x}^{1,l'_1}, \ldots, \boldsymbol{x}^{j,l'_1,\ldots,l'_j}, v^{j,l'_1,\ldots,l'_j}) \end{array}
\end{array}
\right)
\end{gathered}
$$

The used x's and $\tilde{y}(\ldots)$'s are pairwise different variables and term-variables, respectively. We put $s^1 = 1$, $s^2 = k^1_1$ and define $s^j = \sum_{r=1}^{s^{j-1}} k^{j-1}_r$ for $j = 3, \ldots, n+1$. Similarly, $q^1 = 1$, $q^i = l_{i-1} + \sum_{r=1}^{q^{i-1}-1} k^i_r$, and $q'^1 = 1$, $q'^i = l'_{i-1} + \sum_{r=1}^{q'^{i-1}-1} k^i_r$ for $i = 2, \ldots, n+1$.
We shall use term-variables of the form $\tilde{y}^2_0(t_1, t_2)$ and $\tilde{y}^0_0$ in order to build up "strings of variables or terms". For the sake of simplicity, we shall abbreviate the expression $\tilde{y}^2_0(t_1, \tilde{y}^2_0(t_2, \ldots \tilde{y}^2_0(t_n, \tilde{y}^0_0) \ldots))$ with $\tilde{\boldsymbol{y}}[t_1, \ldots, t_n]$. We let

$$
\begin{aligned}
v^{j,l_1,\ldots,l_j} = \tilde{\boldsymbol{y}}[&\boldsymbol{x}^{1,1}, \ldots, \boldsymbol{x}^{1,l_1-1}, \boldsymbol{x}^{1,l_1+1}, \ldots \boldsymbol{x}^{1,k^1_{s^1}}, \ldots, \\
&\boldsymbol{x}^{j,\underbrace{1,\ldots,1}_{j}}, \ldots, \boldsymbol{x}^{j,l_1,\ldots,l_j-1}, \boldsymbol{x}^{j,l_1,\ldots,l_j+1}, \ldots, \boldsymbol{x}^{j,k^1_{s^1},\ldots,k^j_{s^j}}]
\end{aligned}
$$

where $1 \le j \le n$ and $1 \le l_i \le k^i_{q^i}$ for $i = 1, \ldots, j$. The index sets of the form I^α satisfy $I^\alpha \subseteq I$. The renamings are defined as follows:

$$\vartheta^{n+1,l_1,\ldots,l_{n+1}} = \{\boldsymbol{x}^1/\boldsymbol{x}^{1,l_1}, y^1/\tilde{y}^1(\boldsymbol{x}^{1,l_1}, v^{1,l_1}), \ldots, \boldsymbol{x}^n/\boldsymbol{x}^{n,l_1,\ldots,l_n},$$
$$y^n/\tilde{y}^n(\boldsymbol{x}^{1,l_1}, \ldots, \boldsymbol{x}^{n,l_1,\ldots,l_n}, v^{n,l_1,\ldots,l_n}),\ \boldsymbol{x}^{n+1}/\boldsymbol{x}^{n+1,l_1,\ldots,l_{n+1}}\}$$

where $1 \le l_i \le k^i_{q^i}$ for $i = 1, \ldots, n+1$. Note that $\models C \longrightarrow C^\wedge$.

Let $e_1, \ldots, e_n$ be arbitrary expressions of $\mathcal{L}$ (terms, tuples of terms, or formulae). By $vars(e_1, \ldots, e_n)$ we denote the set of all common (non-term) variables occurring in the expressions; and by $vars^*(e_1, \ldots, e_n)$ we denote the set of all (common and term) variables appearing in the expressions and having at least one occurrence which is not a part ('subterm') of a term-variable from these expressions.
A formula $IC^\wedge$ of $\mathcal{L}$ is said to be an instance of $C^\wedge$ if and only if there exists a substitution ϑ of $\mathcal{L}$, $dom(\vartheta) = vars(C^\wedge)$, such that $IC^\wedge = C^\wedge\vartheta$ where each occurrence of the form $\forall x$, $x \in dom(\vartheta)$, is replaced by the string $\forall z_1 \ldots \forall z_r$, $z_1, \ldots, z_r \in vars^*(\vartheta(x))$. Using the term-variables $\tilde{y}(\ldots)$'s and their 'automatic' renaming at instantiation, we ensure that no $\exists z_1, \ldots, \exists z_r$ are inside the scope of $\forall z_1 \ldots \forall z_r$. So $C^\wedge \models IC^\wedge$.

A.1 Sketched proof of Theorem 2.4

Proof. Let P^S be a set of literal disjunctions of $\mathcal{L}$ obtained from a modified skolemisation of P by omitting all $\wedge$ connectives. Let S be the set of all new term-variables used in this skolemisation. Consider the closure operator $\mathcal{C}_{P^S \cup Eq_{\mathcal{L}\cup S}}$; we add to P^S all the equality axioms of $\mathcal{L} \cup S$:

$$Eq_{\mathcal{L}\cup S} = Eq_{\mathcal{L}} \cup \{\tilde{y}^{n+1}(x_1, \ldots, x_n, v) = \tilde{y}^{n+1}(x'_1, \ldots, x'_n, v') \vee$$
$$x_1 \ne x'_1 \vee \cdots \vee x_n \ne x'_n \,|\, \tilde{y}^{n+1} \in S\}$$

where the $x_1, \ldots, x_n$, v, $x'_1, \ldots, x'_n$, v' are pairwise different variables.
Since $\mathcal{C}_{P^S \cup Eq_{\mathcal{L}\cup S}}$ is continuous, at worst, we reach the least fixpoint at the ωth step of the iteration of the operator from $\emptyset$, denoted as $\mathcal{C}_{P^S \cup Eq_{\mathcal{L}\cup S}} \uparrow \omega$.

At first, consider the case $\Box \in \mathcal{C}_{P^S \cup Eq_{\mathcal{L}\cup S}} \uparrow \omega$. It can be proved that P has no disjunctive model for $\mathcal{L}$, which contradicts the assumption of the statement.

Suppose that $\Box \notin \mathcal{C}_{P^S \cup Eq_{\mathcal{L}\cup S}} \uparrow \omega$. We proceed as follows. We construct a model $\mathrm{M} = (\mathcal{U}_\mathrm{M}, \mathcal{B}_\mathrm{M})$ of P for $\mathcal{L}$ satisfying the condition:

for any prenex conjunctive normal form C' of $\mathcal{L}$, if $\mathrm{M} \models^\vee C'$, then $P \models C'$.

Let $\mathcal{A}_\mathrm{M} = \{f^m_n \,|\, m, n \in \omega\}$ where f^m_n's are new pairwise different m-ary function symbols not occurring in $\mathcal{L}$.
We put $\mathcal{U}_\mathrm{M} = Term_{Func_{\mathcal{L}} \cup \mathcal{A}_\mathrm{M} \cup S}$ - i.e. $\mathcal{U}_\mathrm{M}$ is the set of all terms constructed by symbols in $Func_{\mathcal{L}} \cup \mathcal{A}_\mathrm{M} \cup S$. Since $\mathcal{A}_\mathrm{M}$ contains constants f^0_n's, $\mathcal{U}_\mathrm{M} \ne \emptyset$. A function symbol $f \in Func_{\mathcal{L}}$ is interpreted as the function $f^\mathrm{M} : \mathcal{U}_\mathrm{M}^{ar(f)} \to \mathcal{U}_\mathrm{M}$, $f^\mathrm{M}(u_1, \ldots, u_{ar(f)}) = f(u_1, \ldots, u_{ar(f)})$ where $u_1, \ldots, u_{ar(f)} \in \mathcal{U}_\mathrm{M}$.
We put $\mathcal{B}_\mathrm{M} = \{\overline{b}\,|$

- $\exists d \in \mathcal{C}_{P^S \cup Eq_{\mathcal{L}\cup S}} \uparrow \omega$ such that b, $vars(b) = \emptyset$, is an instance of d in the language $\mathcal{L} \cup \mathcal{A}_\mathrm{M}$;
- $\nexists b'$, $vars(b') = \emptyset$, being an instance of d' in the language $\mathcal{L} \cup \mathcal{A}_\mathrm{M}$ where $d' \in \mathcal{C}_{P^S \cup Eq_{\mathcal{L}\cup S}} \uparrow \omega$ so that $b' \sqsubset b$ (b is a proper superdisjunction of b');
- b is not a tautology$\}$.

It is a simple matter to see that $\mathcal{B}_\mathrm{M}$ does not contain tautologies, the empty disjunction (by the above assumption, no instance may be the empty disjunction), proper superdisjunctions, and it is closed under binary resolution. So the first, second, third, and fourth conditions from the definition of disjunctive model are satisfied.

By the definition of disjunctive model, it can be also seen that $\mathrm{M} \models^\vee P \cup Eq_{\mathcal{L}}$. Thus M is indeed a disjunctive model of P for $\mathcal{L}$.

It remains to prove the main property of M.
Let $C' = \forall x^1_1 \ldots \forall x^1_{m_1} \exists y^1 \ldots \forall x^n_1 \ldots \forall x^n_{m_n} \exists y^n \forall x^{n+1}_1 \ldots \forall x^{n+1}_{m_{n+1}} \bigwedge_{i \in I} D_i$ be a prenex conjunctive normal form of $\mathcal{L}$ where $x^0_1, \ldots, x^0_{m_0}$ are all the free variables in C'; $0 \le n, m_0, \ldots, m_{n+1}$; the x's,

y's are pairwise different variables; and $\bigwedge_{i \in I} D_i$ is a conjunctive normal form. Suppose $\mathrm{M} \models^{\vee} C'$. Consider a variable assignment e of the form $e(x_1^0/f_1^0, \dots, x_{m_0}^0/f_{m_0}^0)$ where $f_i^0 \in \mathcal{A}_{\mathrm{M}}$ (f_i^0's are new pairwise different constants not appearing in C'). By the definition of disjunctive model, there exist $s \geq 1$ and n-ary tuples $T_1', \dots, T_s'$ of terms in $\mathcal{U}_{\mathrm{M}}$ such that for every finite set of triples

$$\overline{D} = \overline{\bigvee_{(y^1,\dots,y^n) \in \{T_1',\dots,T_s'\}} D_{i(y^1,\dots,y^n)} \{x_1^0/f_1^0, \dots, x_{m_0}^0/f_{m_0}^0, x_1^1/f_{m_0+1}^0, \dots, x_{m_1}^1/f_{m_0+m_1}^0, x_1^2/f_1^1(y^1), \dots, x_{m_2}^2/f_{m_2}^1(y^1), \dots, x_1^{n+1}/f_1^n(y^1, \dots, y^n), \dots, x_{m_{n+1}}^{n+1}/f_{m_{n+1}}^n(y^1, \dots, y^n)\}}$$

where $f_b^a \in \mathcal{A}_{\mathrm{M}}$ (f_b^a's are new pairwise different a-ary function symbols not occurring in C') and $i(y^1, \dots, y^n) \in I$, either $\exists b \in \mathcal{B}_{\mathrm{M}}\, b \subseteq \overline{D}$ or $\overline{D}$ contains a pair of contradictory triples. From the definition of $\mathcal{B}_{\mathrm{M}}$, we can deduce that there exist $d_1, \dots, d_{|I|^s}$ not sharing variables mutually and with C' where each d_i is either a variant (of $\mathcal{L}$) of a literal disjunction in $\mathcal{C}_{P^S \cup Eq_{\mathcal{L} \cup S}} \uparrow \omega$ or a tautology of $\mathcal{L}$; and for every literal disjunction D whose $\overline{D}$ is of the form given above, there exists an instance d_i^* of some d_i such that d_i^* subsumes D (D is a superdisjunction of d_i^*), in symbols $d_i^* \sqsubseteq D$. Using the property of most general unifier that an mgu for some expressions contains only variables, function, and predicate symbols appearing in these expressions, we can show there are n-ary tuples $T_1, \dots, T_s$ of terms of $\mathcal{L}$, generalisations of $T_1', \dots, T_s'$, such that for every literal disjunction

$$D' = \bigvee_{(y^1,\dots,y^n) \in \{T_1,\dots,T_s\}} D_{i(y^1,\dots,y^n)} \{x_1^0/z_1^0, \dots, x_{m_0}^0/z_{m_0}^0, x_1^1/z_{m_0+1}^0, \dots, x_{m_1}^1/z_{m_0+m_1}^0, x_1^2/z_1^1(y^1), \dots, x_{m_2}^2/z_{m_2}^1(y^1), \dots, x_1^{n+1}/z_1^n(y^1, \dots, y^n), \dots, x_{m_{n+1}}^{n+1}/z_{m_{n+1}}^n(y^1, \dots, y^n)\}$$

where $i(y^1, \dots, y^n) \in I$; $z_b^a(t_1^i, \dots, t_a^i)$'s are specially denoted pairwise different variables; $(t_1^i, \dots, t_a^i)$ is the prefix of the tuple of terms $T_i = (t_1^i, \dots, t_n^i)$; and the variable $z_b^a(t_1^i, \dots, t_a^i)$ does not appear in the terms $t_1^i, \dots, t_a^i$; there exists an instance d_i' of some d_i satisfying $d_i' \sqsubseteq D'$. Note that d_i' and D' are of $\mathcal{L}$. We may conclude:

$$d_1', \dots, d_{|I|^s}' \models_{Taut} \bigvee_{(y^1,\dots,y^n) \in \{T_1,\dots,T_s\}} \Big(\bigwedge_{i \in I} D_i\Big) \{x_1^0/z_1^0, \dots, x_{m_0}^0/z_{m_0}^0, x_1^1/z_{m_0+1}^0, \dots, x_{m_1}^1/z_{m_0+m_1}^0, x_1^2/z_1^1(y^1), \dots, x_{m_2}^2/z_{m_2}^1(y^1), \dots, x_1^{n+1}/z_1^n(y^1, \dots, y^n), \dots, x_{m_{n+1}}^{n+1}/z_{m_{n+1}}^n(y^1, \dots, y^n)\} \qquad (1)^*$$

We know that $d_1, \dots, d_{|I|^s}$ are either variants (of $\mathcal{L}$) of literal disjunctions from $\mathcal{C}_{P^S \cup Eq_{\mathcal{L} \cup S}} \uparrow \omega$ or tautologies of $\mathcal{L}$ and $d_1', \dots, d_{|I|^s}'$ are instances (of $\mathcal{L}$) of them. Using the property similar to the lifting lemma that an instance In of a resolvent of two literal disjunctions is a tautological consequence of instances In_1, In_2 of these two literal disjunctions, we can deduce there exist instances $if_1^{\wedge}, \dots, if_r^{\wedge}$ (of $\mathcal{L}$) of some literal disjunctions $ld_1, \dots, ld_r$ in $P^S \cup Eq_{\mathcal{L} \cup S}$ such that $if_1^{\wedge}, \dots, if_r^{\wedge} \models_{Taut} d_1', \dots, d_{|I|^s}'$. So, $if_1^{\wedge}, \dots, if_r^{\wedge} \models_{Taut} (1)^*$. Moreover, it can be proved that there exist

- formulae $F_1, \dots, F_k$ ($k \geq 0$), some renamed and universally closed prenex conjunctive normal forms of $P \cup Eq_{\mathcal{L}}$ not sharing variables mutually; note that $P \models \bigwedge_{i=1,\dots,k} F_i$;
- a prenex conjunctive normal form F obtained from $F_1, \dots, F_k$ by the concatenation of their preneces and the conjunction of their matrices:

$$F = \mathcal{P}renex_1 \dots \mathcal{P}renex_k \bigwedge_{i=1,\dots,k} CNF_i$$

  where $\mathcal{P}renex_i$ is the prenex of F_i and CNF_i is the matrix of F_i; note that $\models \bigwedge_{i=1,\dots,k} F_i \longrightarrow F$;
- and an $\wedge$-expansion $F^{\wedge}$ of F and its instance $IF^{\wedge}$ of the form:

$$IF^{\wedge} = \mathcal{P}renex \bigwedge_{i=1,\dots,r} if_i^{\wedge};$$

  note that $\models F \longrightarrow F^{\wedge}$ and $F^{\wedge} \models IF^{\wedge}$.

We may conclude: $P \models \bigwedge_{i=1,\dots,k} F_i \models F \models F^\wedge \models IF^\wedge = \mathcal{P}renex \bigwedge_{i=1,\dots,r} if_i^\wedge \models \mathcal{P}renex\,(1)^* \models \forall x_1^1 \dots \forall x_{m_1}^1\, \exists y^1 \dots \forall x_1^n \dots \forall x_{m_n}^n\, \exists y^n\, \forall x_1^{n+1} \dots \forall x_{m_{n+1}}^{n+1} \bigwedge_{i\in I} D_i = C'$. So, the main property of M is proved; hence, M is a characteristic disjunctive model of P for $\mathcal{L}$.

In the end, we show an immediate generalisation to arbitrary formulae in prenex form. Consider C of the form:

$$\forall x_1^1 \dots \forall x_{m_1}^1\, \exists y^1 \dots \forall x_1^n \dots \forall x_{m_n}^n\, \exists y^n\, \forall x_1^{n+1} \dots \forall x_{m_{n+1}}^{n+1}\, B$$

where B is a quantifier-free formula such that $\bigwedge_{i\in I} D_i$ is one of its conjunctive normal forms. Then $\bigwedge_{i\in I} D_i \models_{Taut} B$ and also $\bigwedge_{i=1,\dots,r} if_i^\wedge \models_{Taut} (1)^* \models_{Taut} (1)$ what is the disjunction obtained from $(1)^*$ by replacing $\bigwedge_{i\in I} D_i$ with B. So, $P \models \bigwedge_{i=1,\dots,k} F_i \models F \models F^\wedge \models IF^\wedge = \mathcal{P}renex \bigwedge_{i=1,\dots,r} if_i^\wedge \models \mathcal{P}renex\,(1)^* \models \mathcal{P}renex\,(1) \models \forall x_1^1 \dots \forall x_{m_1}^1\, \exists y^1 \dots \forall x_1^n \dots \forall x_{m_n}^n\, \exists y^n\, \forall x_1^{n+1} \dots \forall x_{m_{n+1}}^{n+1}\, B = C$.

The proof is completed.

∎

References

[1] Buss, S. R. On Herbrand's Theorem. *Technical report, University of California, San Diego, U.S.A.*, 1995.

[2] Buss, S. R. An introduction to proof theory. *Typeset manuscript, University of California, San Diego, U.S.A.*, 1995.

[3] Goldfarb, W. D. Herbrand's Theorem and the incompleteness of arithmetic. *A Jerusalem Philosophical Quarterly, 39, Iyyun, Israel*, 45-64, 1990.

[4] Goldfarb, W. D. Herbrand's error and Gödel's correction. *Modern Logic, 3*, 103-118, 1993.

[5] Guller, D. Semantics for disjunctive programs. *Proceedings of the Fifth Workshop of Logic, Language, Information and Computation*, 1998. *Abstract in Logic Journal of the IGPL*, Oxford University Press, Vol. 6, N. 6, November 1998.

[6] Guller, D. On clausal models. *Proceedings of the Sixth Workshop of Logic, Language, Information and Computation*, 1999. *Abstract in Logic Journal of the IGPL*, Oxford University Press, Vol. 7, N. 4, July 1999.

[7] Guller, D. One generalisation of Herbrand's theorem. *Proceedings of the Second Panhellenic Logic Symposium*, 1999.

[8] Minker, J. and Ruiz, C. Semantics for disjunctive logic programs with explicit and default negation. *Fundamenta Informaticae*, 20(3/4):145-192, 1994.

[9] Shoenfield, J. R. Mathematical Logic. *Addison-Wesley*, 1967.

[10] Takeuti, G. Proof Theory. *North-Holland, Amsterdam, 2nd ed.*, 1987.

[11] Voronkov, A. Herbrand's theorem, automated reasoning and semantic tableaux. *Technical Report 151, Uppsala University, Computing Science Department*, 1998.

[12] Voronkov, A. Simultaneous rigid E-unification and other decision problems related to the Herbrand theorem. *Technical Report 152, Uppsala University, Computing Science Department*, 1998.

Lambek Grammars as Combinatory Categorial Grammars

GERHARD JÄGER, *Zentrum für Allgemeine Sprachwissenschaft (ZAS), Jägerstr. 10/11, 10117 Berlin, Germany.*
E-mail: jaeger@zas.gwz-berlin.de

Abstract

We propose a combinatory reformulation of the product free version of the categorial calculus **LL**, i.e. the associative Lambek calculus that admits empty premises. We prove equivalence of the combinatory with the standard Natural Deduction presentation of **LL**. The result offers a new perspective on the relation between the type logical and the combinatory branch of the Categorial Grammar research program.

Keywords: Categorial Grammars, Lambek Calculus, CCG

1 The Lambek calculus

1.1 Sequent presentation

In his seminal paper [7], Joachim Lambek introduced the (associative) Lambek calculus, a type logical extension of Bar Hillel's [3] Basic Categorial Grammars. Lambek gives two equivalent proof theories for his calculus, an axiomatic one and a Gentzen style sequent calculus. We will restrict attention to the product-free fragment of this calculus. There the categorial slashes "/" and "\" are the only logical constants. The sequent calculus consists of the rules of use and of proof for them, together with the identity axiom scheme for arbitrary types. Being a subsystem of positive implicational intuitionistic logic, the history of a proof in the Lambek calculus can be recorded in λ-terms, using the Curry-Howard correspondence.

DEFINITION 1.1 (**LL**: Sequent presentation)

$$\frac{}{x : A \to x : A}\,id$$

$$\frac{X \to M : A \qquad Y, x : B, Z \to N : C}{Y, y : B/A, X, Z \to N[x \leftarrow (yM)] : C}\,/L \qquad \frac{X, x : A \to M : B}{X \to \lambda x.M : B/A}\,/R$$

$$\frac{X \to M : A \qquad Y, x : B, Z \to N : C}{Y, X, y : A \backslash B, Z \to N[x \leftarrow (yM)] : C}\,\backslash L \qquad \frac{x : A, X \to M : B}{X \to \lambda x.M : A \backslash B}\,\backslash R$$

Lambek originally added the additional constraint that the premises of each sequent be non-empty. We omit this requirement as an essentially superfluous complication, thus arriving at the product free version of the calculus **LL**.

In [7] it is proved that the Cut rule is admissible in the Lambek calculus. The proof carries over to the version of **LL** considered here without complications.

1.2 Natural Deduction

Alternatively, **LL** can be presented in a Natural Deduction format, i.e. the rules of use for the implicational slashes can be replaced by explicit Modus Ponens rules. Here Curry-Howard labels mirror the proof history directly.

DEFINITION 1.2 (**LL**: Natural Deduction)

$$\frac{}{x : A \to x : A}\, id$$

$$\frac{X \to M : B/A \qquad Y \to N : A}{X, Y \to (MN) : B}\, /E \qquad \frac{X, x : A \to M : B}{X \to \lambda x.M : B/A}\, /I$$

$$\frac{X \to M : A \qquad Y \to N : A \setminus B}{X, Y \to (NM) : B}\, \setminus E \qquad \frac{x : A, X \to M : B}{X \to \lambda x.M : A \setminus B}\, \setminus I$$

It is easy to show that the two inference formats are equivalent. Even stronger, sequent derivations and ND derivations generate the same set of proof terms.

2 Combinatory Categorial systems

The Combinatory branch of Categorial grammar was initiated by the work of [1]. As in Basic Categorial Grammars, CCG (Combinatory Categorial Grammar) makes use of the identity axiom and the slash elimination rules [/E] and [\E], but it does without the slash introduction rules [/I] and [\I]. The Basic Categorial core is extended by other schemes of inference though. Most works in the CCG tradition assume some—possibly restricted—version of Type Raising **T** and (generalized) Function Compositions **B** (for ease of comparison, we employ a sequent style notation here). As in the systems discussed above, the history of a derivation can be recorded in a label in CCG; labels are terms in the closure of the set of typed variables under the set of combinators.

DEFINITION 2.1 (Type Raising)

$$\frac{X \to M : A}{X \to \mathbf{T}_{>}(M) : B/(A \setminus B)}\, \mathbf{T}_{>} \qquad \frac{X \to M : A}{X \to \mathbf{T}_{<}(M) : (B/A) \setminus B}\, \mathbf{T}_{<}$$

DEFINITION 2.2 (Function Composition)

$$\frac{X \to M : A/B \qquad Y \to N : C_{n_l} \setminus \cdots \setminus B/D_1/\cdots/D_{n_r}}{XY \to \mathbf{B}_{>}(M, N) : C_{n_l} \setminus \cdots \setminus A/D_1/\cdots/D_{n_r}}\, \mathbf{B}_{>}$$

$$\frac{X \to N : C_{n_l} \setminus \cdots \setminus B / D_1 / \cdots / D_{n_r} \qquad M : B \setminus A \to Y}{XY \to \mathbf{B}_<(N, M) : C_{n_l} \setminus \cdots \setminus A / D_1 / \cdots / D_{n_r}} \mathbf{B}_<$$

It is easy to see that the well-known schemes of ordinary function composition

$$A/B, B/C \to A/C$$

and crossed composition

$$A/B, C \setminus B \to C \setminus A$$

(and their mirror images) are special cases of the more general operations given above.

Next to these combinatory inference schemes, CCGs might contain empty categories, i.e. axioms of the form

$$\frac{}{\to M : A}$$

for some term M and type A. They may be considered as 0-place combinators.

3 Combinatory presentation of **LL**

Even though both branches of modern Categorial grammar are based on the same basic intuitions, the formal properties of the the systems differ considerably. This is most obvious when we consider issues of weak generative capacity: Since [9] it is known that Lambek grammar recognize exactly the context free languages (and the proof carries over directly to **LL**), while the generative capacity of CCGs is mildly context sensitive (cf. [5]). So the two approaches seem to be irreconcilable.

Recent developments in type logical grammar have changed this rigid picture. The use of multi-modality (see for instance [8]) increases the generative capacity of type logical grammars in principle to Turing completeness (cf. [4]), so there is no principal obstacle to simulate CCGs in the type logical framework. [6] contains some general results based on this idea.

In this section, we will pursue the opposite strategy. It will be shown that under a slight generalization of the notion of function composition as it is used in the CCG tradition, **LL** can equivalently be reformulated as a combinatory system.

First some terminology: we will call the first operand of forward function composition $\mathbf{B}_>$ and the second operand of backward composition $\mathbf{B}_<$ ("M" in the schemes above) the *functor* and the other operand ("N") the *argument* of the composition operation. In the schemes given above, the argument is always matched with an outermost argument place of of the functor (the rightmost one in the case of $\mathbf{B}_>$ and the leftmost one in the case of $\mathbf{B}_<$). The notion of function composition to be employed here generalizes this aspect: $\mathbf{B}_>$ can target any forward looking (and $\mathbf{B}_<$ any backward looking) argument slot of the functor. To keep track of the argument slot to be addressed, we use a superscript notation. Generalized Function Composition is subject to certain constraints; certain instances are only licit if one or both of the operands are closed, i.e. do not contain free variables.

DEFINITION 3.1 (Generalized Function Composition)

$$\frac{X \to M : A/B_1/\cdots/B_i \qquad Y \to N : C_{n_l} \setminus \cdots \setminus C_1 \setminus B_1/D_1/\cdots/D_{n_r}}{XY \to \mathbf{B}^i_>(M, N) : C_{n_l} \setminus \cdots \setminus C_1 \setminus A/D_1/\cdots/D_{n_r}/B_2/\cdots/B_i}\ \mathbf{B}^i_>$$

$i = 1$ or $n_l = 0$.
If $i > 1$, then N is closed.
If $n_l > 0$ then M is closed.

$$\frac{X \to N : C_{n_l} \setminus \cdots \setminus C_1 \setminus B_1/D_1/\cdots/D_{n_r} \qquad Y \to M : B_i \setminus \cdots \setminus B_1 \setminus A}{XY \to \mathbf{B}^i_<(N, M) : B_i \setminus \cdots \setminus B_2 \setminus C_{n_l} \setminus \cdots \setminus C_1 \setminus A/D_1/\cdots/D_{n_r}}\ \mathbf{B}^i_<$$

$i = 1$ or $n_r = 0$.
If $i > 1$, then N is closed.
If $n_r > 0$ then M is closed.

Furthermore we assume any type instance of the identity combinator $\mathbf{I}$:

DEFINITION 3.2 (Identity Combinator)

$$\frac{}{\to \mathbf{I}^A_> : A/A}\ \mathbf{I}^A_>$$

$$\frac{}{\to \mathbf{I}^A_< : A \setminus A}\ \mathbf{I}^A_<$$

Finally, an appropriate combinatory reformulation of $\mathbf{LL}$ would require generalizations of the "swapping" rule $A\setminus(B/C) \Leftrightarrow (A\setminus B)/C$. Since this rule is reversible, it induces an equivalence relation on types. We thus suppress the swapping rule an instead tacitly consider each type in a derivation as a representative of its equivalence class induced by swapping.

Let us say that a sequent $X \to A$ is derivable in $\mathbf{LL}_C$ iff it is derivable by using only the identity axiom $[id]$ and the combinatory schemes $\mathbf{I}^A_>, \mathbf{I}^A_<, \mathbf{B}^i_>$, and $\mathbf{B}^i_<$ for arbitrary types A and natural numbers i.

THEOREM 3.3
A sequent $X \to A$ is derivable in $\mathbf{LL}_C$ if and only if it is derivable in $\mathbf{LL}$.

Proof: Let us start with introducing some syntactic sugar. Following [10], we write $\lambda^l x.M$ in Curry-Howard terms rather than $\lambda x.M$ if the main functor of the type of the whole term is "$\setminus$"; otherwise we write $\lambda^r x.M$. Furthermore, if M has type $B \setminus A$ and N has type B, we write (NM) rather than (MN). If we talk about the rightmost or leftmost free variable occurrence in a term, we assume the linearization induced by this modified notation. This notation has the obvious advantage that the sequence of free variables in a derivable proof term directly mirrors the sequence of premises of the corresponding proof (a feature that is shared by the combinatory system).

Wansing ([10], p. 70) proves the following lemma.

LEMMA 3.4
A term M is a proof term of an $\mathbf{LL}$-proof iff

1. Every λ binds exactly one variable occurrence.
2. For every subterm $\lambda^l x.N$ of M, x is the leftmost free variable occurrence in N.

3. For every subterm $\lambda^r x.N$ of M, x is the rightmost free variable occurrence in N.

Now we can start with the proof of the *if*-direction. Let a labeled $\mathbf{LL}_C$-proof be given. We define a translation from $\mathbf{LL}_C$ proof terms to Curry-Howard terms of the ND calculus.

DEFINITION 3.5 (Translation $\mathbf{LL}_C \to$ ND)

1. $(\mathbf{I}^A_>)^\# = \lambda^r x.x : A/A$
2. $(\mathbf{B}^i_>(M,N))^\# =$
   $\lambda^r y_i \dots y_2 \lambda^l x_{n_l} \dots x_1 \lambda^r z_{n_r} \dots z_1.(M^\# y_i \dots y_2(x_1 \dots x_{n_l} N^\# z_{n_r} \dots z_1))$
3. $(\mathbf{I}^A_<)^\# = \lambda^l x.x : A \setminus A$
4. $(\mathbf{B}^i_<(N,M))^\# =$
   $\lambda^l y_i \dots \lambda^l y_2 \lambda^r z_{n_r} \dots z_1 \lambda^l x_{n_l} \dots x_1.((x_1 \dots x_{n_l} N^\# z_{n_r} \dots z_1) y_2 \dots y_i M^\#)$
   where the types of M, N are as above and the types of x_k, y_k, and z_k are C_k, B_k, and D_k respectively.

It is easy to see that the translation of every well-formed combinatory proof term is a derivable ND term, so every derivation in $\mathbf{LL}_C$ can be transformed into a Natural Deduction proof.

As a consequence of this, adding the combinatory rules to the ND calculus does not extend the notion of derivability of the calculus. We can thus work with a hybrid calculus that makes use both of ND and of combinatory rules without transcending the bounds of $\mathbf{LL}$. At the level of terms, we work with a hybrid system as well; the new term language contains application, abstraction, and the combinators from $\mathbf{LL}_C$. It is easy to see that a term of the hybrid term language is a derivable proof term if and only if it obeys both the constraints on the applicability of the combinators and Wansing's constraints on abstraction.

As next step we define a reduction relation on this hybrid term language.

DEFINITION 3.6 (Reduction)

$$
\begin{aligned}
(M : A/B, N : B) &\rightsquigarrow \mathbf{B}^1_>(M,N) \\
(N : B, M : B \setminus A) &\rightsquigarrow \mathbf{B}^1_<(N,M) \\
\lambda^l x.x &\rightsquigarrow \mathbf{I}_< \\
\lambda^l x.\mathbf{B}^i_>(M[x],N) &\rightsquigarrow \mathbf{B}^i_>(\lambda^l x.M,N) \\
\lambda^l x.\mathbf{B}^i_>(M,N[x]) &\rightsquigarrow \mathbf{B}^i_>(M,\lambda^l x.N) \\
\lambda^l x.\mathbf{B}^i_<(N[x],M) &\rightsquigarrow \mathbf{B}^i_<(\lambda^l x.N,M) \\
\lambda^l x.\mathbf{B}^i_<(N,M[x]) &\rightsquigarrow \mathbf{B}^{i+1}_<(N,\lambda^l x.M) \\
\lambda^r x.x &\rightsquigarrow \mathbf{I}_> \\
\lambda^r x.\mathbf{B}^i_>(M[x],N) &\rightsquigarrow \mathbf{B}^{i+1}_>(\lambda^r x.M,N) \\
\lambda^r x.\mathbf{B}^i_>(M,N[x]) &\rightsquigarrow \mathbf{B}^i_>(M,\lambda^r x.N) \\
\lambda^r x.\mathbf{B}^i_<(N[x],M) &\rightsquigarrow \mathbf{B}^i_<(\lambda^r x.N,M) \\
\lambda^r x.\mathbf{B}^i_<(N,M[x]) &\rightsquigarrow \mathbf{B}^i_<(N,\lambda^r x.M)
\end{aligned}
$$

The reduction relation is generalized to subterms in the obvious way:
if $M \leadsto M'$, then $N[M/x] \leadsto N[M'/x]$.

Note that the output of each reduction step is a well-formed proof term provided the input is well-formed. So term reduction does not lead beyond the bounds of the hybrid combinatory/ND system. Furthermore, neither the sequence of free variables of a proof term nor the type of the term itself is affected by any reduction step. So input and output of a reduction step represent proofs of the same sequent. Note furthermore that any occurrence of application in a well-formed proof term matches the left hand side of a reduction step. The same holds for λ-abstraction, with the exception of configurations where a λ-abstraction takes immediate scope over another abstraction.

Given this, it is straightforward to show that every well-formed proof term of the hybrid system can be normalized to a pure combinatory term. A possible algorithm starts with replacing all instances of application by **B**. In the next step, the leftmost λ-operator that has no other λ in its scope is targeted by a reduction step (as pointed out in the last paragraph, this is always possible). Every λ-reduction step either reduces the number of combinators in the scope of the targeted λ-operator, or it removes it altogether. So after finitely many reduction steps, the number of λ-operators is reduced by one. Repeating this procedure eventually leads to a normal form term that neither contains λs nor applications, i.e. a pure combinatory term. So starting with an arbitrary ND-proof of a sequent $X \to A$, by normalizing its Curry-Howard term we end up with a $\mathbf{LL}_C$-proof term and thus an $\mathbf{LL}_C$-proof of this very sequent. $\dashv$

4 Conclusion

The present paper presented a novel Combinatory Categorial system making use of the identity combinator, a generalized version of function composition and—implicitly—swapping. As main result, it was proved that under the appropriate restriction of generalized function composition, this combinatory system is equivalent to the product free version of the calculus **LL**, the variant of the associative Lambek calculus that does without the restriction to empty premises.

At the present point, the significance of the result lies in the fact that is makes a connection between the two main branches of Categorial Grammar; it is thus a base for transferring concrete analyses from the Lambek tradition into the Combinatory tradition. Two directions for further research suggest themselves: Are there similar constructions for other type logical calculi like full **LL**, **L**, the non-associative Lambek calculus **NL** etc.? And, does the combinatory reformulation offer new insights about the Lambek calculus as such? Especially investigations of the proof theoretic complexity of type logical grammars might profit from the combinatory perspective.

References

[1] Anthony E. Ades and Mark J. Steedman. On the order of words. *Linguistics and Philosophy*, 4:517–558, 1982.

[2] Kazimierz Ajdukiewicz. Die syntaktische Konnexität. *Studia Philosophica*, 1:1–27, 1935.

[3] Yehoshua Bar-Hillel. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58, 1953.

[4] Bob Carpenter. The Turing-completeness of multimodal Categorial Grammars. unpublished ms., 1995.

[5] Aravind Joshi, K. Vijay-Shanker, and David Weir. The convergence of mildly context-sensitive grammar formalisms. In Tom Wasow and Peter Sells, editors, *Processing of Linguistic Structure*, pages 31–81. MIT Press, Cambridge (Mass.), 1990.

[6] Geert-Jan Kruijff and Jason Baldridge. Relating Categorial Type Logics and CCG through simulation. ms., University of Edinburgh, 2000.

[7] Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958.

[8] Michael Moortgat. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 2, pages 93–178. Elesvier, MIT Press, 1997.

[9] Martin Pentus. Lambek grammars are context-free. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*. Montreal, 1993.

[10] Heinrich Theodor Wansing. *The Logic of Information Structure*. PhD thesis, Freie Universität Berlin, 1992.

Judging Unresolved Utterances in Context

PETER KRAUSE, *Institute for Computational Linguistics, Azenbergstrasse 12, 70174 Stuttgart, Germany.*
E-mail: peter@ims.uni-stuttgart.de

Abstract

A theory of judging unresolved utterances in dialogue is developed in discourse representation theory using techniques from belief revision and applied to a classical problem for the presuppositional theory of definite descriptions.[1]

1 Introduction

In this paper, a step toward a formalized solution of a basic problem from the debate on truth value gaps induced by presuppositions is taken. [9] had claimed that sentences like

Example 1.1
The king of France is wise.

are neither true nor false, but rather truth-valueless. The problem for this theory that will be in the focus of this paper is a certain class of counterexamples against the original proposal of [9]. The following example has been given by [7].

Example 1.2
The king of France is sitting in that chair.

This sentence should be considered in the following context. The speaker is pointing to a chair which is, for both the speaker and the hearer, visibly empty. In this case, most informants judge the utterance false, and the reaction of calling it undefined or lacking a truth-value does not come to mind easily. Nevertheless, the definite description in such an example remains presuppositional, as the projection behaviour shows.

Example 1.3
Whenever the king of France is sitting in that chair, the chair to its right remains empty.

The existence of a king of France can be inferred from this sentence. This would not be expected if the presupposition that is normally attributed to definite description disappeared due to the predicate. What is the characteristic property of contexts in which a sentence containing an empty definite description is judged false ? [7] proposes the following analysis: a sentence containing a definite description that causes presupposition failure is false instead of indeterminate if the context supports

[1] Short version

its falsity even after it has been revised with the statement of unique existence that is presupposed by the definite description. As [11] points out, this account has to be refined in two ways. If we assume that the chair in question in example 1.2 is in fact the throne of the French monarch, and that at the time of utterance the head of state would in any case be in the place from which he or she is governing (the president's office in a republic, the throne in a monarchy), then the sentence would be predicted to be true rather than false. However, if the chair is visibly empty, the sentence would be judged false in such a situation. This can be accounted for by excluding perceptually given information from the revision process on which Lasersohn's account is based. The second problem is that there are examples that are judged false on another basis[2]:

EXAMPLE 1.4
The king of France is someone's subject.

The king of France cannot possibly be a woman (the expression *the queen of France* would have to be used instead) even if the context is revised to allow for his existence. This phenomenon can be accounted for by excluding general laws such as the meaning postulate that kings are male from revision. In this paper, I will start from the assumption that the empirical analysis of Lasersohn as refined by [11] is correct and give a technical account of it in terms of an independently motivated theory of judging utterances in dialogue. The theory is based on discourse representation theory (DRT) and dynamic semantics, presupposition justification, and a simple method for belief revision under integrity constraints. DRT is needed to account for the proper interpretation of definite noun phrases. The presupposition justification module is used to detect the cases in which a revision of the context is called for, and to bridge the gap between an unresolved sentence and one that can be judged by standard logical notions. The use of integrity constraints in belief revision accounts for the observations about the role of non-defeasible information in judging the examples after the context has been adjusted. The theory of judging utterances is applied in the formulation of an algorithm to judge unresolved sentences in context.

2 Expressing integrity constraints in DRT

In order to formulate an operation of belief contraction (and revision) that respects integrity constraints, the different kinds of information must be distinguishable from each other. In principle, there are two possibilities for achieving this. One option is to let different DRSs represent the information that is perceptually accessible information from the speech situation, the general laws and the defeasible information. Then a method for integrating and crossreferencing these different knowledge bases is needed. Secondly, and I will follow this option here because it seems technically simpler, the entire information can be represented in one DRS, but each discourse referent introduction and each condition on the top level of the DRS is prefixed by a label indicating what kind of knowledge the DRS-component expresses. Some simple functions are needed to handle the labelling. Meaningful DRSs are referentially coherent. Belief revision operations could easily destroy important anaphoric links and make the result uninterpretable if no special care is taken to avoid this problem. For instance, it must be avoided that an antecedent is removed by contraction without the

[2] the example is similar to one mentioned by von Fintel

conditions in which the discourse referent occurs free being removed as well. These are the problems that will be addressed in this section.

2.1 Basic DRT notions

The following syntactic notions will be used[3]: Labelled DRSs K with free and active variables $(fv(K), av(K))$ containing demonstrative descriptions $\delta x : K$ and definite descriptions $\iota x : K$. The top-level components are labelled with S_i (indefeasible situational knowledge), R_i (indefeasible rules), and D_i (defeasible information). The top-level components $tlc(K)$ of a DRS K are the conjunction-minimal non-embedded discourse referent introductions and conditions in K. A DRS K_1 is a proper part of another DRS K_2, written $K_1 \sqsubset_{DR} K_2$, if K_1 is the dynamic conjunction of all the top-level components of K_2 in the same order. As usual, $K' \sqsubseteq_{DR} K$ iff $K' \sqsubset_{DR} K$ or $K' = K$. The notion of test is defined as in dynamic predicate logic. An equation series is a dynamic conjunction of only equations. A DRS is called simple if it consists of atomic conditions and equations only. DRSs without definite or demonstrative descriptions are called description-free. A DRS K satisfies anaphoric integrity, $ai(K)$, if K has no free variables. The functions Sl and Rl return the situation and rule labels of a DRS, respectively. The integrity constraint $IC(K)$ of a DRS K is the maximal part of it such that $ai(IC(K))$ and all situation and rule labels are preserved.
As for semantics, an update semantics using partial variable assignments will be assumed, see [3]. Let σ_0 be the initial information state.
A DRS K_1 entails a DRS K_2, $K_1 \models K_2$, if (assuming $ai(K_1)$ and $ai(K_2)$) the corresponding predicate logical formulas entail each other. $K_1 \vdash K_2$ will be used for derivability in a DRS calculus. Then $K_1 \vdash_{update} K_2$ iff

- $ai(K_1; K_2)$
- $K_1 \vdash K_1; K_2$

I will write $K \vdash_{update} def(x)$ iff $x \in av(K)$.

3 Justification of context-dependent expressions

Although the theory of [9] has been confronted with many apparent counterexamples (not least the ones discussed here), a treatment of definite noun phrases in terms of presuppositions is commonly assumed in linguistics. I will here build on the anaphoric account of definite descriptions of [5]. In this theory and in particular in the in some respects similar DRT-account [10], a definite description gives rise to a process of presupposition justification, which can either result in resolution, in accommodation or in a failure to resolve the presupposition. By the way, the account of [7] is not formulated in terms of an anaphoric account of presupposition, but rather using non-dynamic predicate logical formulas. In the account developed below, only the assumption that in case of failure of the existence presupposition, the context will be revised will be taken over, not the treatment of presuppositions itself. The importance of the process of presupposition resolution for the problem described in the introduction lies not only in the fact that a presupposition failure triggers a revision

[3] Explicit definitions can be found in the long version of the paper. The notions are minor variants of standard ones.

process, but also in the process following the contraction of the context. The presupposition must be justified with respect to the new context. Whether the contraction with the negation of a presupposition that will be described below is actually part of the process of accommodation or a preparation for it is mainly a terminological issue. I decided to use the latter option. Because this paper is about the problem of presupposition failure and not about the problem of presupposition projection, only global binding and accommodation (using the terminology of [10]) will be considered. Local and intermediate accommodation will not play any role[4].

According to the theory of [10], there are basically two options for resolving definite descriptions: binding and accommodation. Both may be combined in cases of so-called partial match. In [10], the possibility of revision with a presupposition is not considered. This is the third basic case that is important for our analysis here. An additional complication comes from the fact that sometimes presuppositions may be bound in more than one way. This can occur also when some information has to be accommodated. For instance, a single definite description may be bound to two different antecedents after the descriptive content has been accommodated about either one. Another case would be a sentence with two definite descriptions, one giving rise to accommodation, the other to multiple bindings.

The following assumption will be made to avoid accidental identities between variables.

Convention 3.1 (Variable choice in translations)
When a sentence is formalized in context K, new variables (i.e. variables that do not occur in K) are used for indefinite, definite and quantificational expresssions.

The following notion is used in the definition of admittance below. The purpose is to recursively collect the presupposed *conditions* of definite and demonstrative descriptions occurring in an atomic condition.

Definition 3.2 (Elementary presuppositions)
Let K be an atomic condition with possibly recursive definite description arguments (where the restrictors of the definite descriptions again are atomic conditions) DRS. Then $ep(K)$, the DRS representing the set of elementary presuppositions of K, is given by the recipe:

- $ep(\iota x : K) \; = \; epaux(K)$
- $ep(\delta x : K) \; = \; epaux(K)$
- $epaux(R(t_1^{x_1}, \ldots t_n^{x_n})) \; = \; ep(t_1^{x_1}); \ldots \; ep(t_n^{x_n}); R(x_1, \ldots \; x_n)$
- $epaux(K_1; K_2) \; = \; ep(K_1); ep(K_2)$

When is each presupposition in a sentence satisfied if it is evaluated with respect to a given context K ? The definition of admittance, which is modelled after [6] (see also [2], whose notation I adopt here) provides an answer to this question with respect to the DRT-language used here.

[4] Of course it would be interesting to extend the analysis to cover embedded occurrences of presupposition triggers and their effects on perceived presupposition failure. However, for such sentences the empirical analysis is complicated by the fact that there are usually other options besides belief revision to avoid presupposition failure, such as local accommodation.

DEFINITION 3.3 (Admittance)
A simple DRS K^+ is defined with respect to a context DRS K, or K^+ is admitted by K, written $K \rhd K^+$ as determined by the following recursion.

- $K \rhd \bot$
- $K \rhd R^n t_1 \ldots t_n$ iff, for each $t_i \in \{t_1 \ldots t_n\}$
  - for each $x \in fv(t_i)$:
    $K \vdash def(x)$ and
  - $K \vdash_{update} ep(t_i)$
- $K \rhd K_1; K_2$ iff $K \rhd K_1$ and $K; K_1 \rhd K_2$.
- $K \rhd t_1 = t_2$ analoguously to $R^n t_1 \ldots t_n$.
- $K \rhd \overline{x}$.
- $K \rhd \neg K_1$ iff $K \rhd K_1$.
- $K \rhd K_1 \Rightarrow K_2$ iff $K \rhd K_1$ and $K; K_1 \rhd K_2$.
- $K \rhd K_1 \vee K_2$ iff $K \rhd K_1$ and $K \rhd K_2$.

The minimal presuppositions that a sentence places on a context are given by the following definition.

DEFINITION 3.4 (Minimal presuppositions in terms of admittance)
Let K^+ be a DRS. A DRS K with $ai(K)$ represents the minimal presuppositions of K^+, written $mp(K^+) = K$, if $K \rhd K^+$ and there is no K' with $ai(K')$ and $K' \sqsubseteq_{DR} K$ for which $K' \rhd K^+$.

As long as the presuppositions are not satisfied by their local context, they can be computed in a cumulative manner:

DEFINITION 3.5 (Minimal justifying context)
Let K^+ be a DRS containing anaphoric expressions. Then K^{min} with $ai(K^{min})$ is called a minimal justifying context for K^+ if

- every anaphoric expression in K^+ is defined in its local context if K^+ is placed in the context of K^{min}.
- There is no K^{sub} with $K^{sub} \sqsubseteq_{DR} K^{min}$ with respect to which K^+ is already defined.

Now we can turn to the basic concepts of presupposition justification. First the notions of resolution and

DEFINITION 3.6 (Resolution)
Let K be a DRS and let K^+ be an update DRS such that $K; K^+$ does not satisfy anaphoric integrity. A *resolution* of K^+ with respect to the context K is a minimal equation series K^{res} such that $ai(K; K^{res}; K^+)$ and $K; K^{res} \rhd K^+$.
Let $Res(K, K^+)$ denote the set of such resolutions K^{res}. K^+ is called resolvable with respect to K if $Res(K, K^+)$ is non-empty. It is called uniquely resolvable if $Res(K, K^+)$ has exactly one element.

DEFINITION 3.7 (Accommodation)
A DRS K^+ with elementary presupposition $ep(K^+)$ is called resolvable by accommodation with respect to a DRS K if there is a $\subseteq_{DR}$-minimal K^{acc} with $ai(K; K^+)$ and $K \rhd K^{acc}$ such that $Res(K; K^{acc}, K^+)$ is non-empty.

Let K^{accres} be in $AccRes(K, K^+)$, the set of accommodations, if $K^{accres} = K^{acc}; K^{Res}$, where K^{acc} is as above and $K^{Res} \in Res(K; K^{acc}, K^+)$.

Definition 3.8 (Justifications)
Let K^+ be a DRS with elementary presuppositions $ep(K^+)$ and K be a context DRS satisfying anaphoric integrity.
The set of justifications of K^+ with respect to K, $Just(K, K^+)$, is determined as follows:

$$\begin{aligned} \text{If } Res(K, K^+) \neq \emptyset, \text{ then } Just &= Res(K, K^+) \\ \text{If } Res(K, K^+) = \emptyset, \text{ then } Just &= AccRes(K, K^+) \end{aligned}$$

4 DRS base revision under integrity constraints

In this section, the elements of a method for belief revision usable with DRSs are described. The method takes into account integrity constraints and preserves anaphoric integrity. First some basic notions are defined. Then the properties a DRS contraction operation suitable for treating the problems described in the introduction are formulated. Finally, a concrete operation of guarded maxichoice contraction is formulated. The standard AGM method of belief revision [1] uses so-called belief sets, i.e. deductively closed theories as a basis for belief revision. Of course it is possible to form the logical closure of a DRS. However, certain computational intuitions and the important property that implicit entities are not available for unrestricted anaphoric reference would be lost under such an approach. This is why discourse representation structures correspond to belief bases rather than to belief sets. In addition, an adequate treatment of integrity constraints has to be given.
The problem is to contract a labelled DRS K in such a way with a DRS K^{new} that in the result, $K - K^{new}$, all top-level components labelled S_i or R_j are preserved and that it satisfies anaphoric integrity.

Definition 4.1 (Part containing the integrity constraint)
Let K and K' be labelled DRSs. and let K satisfy anaphoric integrity. K' is a part containing the integrity constraint of K, written $K' \sqsubseteq_{DR}^{IC} K$, if $K' \sqsubseteq_{DR} K$ and both $Sl(K') = Sl(K)$ and $Rl(K') = Rl(K)$.

There are two kinds of postulates for the DRS contraction operation: the requirements from discourse representation theory (anaphoric integrity and definedness) and the general conditions on contractions (success and inclusion). They are recorded in the following definition. Further conditions can be imposed if needed.

Definition 4.2 (DRS contraction operation)
A function $-$ from DRSs satisfying anaphoric integrity and defined update DRSs to DRSs is called a DRS contraction if the following conditions hold:

Anaphoric Integrity $ai(K - K_\alpha)$

Success Either $IC(K) \vdash K_\alpha$

Or $K - K_\alpha \nvdash K_\alpha$

Inclusion The result of the contraction is a part of the original DRS: $K - K_\alpha \sqsubseteq_{DR} K$

The topic of the next subsection is a concrete realization of a contraction operation. Now maxichoice contraction for DRSs taking into account anaphoric integrity will be defined. Given the decision to view DRSs as belief bases rather than belief sets, the fact that most accounts of presupposition justification insist on *minimality* of the adjustments of the context and the arguments of [8] to the effect that the problems of maxichoice contraction are only acute when it is applied to belief sets, but not with belief bases, the choice of maxichoice contraction has at least some justification. In order to save space, I will directly define maxichoice contraction respecting integrity constraints instead of first presenting a DRT- version of conventional maxichoice contraction. The handling of the integrity constraint is built into the definition of remainder set.

DEFINITION 4.3 (Remainder set)
Let K be a non-empty DRS satisfying anaphoric integrity and let K_α be defined with respect to K. Suppose furthermore that $K \vdash K_\alpha$. The remainder set $K \perp K_\alpha$ is the set of DRSs that contains exactly those DRSs K^{rem} such that

- $K^{rem} \sqsubseteq_{DR} K$
- $K^{rem} \not\vdash K_\alpha$
- There is no K^{rem}_{sub} with $K^{rem}_{sub} \sqsubseteq_{DR} K$ and $K^{rem}_{sub} \not\vdash K_\alpha$
- K^{rem} satisfies the integrity constraint both with respect to information from the immediate situation and with respect to general rules:
  - $Sl(K^{rem}) = Sl(K)$
  - $Rl(K^{rem}) = Rl(K)$

Note that the remainder set may be empty if K_α is entailed by the situational and rule knowledge.

DEFINITION 4.4 (Selection function)
A selection function γ yields, for each non-empty set of DRSs D, a member of D.

The following method of belief contraction is conservative, the change effected is minimal. The operation is called *guarded* maxichoice contraction because of the provision for the case in which the remainder set is empty because of the integrity constraints. In this case it is not possible to contract, but instead of returning an error message, the original belief base is returned.

DEFINITION 4.5 (Guarded Maxichoice Contraction)
The guarded maxichoice contraction of K with K_α modulo the selection function γ, $K - K_\alpha$ is given by

- If $K \perp K_\alpha \neq \emptyset$, then $K - K_\alpha = \gamma(K \perp K_\alpha)$,
- if $K \perp K_\alpha = \emptyset$, then $K - K_\alpha = K$.

Revision can be defined in terms of contraction in the usual way.

5 Judgments of unresolved sentences

This paper is about the way in which conversation participants judge the assertive natural language utterances of other participants on the basis of their available information, not about the assignment of truth values to sentences relative to a model.

There are two main differences between the two processes that are relevant here. A model of predicate logic is complete in assigning a valuation to each atomic sentence. Information states are not, the information of a participant is partial and will only decide the truth or falsity of some sentences. Another difference between the two scenarios is that the translation of natural language sentences often contains free variables and presuppositions. In the literature on dynamic semantics (see e.g. [4], [3]) the resolution of anaphoric expression has usually been taken for granted by pre-indexing the translations of e.g. pronouns as variables to coincide with the variable dynamically bound by a currently active existential quantifier. In DRT, pronouns are usually translated as new variables that have to be equated, in a separate process of anaphora resolution, with an accessible discourse referent. The modelling in DRT is certainly more realistic from a computational point of view. However, it also makes the task of judging a sentence more complicated. By way of motivation for the definitions and the algorithm that will follow, let us ask: What would be a natural method of judging a sentence containing unresolved pronouns and definite descriptions ? Of course, a very radical and uncooperative method would be to simply refuse judging sentences when the reference cannot be uniquely determined. But in communication in natural language, speakers are usually interested in obtaining as much information as possible from what is said, even if this involves substantial reasoning. One reason for this is certainly that communication often takes place under severe efficiency constraints, there is no time for unlimited clarification dialogues. The following assumptions almost follow from the assumption that a participant tries to assign a specific judgment to a sentence whenever possible, and to move to a more differentiated judgment if a simple one is not available. The participant can use two main tools for arriving at judgments when, at first sight, none would be available: supervaluation with respect to the different anaphor resolution possibilities and contraction with the negation of a presupposition. It is plausible to assume that more complicated ways to justify a sentence are only tried out if no simpler ones are available, thus, the repair mechanism of accommodation is only resorted to when there is no resolution available, in order to keep the number of possible justifications minimal, and contraction with the negation of the presupposition is only carried out if there is no binding or accommodation. This assumption can be justified by the pragmatic consideration that if there is a resolution, it would normally be uncooperative that the speaker had intended an accommodation interpretation (although this argument depends on the exact role of the common ground that is not discussed here) and by the higher complexity of accommodation and contraction. Let us take a method for classifying resolved sentences as true, false or undecided with respect to an information state for granted. The term *resolution* is used for justifying a context-dependent expression by only introducing resolution equations. The more general term *justification* covers cases where additional assumptions are made (accommodations) as well. If the sentence can be resolved in a unique way, it should be judged just like its unique resolution. If there are several resolution possibilities, the only possibility is to see whether the resolution does not matter for the truth value. The sentence should be judged true (false) only if it is true under all resolution possibilities. Otherwise it should be judged as referentially unclear. If there is *no* resolution possibility, the sentence would normally be judged as undefined or truth-valueless in the sense of [9]. However, a cooperative participant may try to adapt his information state in order to nevertheless make room for the presupposition of the speaker – after all, presupposed information is presented as uncontroversial, so the speaker may have had good reasons in introducing the presupposition, and in any case it is cooperative to go along with it in order to keep the conversation going. There are two cases. Either such a contraction of the information state is not possible. Then the sentence should be judged as radically undefined. Or it is possible, and then the final judgment depends on the justification possibilities with respect to the contracted context. The judgments *undefined but false/true/unclear after revision* may result. This derivation of a certain method of judging sentences only from the requirement of saying as much as possible about the truth of a sentence on the basis of the given information as independent motivation for the algorithm in section 6. But first the basic notions of truth (or support), falsity (or incompatibility) and undecidedness of a *resolved* open proposition with respect to a context have to be defined.

DEFINITION 5.1 (Support)
Let K be a DRS and let K^+ be defined with respect to K. Then K^+ is said to be supported by K iff $K \vdash_{update} K^+$.

DEFINITION 5.2 (Incompatibility)
Let K be a DRS and let K^+ be defined with respect to K.
Then K^+ is said to be incompatible with K iff $K \vdash_{update} \neg K^+$.

DEFINITION 5.3 (Undecided)
Let K be a DRS and let K^+ be defined with respect to K.
Then K^+ is said to be *undecided* with respect to K iff neither $K \vdash_{update} K^+$ nor $K \vdash_{update} \neg K^+$.

In the next section, an algorithm is defined that forms the judgments described at the beginning of this section in the order that suggests itself, namely by increasing complexity of the judgment[5]. First come the cases where the DRS to be judged is already resolved, then the cases where it is resolvable by (i) binding, (ii) accommodation without revision, (iii) accommodation with revision.

6 Judgment algorithm

Let K be a DRS satisfying anaphoric integrity and K^+ be a simple DRS representing the sentence s.

(1) If K^+ is resolved with respect to K:
   If K^+ is supported by K according to Def. 5.1, then judge s as *true*.
   If K^+ is incompatible with K according to Def. 5.2, then judge s as *false*.
   If K^+ is undecided by K according to Def. 5.3, then judge s as *unknown*.

(2) Otherwise compute the justifications $Just(K, K^+)$ of K^+ with respect to K, according to Def. 3.8.
   If there is a unique justification $Just(K, K^+) = \{K^{just}\}$ consisting of resolution equations only, then judge K^+ with respect to $K; K^{just}$ according to step (1).
   If there is a unique justification $Just(K, K^+) = \{K^{just}\}$ consisting of accommodated information and possibly resolution equations, then judge K^+

[5]The method is an algorithm in the same sense as e.g. the presupposition justification algorithm of [10]. Termination is not guaranteed in the general case, because checking DRS consistency is equivalent to checking consitency for first-order logic. For the typical contexts used in discussions of linguistic examples this problem does not arise.

with respect to K; K^{just} according to step 1 and add the qualification *after accommodation* to the resulting judgment.
If the set of justifications $Just(K, K^+)$ contains more than one element, then
if s is true under all justifications, then judge it as *true*.
if s is false under all justifications, then judge it as *false*.
otherwise judge s as *referentially unclear*.
If $Just(K, K^+) = \emptyset$, then contract K with $mp(K^+)$.
if $K - \neg mp(K^+) = K$ (the contraction fails),
then judge s *completely unresolvable* (radical presupposition failure).
if $K - \neg mp(K^+) \neq K$ (the contraction goes through), then apply step 2 with respect to $K - \neg mp(K^+)$ and K^+ and wrap the result Res in the qualification *undefined but Res after revision with the presupposition.*

The case division is exhaustive because there is no justification if and only if the context DRS entails the negation of the presupposition. This aspect should still be refined, because judging a sentence true only after accommodation is intuitively different from judging it true directly in its context. The presence of the outcome *unclear after revision* is motivated by situations in which, e.g., the presupposition of one definite description in a sentence is inconsistent with the context, while another definite description has several possible antecedents.
Using the order given in the algorithm has the advantage of calling the expensive contraction operation only as a last resort.

7 Treatment of an example

EXAMPLE 7.1
The king of France is sitting on this chair.

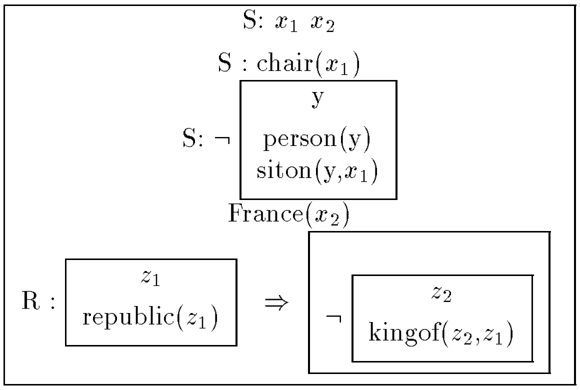
Let us assume that we have the following context DRS K:

$$\boxed{\begin{array}{c} \text{S: } x_1\ x_2 \\ \text{S : chair}(x_1) \\ \text{S: } \neg \boxed{\begin{array}{c} \text{y} \\ \text{person(y)} \\ \text{siton(y,}x_1) \end{array}} \\ \text{France}(x_2) \\ \text{republic}(x_2) \\ \boxed{\begin{array}{c} z_1 \\ \text{republic}(z_1) \end{array}} \Rightarrow \boxed{\neg \boxed{\begin{array}{c} z_2 \\ \text{kingof}(z_2,z_1) \end{array}}} \end{array}}$$

The sentence is formalized by the open DRS K^+:

$$\boxed{\text{siton}(\iota x_3 : kingof(x_3, \iota x_4 : France(x_4)),\delta x_5 : chair(x_5))}$$

$Just(K, K^+)$ is empty. The contraction $K - \neg x_3; x_4; France(x_4); kingof(x_3, x_4)$ with the critical presupposition (the demonstrative can be resolved unproblematically) yields



The belief that France is a republic had to be given up .
In the new context, K^+ is judged *false (after revision)* using definition (5.2).

References

[1] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[2] David Beaver. Presupposition. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 939–1009. North-Holland, 1997.

[3] Paul Dekker. *Transsentential Meditations.* PhD thesis, University of Amsterdam, 1993.

[4] Jeroen Groenendijk and Martin Stokhof. Dynamic Predicate Logic. *Linguistics and Philosophy*, 14:39–100, 1991.

[5] Irene Heim. File Change Semantics and the Familiarity Theory of Definiteness. In R. Bäuerle, C. Schwarze, and A. v. Stechow, editors, *Meaning, Use and Interpretation of Language*, pages 164–189. De Gruyter, Berlin, 1983.

[6] Lauri Karttunen. Presupposition and linguistic context. *Theoretical Linguistics*, 1:181–94, 1974.

[7] Peter Lasersohn. Existence Presuppositions and Background Knowledge. *Journal of Semantics*, 10:112–122, 1993.

[8] David Makinson. On the status of the postulate of recovery in the logic of theory change. *Journal of Philosophical Logic*, 16:383–394, 1987.

[9] P. F. Strawson. On Referring. *Mind*, LIX:320–344, 1950.

[10] Rob van der Sandt. Presupposition projection as anaphora resolution. *Journal of Semantics*, 9:333–377, 1992.

[11] Kai von Fintel. Would you believe it ? The King of France is Back ! Handout of a talk at Tübingen University, 1998.

Completeness of an Action Logic Featuring a δ-Operator for Timed Transition Systems

FERNANDO NÁUFEL DO AMARAL and EDWARD HERMANN HAEUSLER, *Department of Informatics, PUC–RJ (Catholic University of Rio de Janeiro), Brazil. E-mail: {fnaufel, hermann}@inf.puc-rio.br*

Abstract

This paper defines an action logic featuring an operator that denotes necessary conditions and postconditions of actions in a timed computational transition system. Three different semantics of the operator are discussed, and weak completeness is proved for one of them. It is also briefly shown how the action logic can be combined with a temporal logic to derive temporal properties of actions from the logical description of a real-time computational system.

Keywords: Modal Logic in Computing; Action Logic; Timed Transition Systems; Metric Temporal Logic

1 Introduction

In [10], Segerberg introduced the notion of a "bringing-it-about" δ operator, which, when applied to a given proposition q, denotes the set of actions that bring about the truth of q. In this paper, we discuss the use of a similar operator in a logical framework having computational transition systems as its semantics.

Segerberg's δ operator has led to an extension of dynamic logic containing elements from logics of action, where δq denotes actions that lead to states where q holds. In [3, 6], the operator was modified in two different ways: the first extension was introduced to reason about preconditions as well as postconditions, with the dyadic form $p\delta q$. The second extension, concerned with real-time applications, added a minimal and a maximal delay for the occurrence of each action: here, the action term $p_l\delta^u q$ denotes the actions that achieve q in less than u units of time, provided they were enabled in states satisfying p for at least l units of time.

In [3, 6], this last extension was dubbed RETOOL (Real-Time Object-Oriented Logic, in reference to the ultimate goal of achieving a formalization of real-time object-oriented software). The exact denotation of an action term $p\delta q$, however, has not been agreed upon. Two different semantics have been proposed for RETOOL, and this paper presents yet a third one:

- [3] defined $p\delta q$ to mean all actions that achieve q starting from states where p holds, with the further requirement that these actions be enabled in every state where p holds; we call this the *Enabling Condition Semantics* (ECS) of the δ operator.
- In [6], $p\delta q$ was defined as denoting all actions a such that, if a starts from a state

where p holds, then it achieves q; we call this the *Material Implication Semantics* (MIS) of the δ operator.

- In this paper, we present a definition of $p\delta q$ where this term means the set of actions achieving q from states where p holds; this is the *Necessary Condition Semantics* (NCS) of the δ operator.

The most promising semantic approaches regarding RETOOL as a logical language for describing computational transition systems seem to be the ECS and the NCS. In this paper we briefly discuss the reasons why the MIS may not be so suitable for modeling computational transition systems. We define a version of RETOOL corresponding to the NCS approach with a sound and (weakly) complete proof theory. This version of RETOOL may be useful as a logical language in a categorial framework for describing concurrent (possibly open) systems.[1] We conclude by briefly discussing the use of RETOOL combined with a temporal logic (particularly MTL [4]) for validating logic-based system descriptions.

2 Defining RETOOL

2.1 The Language

The primitive syntactic entities of RETOOL are *attribute symbols* (which, in this propositional version of the logic, are simple propositional letters), and *action symbols*. We denote the set of attribute symbols as A, and the set of action symbols as Γ.
The logic presupposes an infinite totally ordered set (TIME, $\leq$), with minimum 0. A constant ∞ is available, such that $\infty \notin$ TIME and $t \leq \infty, \forall t \in$ TIME. Also available are countably many constants, one for each member of TIME. An adequate theory of (TIME $\cup\{\infty\}$, $\leq$) is assumed to be contained in the logic; however, to make the presentation clearer, the axioms of this theory are not made explicit here. The other syntactic categories are:

- State Propositions (SP): $p ::= a \mid \neg p \mid p \rightarrow p'$, where $a \in A$;
- Action terms (AT): $t ::= g \mid p_l \delta^u q$, where $g \in \Gamma$, $p, q \in SP$, $l \in$ TIME, $u \in$ TIME $\cup\{\infty\}$, and $l \leq u$;
- Formulae: $\phi ::= a \mid t_1 \supset t_2 \mid \neg\phi \mid \phi \rightarrow \phi' \mid [t]\phi \mid [\,]p$, where $a \in A$, $t, t_1, t_2 \in AT$, and $p \in SP$.

There are also two unary function symbols, l and u, which can be applied to action terms to yield their time bounds (i.e., elements of TIME $\cup\{\infty\}$ – see below). The time bounds $l(g)$ and $u(g)$ of a primitive action symbol g are of an extra-logical nature; the time bounds $l(p_x\delta^y q)$ and $u(p_x\delta^y q)$ of an action term built with the δ operator are, respectively, the constants x and y, and can be considered abbreviations thereof.

2.2 Semantics

The semantics of RETOOL is defined over structures that are based on the notion of *timed transition systems* [8]: given a set A of attribute symbols and a set Γ of action symbols, a *timed frame* $\mathcal{F}$ for A and Γ is a sextuple $(W, \rightarrow, l, u, I, w_0)$, where

[1] We do not discuss this framework here. The reader is referred to [2, 5].

- W is a set of states;
- For each $g \in \Gamma$, $\xrightarrow{g} \subseteq W \times W$ is the transition relation for action g;
- l maps each $g \in \Gamma$ to an element $l(g) \in$ TIME;
- u maps each $g \in \Gamma$ to an element $u(g) \in$ TIME $\cup\{\infty\}$ such that $u(g) \geq l(g)$;
- $I : A \rightarrow 2^W$ is an interpretation of the attributes, where each $a \in A$ is assigned the set of worlds where a is true;
- w_0 is the initial state.

Every action $g \in \Gamma$ has a lower bound $l(g)$ and an upper bound $u(g)$. Intuitively, the lower bound defines the minimum delay that must be observed for the transition to take place (provided all the necessary conditions for the occurrence of such a transition are satisfied). The upper bound defines the maximum delay during which the transition must occur (again, provided all necessary conditions are satisfied). Formally, lower and upper bounds are defined through the use of the notion of *computation*:
A *timed state sequence* [8] for a timed frame is a pair $\rho =< \sigma, \mathrm{T} >$, where σ is an infinite sequence of states ($\sigma_i \in W$) and T is an infinite sequence of corresponding times ($\mathrm{T}_i \in$ TIME), satisfying:

- monotonicity: for all $i \geq 0$, either $\mathrm{T}_{i+1} = \mathrm{T}_i$, or ($\mathrm{T}_{i+1} > \mathrm{T}_i$ and $\sigma_{i+1} = \sigma_i$).
- progress: for every $t \in$ TIME, there is $i \geq 0$ such that $\mathrm{T}_i \geq t$.

A *computation* [8] over a timed frame is a timed state sequence $< \sigma, \mathrm{T} >$ such that

- σ is a computation of the underlying transition system, i.e., for every $i \geq 0$, there is a transition $\sigma(i)$ such that $\sigma_i \xrightarrow{\sigma(i)} \sigma_{i+1}$;
- (lower bound): for every $i \geq 0$ in the domain of σ, there is a $j \leq i$ such that $\mathrm{T}_i - \mathrm{T}_j > l(\sigma(i))$ and $\sigma(i)$ is enabled in every state σ_k for $j \leq k \leq i$.
- (upper bound): for every $g \in \Gamma$ and $i \geq 0$, there is $j \geq i$ with $\mathrm{T}_j - \mathrm{T}_i \leq u(g)$ such that either g is not enabled at σ_j or $g = \sigma(j)$.

The denotation of a state proposition p in a timed frame $\mathcal{F}$ is the set of states defined as follows:

- $[\![a]\!] = I(a)$;
- $[\![\neg p]\!]^{\mathcal{F}} = W \setminus [\![p]\!]^{\mathcal{F}}$;
- $[\![p \rightarrow p']\!]^{\mathcal{F}} = (W \setminus [\![p]\!]^{\mathcal{F}}) \cup [\![p']\!]^{\mathcal{F}}$.

The denotation of an action term t in a timed frame $\mathcal{F}$ is the set of transitions defined as follows (where $en(g)$ is the set of states where g is enabled):

- $[\![g]\!]^{\mathcal{F}} = \{(w, w') \mid w \xrightarrow{g} w'\}$;
- $[\![p_l \delta^u q]\!]^{\mathcal{F}} = \{(w, w') \mid \exists g \in \Gamma \ \ [(w \xrightarrow{g} w') \ \wedge \ en(g) \subseteq [\![p]\!]^{\mathcal{F}} \ \wedge$ $\forall v, v'((v \xrightarrow{g} v') \Rightarrow (w' \in [\![q]\!]^{\mathcal{F}})) \ \wedge \ (l \leq l(g) \leq u(g) \leq u)]\}$

Finally, the satisfaction of a formula by a timed frame $\mathcal{F}$ at a state w is defined by:

- $\mathcal{F}, w \models p$ iff $w \in [\![p]\!]^{\mathcal{F}}$;
- $\mathcal{F}, w \models (t_1 \supset t_2)$ iff $[\![t_1]\!]^{\mathcal{F}} \subseteq [\![t_2]\!]^{\mathcal{F}}$;

- $\mathcal{F}, w \models \neg\phi$ iff not $\mathcal{F}, w \models \phi$;
- $\mathcal{F}, w \models \phi \rightarrow \phi'$ iff $\mathcal{F}, w \models \phi$ implies $\mathcal{F}, w \models \phi'$;
- $\mathcal{F}, w \models [t]\phi$ iff $\mathcal{F}, w' \models \phi$ for every w' such that $(w, w') \in [\![t]\!]^{\mathcal{F}}$;
- $\mathcal{F}, w \models [\,]p$ iff $\mathcal{F}, w_0 \models p$.

2.3 Axiomatization

The axiom schemes and rules of inference below comprise an adequate axiomatization of RETOOL. In what follows, Λ represents a set of RETOOL formulae, the derivability relation $\vdash$ is defined in the usual manner, and $enabled(t)$ is an abbreviation for the formula $\neg[t]\bot$ (see comments in the next section). For lack of space, we do not prove the soundness of this axiomatization, but it should be easy to see that the axioms and rules below are indeed sound for the Necessary Condition Semantics of RETOOL.

2.4 Comments

- "$\sqsupset$" is the *subsumption* operator. To say that action term t_1 subsumes action term t_2 is to say that every action denoted by t_1 is also denoted by t_2 (but not necessarily the other way around). Subsumption can be seen as a refinement on actions: the actions denoted by t_1 refine those denoted by t_2.

- We define, for an action term t, the set $en(t) = \{w \in W \mid \exists w'$ such that $(w, w') \in [\![t]\!]\}$. This is the set of all states where at least one of the actions in the denotation of t is enabled. Note that saying that a world w is a member of $en(t)$ is equivalent to saying that the formula $\neg[t]\bot$ is true in w. In fact, we define the abbreviation $enabled(t)$ to mean $\neg[t]\bot$.

The present work departs from the semantics of the δ operator given in [6]. There, $[\![p_l\delta^u q]\!]$ is defined as the set of all transitions (w, w') such that $w \in [\![p]\!]$ implies $w' \in [\![q]\!]$. This semantics mirrors the meaning of the Hoare triples $\{p\}g\{q\}$ (see [9]), namely that all terminating runs of program g starting in states satisfying p will end in states satisfying q (and there is no guarantee about runs that start in states not satisfying p). According to this view, p could be considered a precondition of program g, with q a relative postcondition.

Now, the semantics presented here requires that for (w, w') to be denoted by $p_l\delta^u q$, it must be the case that $w \xrightarrow{g} w'$ for some g such that $en(g) \subseteq [\![p]\!]$. The requirement that p must be true in all worlds where g is enabled forces us to see p as a necessary condition of the program represented by g. It is simply impossible to have runs of g starting in states not satisfying p.

This "necessary-condition" semantics (NCS) of the δ operator seems to us more appropriate to reason about timed transition systems as abstract models of complex real-time reactive systems. The "functionality" of an action, in the sense captured by Hoare triples, can still be expressed using action modalities. I.e., the Hoare triple $\{p\}g\{q\}$ is closely related to the RETOOL formula $p \rightarrow [g]q$.

(PC)	All axioms and rules of propositional calculus
(K)	$[t](\phi \rightarrow \psi) \rightarrow ([t]\phi \rightarrow [t]\psi)$ $[\,](p \rightarrow q) \rightarrow ([\,]p \rightarrow [\,]q)$
(I)	$[\,]p \rightarrow [t][\,]p$ $[t][\,]p \rightarrow (enabled(t) \rightarrow [\,]p)$ $[\,]\neg p \leftrightarrow \neg[\,]p$
(N)	$\dfrac{\Lambda \vdash \phi}{\Lambda \vdash [t]\phi} \quad \dfrac{\Lambda \vdash p}{\Lambda \vdash [\,]p}$
(δ)	$\dfrac{\Lambda \vdash enabled(t) \rightarrow p \quad \Lambda \vdash [t]q \quad \Lambda \vdash l \leq l(t) \leq u(t) \leq u}{\Lambda \vdash t \supset p_l\delta^u q}$
(S1)	$t \supset t$
(S2)	$(t_1 \supset t_2) \rightarrow ((t_2 \supset t_3) \rightarrow (t_1 \supset t_3))$
(S3)	$(t_1 \supset t_2) \rightarrow ([t_2]\phi \rightarrow [t_1]\phi)$
(NC)	$(t \supset p_l\delta^u q) \rightarrow (enabled(t) \rightarrow p)$
(Post)	$(t \supset p_l\delta^u q) \rightarrow ([t]q)$
(Bounds)	$(t \supset p_l\delta^u q) \rightarrow (enabled(t) \rightarrow l \leq l(t) \leq u(t) \leq u)$
(Global-$\supset$)	$(t_1 \supset t_2) \leftrightarrow [t](t_1 \supset t_2)$
(δ-Bounds)	$l(p_l\delta^u q) = l \quad u(p_l\delta^u q) = u$

FIG. 1. An Axiomatization of RETOOL

3 Weak Completeness

In order to show that the axiomatization is complete, we build a canonic model, whose worlds are maximally consistent sets of RETOOL formulae (see [7] for definitions and properties of maximally consistent sets), and whose transitions are determined from the contents of the worlds in the fashion described below. Since every consistent formula is a member of some maximally consistent set, and since satisfaction of a formula ϕ in a world w of the canonic model is equivalent to ϕ being a member of w (proven in the Coincidence Lemma), this ensures that *every* consistent formula will be satisfied in the canonic model.

3.1 Construction of the Canonic Model

The set A^c of attribute symbols of the canonic model is arbitrary: $A^c = \{a_1, a_2, ...\}$. For the construction of the set Γ^c of action symbols, we introduce the following definitions:

DEFINITION 3.1
For each $p \in SP$,[2] $\tilde{p} = \{r \mid \vdash p \leftrightarrow r\}$

DEFINITION 3.2
$\Gamma^c = \{\text{“}\tilde{p}_l\delta^u\tilde{q}\text{”} \mid p, q \in SP \setminus \tilde{\bot}, l \in \text{TIME}, u \in \text{TIME } \cup \{\infty\}\}$

We define here an action symbol for each quadruple (p, q, l, u) composed of satisfiable state propositions p and q (up to tautological equivalence) and time bounds l and u. In order to ensure that these action symbols have the desired meaning, we define the following set Σ of RETOOL formulae:

DEFINITION 3.3
$\Sigma = \{(\text{“}\tilde{p}_l\delta^u\tilde{q}\text{”} \supset p_l\delta^u q) \wedge (p_l\delta^u q \supset \text{“}\tilde{p}_l\delta^u\tilde{q}\text{”}) \mid p, q \in SP \setminus \tilde{\bot},\ \ l \in \text{TIME},\ \ u \in \text{TIME } \cup \{\infty\}\}$

Note that in every model of Σ, each action symbol “$\tilde{p}_l\delta^u\tilde{q}$” will function as a “witness” to action term $p_l\delta^u q$, the denotation of the symbol corresponding exactly to the denotation of the term.
More extra-logical information is necessary to build the canonic model: we must specify the initial state through a set Θ of formulae. If we want state propositions $p_1, p_2, ...$ to be true at the initial state w_0, we define $\Theta = \{[\,]p_1, [\,]p_2, ...\}$.[3] Obviously, the set $\{p_1, p_2, ...\}$ must be consistent.
We also introduce some special notation to refer to “demodalized” formulae, where w is a set of formulae and t is any action term:

DEFINITION 3.4
$w \setminus [t] = \{\phi \mid [t]\phi \in w\}$

DEFINITION 3.5 (The canonic model)
Given the sets A^c, Σ, Θ and Γ^c, we define the canonic model as $M^c = (W^c, \{\xrightarrow{g} \mid g \in \Gamma^c\}, l^c, u^c, I^c, w_0^c)$, where

W^c	$= \{\Phi \mid \Phi$ is a maximally consistent set containing $\Sigma \cup \Theta\}$.
$\xrightarrow{\text{“}\tilde{p}_l\delta^u\tilde{q}\text{”}}$	$= \{(w, w') \in W^c \times W^c \mid w \setminus [\text{“}\tilde{p}_l\delta^u\tilde{q}\text{”}] \subseteq w'\}$
l^c	is such that $l^c(\text{“}\tilde{p}_l\delta^u\tilde{q}\text{”}) = l$ for every “$\tilde{p}_l\delta^u\tilde{q}$” $\in \Gamma^c$.
u^c	is such that $u^c(\text{“}\tilde{p}_l\delta^u\tilde{q}\text{”}) = u$ for every “$\tilde{p}_l\delta^u\tilde{q}$” $\in \Gamma^c$.
I^c	is such that $I^c(p) = \{w \in W^c \mid p \in w\}$ for every $p \in SP$.
w_0^c	is some world in W^c satisfying $\Theta \setminus [\,]$.

Central to the completeness proof is the Coincidence Lemma:

$$\forall w \in W^c,\ \forall \phi :\quad \phi \in w \quad \Leftrightarrow \quad M^c, w \models \phi$$

[2]We refer to the set of state propositions as SP, and to the set of action terms as AT.
[3]Actually, this leads to a *class* of canonic models, one model for each choice of Θ.

This is proved by induction over the formation of ϕ. We sketch the proof of only one case here, using the following lemmas:[4]

LEMMA 3.6
For all $w \in W^c$, for all $p, q \in SP \setminus \tilde{\bot}$, for all $l \in$ TIME , $u \in$ TIME $\cup \{\infty\}$, for all $t \in AT$, and for all formulae ϕ:

$$\begin{array}{lcl} w \vdash t \supset p_l \delta^u q & \Leftrightarrow & w \vdash t \supset \text{``}\tilde{p}_l \delta^u \tilde{q}\text{''} \\ w \vdash p_l \delta^u q \supset t & \Leftrightarrow & w \vdash \text{``}\tilde{p}_l \delta^u \tilde{q}\text{''} \supset t \\ w \vdash [p_l \delta^u q]\phi & \Leftrightarrow & w \vdash [\text{``}\tilde{p}_l \delta^u \tilde{q}\text{''}]\phi \end{array}$$

LEMMA 3.7
For all $w \in W^c$, for all $p, q \in SP \setminus \tilde{\bot}$, for all $l \in$ TIME , $u \in$ TIME $\cup \{\infty\}$:

$$w \in en(p_l \delta^u q) \quad \Leftrightarrow \quad w \vdash enabled(p_l \delta^u q)$$

By lemma 3.6, the equivalence also holds for the formula $enabled(\text{``}\tilde{p}_l \delta^u \tilde{q}\text{''})$.

LEMMA 3.8
In the canonic model, $[\![\text{``}\tilde{p}_l \delta^u \tilde{q}\text{''}]\!] = [\![p_l \delta^u q]\!]$.

The subsumption case of the Coincidence Lemma: $\phi = t_1 \supset t_2$

By lemmas 3.6 and 3.8, we may restrict ourselves to the case where $t_1 = p_l \delta^u q$ and $t_2 = r_m \delta^n s$:

($\Rightarrow$) Suppose $p_l \delta^u q \supset r_m \delta^n s \in w$. Let $(w, w') \in [\![p_l \delta^u q]\!]$. Then

$$\begin{array}{lcl} p_l \delta^u q \supset r_m \delta^n s \in w & \Rightarrow & \text{(by axiom (S3))} \\ \forall \phi \quad [r_m \delta^n s]\phi \to [p_l \delta^u q]\phi \quad \in w & \Rightarrow & \text{(as } w \text{ is maximally consistent)} \\ \forall \phi \quad [r_m \delta^n s]\phi \in w \quad \Rightarrow \quad [p_l \delta^u q]\phi \in w & \Rightarrow & \text{(as } w \setminus [p_l \delta^u q] \subseteq w'\text{)} \\ w \setminus [r_m \delta^n s] \subseteq w' & \Rightarrow & \text{(by the definition of } M^c\text{)} \\ (w, w') \in [\![r_m \delta^n s]\!] & & \end{array}$$

($\Leftarrow$) Suppose $w \models p_l \delta^u q \supset r_m \delta^n s$. Then, for every $w \in W^c$,

$$\begin{array}{lcl} [\![p_l \delta^u q]\!] \subseteq [\![r_m \delta^n s]\!] & \Rightarrow & \\ w \models enabled(p_l \delta^u q) \quad \Rightarrow \quad w \models r & \Leftrightarrow & \text{(by lemma 3.7 and the definition of } I^c\text{)} \\ w \vdash enabled(p_l \delta^u q) \quad \Rightarrow \quad w \vdash r & \Leftrightarrow & \text{(as } w \text{ is maximally consistent)} \\ w \vdash enabled(p_l \delta^u q) \to r & \Leftrightarrow & \text{(as } w \text{ is arbitrary containing } \Sigma \cup \Theta\text{)} \\ \Sigma \cup \Theta \vdash enabled(p_l \delta^u q) \to r & & \end{array}$$

As for the postcondition, we have that, for all $w \in M^c$,

$$\begin{array}{lcl} [\![p_l \delta^u q]\!] \subseteq [\![r_m \delta^n s]\!] & \Rightarrow & \\ w \models [p_l \delta^u q]s & \Rightarrow & \text{(by lemma 3.8)} \\ w \vdash [p_l \delta^u q]s & \Leftrightarrow & \text{(as } w \text{ is arbitrary containing } \Sigma \cup \Theta\text{)} \\ \Sigma \cup \Theta \vdash [p_l \delta^u q]s & & \end{array}$$

As for the time bounds, we have that $m \leq l \leq u \leq n$. There are derived rules that state that necessary conditions, postconditions and time bounds can be weakened. Therefore, $\Sigma \cup \Theta \vdash p_l \delta^u q \supset r_m \delta^n s$, and we conclude that, as any $w \in M^c$ contains $\Sigma \cup \Theta$, it is the case that $w \vdash p_l \delta^u q \supset r_m \delta^n s$.

---

[4] Detailed proofs of these and all other relevant lemmas can be found in [1].

4 Abstracting Temporal Properties of Systems

Since the semantics of the time-bounded δ operator is given in terms of computations, it is only natural that we investigate the combination of RETOOL with a linear-time temporal logic, whose models are also computations. In these concluding remarks, we briefly describe the nature of this combination, which has been studied for the MIS semantics of RETOOL in [6] and can be easily adapted to the NCS semantics presented in this paper.
The temporal logic chosen was MTL (see [4]), which features time-bounded temporal operators $\mathbf{X}_{Rc}$ (next state) and $\mathbf{U}_{Rc}$ (until), where $c \in \mathrm{TIME} \cup \{\infty\}$ and R is a relation on TIME. We extend this language by adding RETOOL's action terms as primitive propositions and stipulate that an action term t is true at step i of a computation iff the denotation of t contains the action responsible for the transition taken at step i.
The following are examples (taken from [6] and modified to fit our NCS semantics) of proof rules relating RETOOL and MTL:

$$\frac{\mathrm{enabled}(t) \rightarrow r \qquad t \supset p_0\delta^{\infty} q}{t \rightarrow r \wedge \mathbf{X}_{=0} q}$$

This rule states that an action denoted by t (with p as a necessary condition and q as a postcondition) establishes q in the next state whenever this action is responsible for the transition to the next state. In compliance with the definition of computation, the transition does not take any time.

$$\frac{p \rightarrow \neg\mathrm{enabled}(t) \qquad t \supset \top_x\delta^{\infty}\top}{p \rightarrow \mathbf{G}_{\leq x}\neg t}$$

This rule captures a safety property concerning the lower bound of an action: if the truth of p means that action t is disabled, and t has lower bound x, we conclude that, p holding in the current state, t will not be taken for at least x time units (the $\mathbf{G}_{\leq x}$ bounded temporal operator meaning "during the next x time units").
Additional rules and an example of a detailed system description can be found in [6] for the MIS semantics. They can be easily adapted to the NCS semantics we employ here.

5 Concluding Remarks

In this paper, we have shown how Segerberg's δ operator can be extended to define RETOOL, an action logic meant to reason about computational timed transition systems. Three alternative semantics have been considered for action terms built with this operator, and a weak completeness proof has been sketched for one of these semantics (the NCS).
It was also indicated how the combination of RETOOL with a temporal logic can be useful in deriving properties of actions from system descriptions.
Work in progress includes the study of the ECS as an alternative semantics, with soundness and completeness proofs, and the comparison of the ECS and NCS approaches in the specification of real-time computational systems.

Acknowlegdements

We would like to thank Tom Maibaum and José Fiadeiro for the many fruitful discussions about RETOOL.

References

[1] Amaral, F.N., "RETOOL: Uma Lógica de Ações para Sistemas de Transição Temporizados", M.Sc. Dissertation, Dept. of Informatics, PUC-RJ, Brazil, 2000.

[2] Bicarregui, J., Lano, K. and Maibaum, T., "Towards a Compositional Interpretation of Object Diagrams", in *Proc IFIP Working Conference on Algorithmic Languages and Calculi*, Chapman and Hall, 1997.

[3] Carvalho, S., Fiadeiro, J. e Haeusler, E.H., "A Formal Approach to Real-Time Object-Oriented Software", in *Proc. 22nd IFAC/IFIP Workshop on Real-Time Programming WRTP'97*, Elsevier 1997.

[4] Chang, E., *Compositional Verification of Reactive and Real-Time Systems*, PhD Thesis, Stanford University, 1995.

[5] Fiadeiro, J. and Maibaum, T., "Temporal Theories as Modularization Units for Concurrent Systems Specification", in *Formal Aspects of Computing* 4(3), 1992.

[6] Fiadeiro, J. and Haeusler, E.H., "Bringing It About On Time (Extended Abstract)", in *Proc. I IMLLAI, Fortaleza, CE, Brazil*, 1998.

[7] Goldblatt, R., *Logics of Time and Computation*, CSLI Lecture Notes 7, CSLI, 1992.

[8] Henzinger, T., Manna, Z., e Pnuelli, A., "Timed Transition Systems", in *Real Time: Theory in Practice* (J.W. de Bakker, C. Huizing, W.P. de Roever e G. Hozenberg (eds.)), LNCS 600, Springer-Verlag, 1992.

[9] Hoare, C.A.R., "An Axiomatic Basis for Computer Programming", in *Comm. ACM* 12, 1967.

[10] Segerberg, K., "Bringing It About", in *Journal of Philosophical Logic* 18, 1989.

A Fibring Semantics for the Semantic-Morphological Interface in Natural Language

RALF NAUMANN and ANJA LATROUITE, *Seminar für Allgemeine Sprachwissenschaft, University of Düsseldorf, Germany.*
E-mail: naumann@phil-fak.uni-duesseldorf.de

Abstract

In Lexical Decompositional Grammar (LDG), Wunderlich (1997), a verb v in the lexicon is analyzed with respect to different levels like the *Semantic Form* (SF), the *Theta-Structure* (TS) and the *syntactic-morphological* (S/M) level. What distinguishes LDG from other grammatical theories like LFG or HPSG is the way the different parts are related to each other. E.g., at the level of linking the assignment of case to an argument NP at the morphological level is constraint by the position of the corresponding argument at the level of SF.
In this paper a formalization of LDG is presented. Structures for the analysis of verbs in LDG are tuples of the form $< \mathbf{S}_1, ..., \mathbf{S}_n, \gamma_1, ..., \gamma_m >$ such that each $\mathbf{S}_i$ is the substructure corresponding to the i-th level of linguistic analysis and each γ_j is a mapping relating the domains of two substructures. If $\mathbf{S}$ and $\mathbf{S}'$ are related by some γ, the information that is associated with the objects that are related by γ is constraint. E.g., if some $d \in D_{\mathbf{S}}$ has property P, $\gamma(d) \in D_{\mathbf{S}'}$ is required to have property P'. The combination of different ontologies is reflected at the syntactic level by *fibring* or *layering* the languages that are used to talk about the structures over each other.

1 Data and Evidence

An argument can be linked to a verb v in several ways. The German examples in (1) illustrate two principle mechanisms: that by case and that by a preposition.

(1) a. Peter aß einen Apfel. (Peter ate an apple)
b. Hans glaubt an Gott. (Hans believes in God)

Both arguments of 'essen' (eat) in (1a) as well as the first gument of 'glauben' ('believe') in (1b) are linked by case whereas the second one in (1b) is linked by a preposition. Two questions that any theory of grammar has to answer are: (α) What are the regularities of how an argument is linked to a verb, e.g., is there a relationship between argument positions and cases? and (β) To what degree is argument linking semantically driven, i.e., does it depend on some properties of the verb? Relative to the data given in (1) the following generalizations at the level of word order can be made: (a) The first argument is realized by nominative case and (b) The last argument that is realized by case gets accusative case. Pustejowsky (1991) answers question (α) in the affirmative. His approach is based on an underlying event ontology in the sense that an n-place verb v is interpreted as an $n+1$-ary relation such that $< d_1, ..., d_n, d_{n+1} > \in [\![v]\!]$ only if d_{n+1} is an event. The relationship between an event e and its participants is defined in terms of thematic roles like *Agent* or *Undergoer*. These roles are likewise used to define the mapping between the semantic component

and the argument structure of a verb. Two examples are: (i) The argument that is semantically linked to the Agent-role is realized as the highest argument of the argument structure (which often corresponds to the grammatical relation of subject) and (ii) The argument that is semantically linked to the Undergoer-role is realized as the lowest argument that is realized by case (which often corresponds to the grammatical relation of direct object). According to these two linking rules, particular thematic roles correspond to particular positions in the argument structure of a verb. If these rules are combined with the generalizations in (a) and (b), they can be extended to rules which relate particular thematic roles to particular cases: (iii) The argument that is semantically linked to the Agent-role is realized by nominative case and (iv) The argument that is semantically linked to the Undergoer-role is realized by accusative case.
The main problem that an approach like that of Pustjowsky faces is that there is strong evidence against the possibility of generalizing the rules in (i)-(iv) to all thematic roles. E.g., in 'John reached the station' and in 'Bill put the book on the table' the station and the table are both assigned the Goal-role. But the corresponding arguments are realized differently.

2 Lexical Decomposition Grammar

In Lexical Decomposition Grammar, Wunderlich (1997), three different grammatical levels of representation are distinguished: Semantic Form (SF), Theta Structure (TS) and Morphology/Syntax (MS). Although each of the three levels is characterized by its own set of structural properties, they are linked to each other by a set of mappings which constrain the relationships between them. Below this three-level architecture is illustrated by the ditransitive verb 'give'.

(2) The three grammatical levels of LDG

TS				SF
λz	λy	λx	λe	{(ACT(x) & BEC(POSS(y,z)))(e)}
+hr	+hr	-hr		
-lr	+lr	+lr		

$\Updownarrow$ $\Updownarrow$ $\Updownarrow$

ACC	DAT	NOM
MS		

When taken together, TS and SF correspond to the translation of 'give' in the representation language L which is assumed to be a variant of a many-sorted type-theory, i.e., they are a well-formed expression α of L. SF, the matrix of α, which is of type t, is based on a decompositional analysis of the verb. It is of the form $(\alpha_1 \,\&\, ... \,\&\, \alpha_n)(e)$. Each conjunct α_i is of type $< \epsilon, t >$ for ϵ the type of events (or eventualities) such that it is interpreted as a unary relation on the domain E of events. The connective & expresses a kind of temporal succession in the following sense: $\alpha_1 \,\&\, ... \,\&\, \alpha_n$ expresses a complex property of events such that this property is true of an event e just in case there are initial stages (prefixes) e_1, ..., e_n of e with α_i being true of e_i, $1 \leq i \leq n$, and e_i being an initial stage of e_j for $1 \leq i < j \leq n$. Thus, if the SF is evaluated with respect to a given event e (and suitable values for the remaining variables), each

conjunct α_i is evaluated with respect to subevents e' of e. As & is not commutative, it is possible to define a linear ordering on the (non-eventive) arguments of the α_i.
The theta-structure TS is defined in terms of an initial prefix of the sequence of variables that are bound by the λ-operators such that each variable represents an argument position. For $n > 1$ it corresponds to the sequence of the first $n-1$ λ-abstractions and to the only λ-abstraction otherwise. Each position of the sequence can be characterized in terms of properties that only depend on its place in the sequence (*structural* properties) or in terms of further information that is lexically determined (*non-structural* properties). The difference between the two sorts of properties is related to the distinction between structural and non-structural (semantic) linking. Arguments that are structurally linked are always linked by case and the assignment of case to an argument NP depends only on structural properties of the corresponding argument position, Naumann/Latrouite (2000). The structural properties are defined in terms of two binary features $[\pm lr]$ and $[\pm hr]$ with intuitive meaning 'there is a/no lower role' and 'there is a/no higher role', respectively.
The syntactic component S of MS is an ordered finite binary tree the nodes of which are decorated with the categorial and subcategorization information determined by the verb such that it contains information about the (syntactic) categories and the relative order of the arguments. The morphological component M of MS consists of a feature structure for each node of the tree that contains information appropriate for the corresponding categorial information like case, gender and number insofar as this information is already determined by the verb.
The different levels are related by various mappings that constrain the distribution of information: (i) SF - TS: relationship between the order of arguments in SF and the order of λ-abstractions. This relationship is constrained by the *Hierarchy Principle*: The order of arguments in TS is the inverse of the order of arguments in SF. Thus, if $< x_1, ..., x_n >$ is the linearly ordered sequence of arguments in SF, $< x_n, ..., x_1 >$ is the ordering in TS; (ii) TS - syntactic component of MS: The default word order is the inverse of the order of arguments in TS; (iii) syntactic component of MS - morphological component of MS: the assignment of case to argument NPs depends on the position of the corresponding argument in TS. The task consists in formalizing the ideas that have informally been presented in this section. In particular, the following questions have to be answered: (a) what is an appropriate ontology (together with a structure) for each level?; (b) what is an appropriate language (logic) to talk about this ontology?; (c) how can the constraints between the different levels be defined at the level of structures and how can they be expressed at the corresponding level of logics? In the next section suitable mathematical structures for each level will be defined, i.e. an ontology, that is a domain of entities of a particular sort, will be determined together with a set of relations.

3 The Formalization

The outline of the basic ideas on which LDG is built in the previous section has shown that the linguistic analysis of a verb in the lexicon is cast in terms of a composite ontology. There are events and 'ordinary' objects (SF), (tree) nodes (TS, S) and points (in feature structures) (M). Each sort of entity comes equipped with a set of relations, giving rise to a structure in the model-theoretic sense, with the set of entities as the domain that underlies the structure. Suitable structures for the analysis of verbs will therefore have a complex ontology consisting of a *dynamic eventuality* (sub-)structure, *ordered finite binary trees*, *feature structures* and, most importantly from a linguistic point of view, *mappings between the different substructures* that mutually constrain the relations given in the structures.

3.1 The Structures

An LDG structure for a verb v in the lexicon is a tuple
$< \mathbf{E}, \mathbf{T}_{TS}, \mathbf{T}_S, \{\mathbf{FS}_n \mid n \in N_S\}, R_{z_f}, R_{z_{TS}}, R_{z_{sf}}, \gamma >$ such that

- $\mathbf{E} = < E, S, P_v, \leq_E, C, \alpha, \omega >$ is a dynamic eventuality structure such that (i) E is a (non-empty) set of events; (ii) S is a (non-empty) set of states (or time points); (iii) P_v is the set of all events of type v; (iv) $\leq_E$ is the material part of relation on E, that is a partial order; (v) C is the composition relation that is required to be associative and that is defined by $C = \{(e, e_1, e_2) \mid e_1 \leq_E e \wedge e_2 \leq_E e \wedge \omega(e_1) = \alpha(e_2) \wedge \alpha(e) = \alpha(e_1) \wedge \omega(e) = \omega(e_2)\}$; (vi) $R_B = \{(e, e') \mid \exists e'' : Ce', ee''\}$; (vii) $\alpha : E \rightarrow S$ and $\omega : E \rightarrow S$ assign to each event $e \in E$ its beginning point and end point, respectively
- $\mathbf{T}_{TS} =< N_{TS}, R_{\succ 1_{TS}}, R_{\succ 2_{TS}}, root_{TS}, \Theta_{TS} >$ and $\mathbf{T}_S = < N_S, R_{\succ 1_S}, R_{\succ 2_S}, root_S, \Theta_S >$ are finite ordered binary trees, respectively
- each $\mathbf{FS}_n = < W_n, \{R_f\}_{f \in F}, \{Q_a\}_{a \in A}, w_{0_n} >$ is a point-generated feature structure; For $n \neq n'$, $\mathbf{FS}_n$ and $\mathbf{FS}_{n'}$ are disjoint in the sense that $W_n \cap W_{n'} = \emptyset$
- R_{z_f}, $R_{z_{ts}}$ and $R_{z_{sf}}$ are mappings between $\mathbf{T}_S$ and $\{\mathbf{FS}_n \mid \mathbf{n} \in \mathbf{N}_S\}$, $\mathbf{T}_S$ and $\mathbf{T}_{TS}$, as well as $\mathbf{E}$ and $\mathbf{T}_{TS}$
- γ is a function that maps each $\mathbf{FS}_n$ to its root w_{0_n}

In the sequel of this section the relationship between the syntactic and the morphological component of MS will be analyzed. We will begin by defining the structures. In a next step, suitable languages, L^{T_S} and L^F for talking about these structures will be defined, followed by a definition of the relation R_{z_f}, which links the two structures. Finally, the two languages are combined so that it is possible to talk about the combined ontology.

Finite Ordered Binary Trees The definition of a finite ordered tree given below applies both to $\mathbf{T}_S$ and $\mathbf{T}_{TS}$. They differ with respect to the (basic part of) languages that are used to talk about them. The definition is taken from Blackburn/Meyer-Viol (1997). A finite ordered binary tree $\mathbf{T}$ is a tuple $< N, R_{\succ_1}, R_{\succ_2}, root, \Theta >$ such that (i) N is a finite, non-empty set of (tree) nodes; (ii) $R_{\succ_i}$ for $1 \leq i \leq 2$ is a binary relation on N that is a partial function; $R_{\succ_1}(n, n')$ just in case n' is the first daughter of n and $R_{\succ_2}(n, n')$ just in case n' is the second daughter of n; (iii) *root* is the (unique) root of the tree; (iv) $\Theta \subseteq N$ is the set of terminal nodes of the tree, i.e., $\forall n[n \in N \wedge n \in \Theta \rightarrow \forall n'[n' \in N \rightarrow \neg R_{\succ_i}(n, n')]]$, i.e., terminal nodes do not immediately dominate any other node of the tree
In terms of $R_{\succ_1}$ and $R_{\succ_2}$ the following binary relations are defined:

(3) a. $R_{\succ} = R_{\succ_1} \cup R_{\succ_2}$ (the 'daughter of' relation)

b. $R_{\prec} = (R_{\succ})^c = \{(n, n') \mid (n', n) \in R_{\succ}\}$ (the 'mother of' relation, which is a partial function that is defined on $N - \{root\}$)
c. $R_{\succ^*}$ and $R_{\prec^*}$ are the transitive, reflexive closure of $R_{\succ}$ and $R_{\prec}$, respectively. $R_{\succ^*}$ is the 'dominates' relation whereas $R_{\prec^*}$ is the 'is dominated by' relation; note that due to the reflexive character one has $\forall n[R_{\succ^*}(n,n) \wedge R_{\prec^*}(n,n)]$.
d. $R_{>} = \{(n, n') \mid \exists n_1 \exists n_2 \exists n_3 [R_{\succ^*}(n_1, n) \wedge R_{\succ_1}(n_2, n_1) \wedge R_{\succ_2}(n_2, n_3) \wedge R_{\succ^*}(n_3, n')]\}$ (the precedence relation on N)
e. $R_{<}$, the succession relation on N, is defined as the inverse of $R_{>}$: $R_{<} = (R_{>})^c$

Finite rooted Feature Structures Feature Structures **FS** are defined as labeled decorated directed graphs that are multi-modal Kripke models in which every relation is a partial function (Blackburn/Meyer-Viol (1997), Blackburn et al. (1993)). A finite rooted feature structure **FS** of signature $< F, A >$, F the set of *features* and A the set of *attributes* (or *decorations*) is a triple $< W, \{R_f\}_{f \in F}, \{Q_a\}_{a \in A}, w_0 >$ such that (i) W is a non-empty set, the set of (feature structure) points; (ii) each R_f, $f \in F$, is a binary relation on W that is required to be a partial function; (iii) each Q_a, $a \in A$, is a unary relation on W; (iv) w_0 is the (unique) root of **FS**, i.e., **FS** is *generated* by w_0 in the sense that each point $w \in W$ can be reached from w_0 in a finite number of steps.

The Languages $L^{T_S}(VAR)$ and L^F The well-formed formulas of the tree language $L^{T_S}(VAR)$ are defined as follows: (i) $p \in L^{T_S}(VAR)_{wff}$, for $p \in VAR$, (ii) if ϕ, $\psi \in L^{T_S}(VAR)_{wff}$, then $\phi \wedge \psi \in L^{T_S}(VAR)_{wff}$ and $\neg\phi \in L^{T_S}(VAR)_{wff}$, (iii) if $\phi \in L^{T_S}(VAR)$, then $\alpha\phi \in L^{T_S}(VAR)$ for $\alpha \in \{\downarrow^1, \downarrow^2, \uparrow, \downarrow^*, \uparrow^*, <, >\}$ and nothing else is an element of $L^{T_S}(VAR)_{wff}$. In terms of the modalities used in clause (iii) the following modalities and constants are defined.

(4) a. $\downarrow \phi =_{def.} \downarrow^1 \phi \vee \downarrow^2 \phi$
b. $\Downarrow^i \phi =_{def.} \neg \downarrow \neg\phi$ for $1 \leq i \leq 2$; $\Downarrow \phi =_{def.} \neg \downarrow \neg\phi$ and $\Uparrow \phi =_{def.} \neg \uparrow \neg\phi$
c. $\Uparrow^* \phi =_{def.} \neg \uparrow^* \neg\phi$ and $\Downarrow^* \phi =_{def.} \neg \downarrow^* \neg\phi$
d. $s_0 =_{def.} \Uparrow \bot$ and $t =_{def.} \Downarrow \bot$

The wffs of $L^{T_S}(VAR)$ are interpreted on finite ordered binary trees. The satisfaction relation is defined as follows (for the base clause see below (7) and (11)).

(5) a. $M, n \models \neg\phi$ iff $M, n \not\models \phi$
b. $M, n \models \phi \wedge \psi$ iff $M, n \models \phi$ and $M, n \models \psi$
c. $M, n \models \downarrow^1 \phi$ iff $\exists n'(R_{\succ_1}(n, n') \wedge M, n' \models \phi)$
d. $M, n \models \downarrow^2 \phi$ iff $\exists n'(R_{\succ_2}(n, n') \wedge M, n' \models \phi)$
e. $M, n \models \uparrow \phi$ iff $\exists n'(R_{\prec}(n, n') \wedge M, n' \models \phi)$
f. $M, n \models \downarrow^* \phi$ iff $\exists n'(R_{\succ^*}(n, n') \wedge M, n' \models \phi)$
g. $M, n \models \uparrow^* \phi$ iff $\exists n'(R_{\prec^*}(n, n') \wedge M, n' \models \phi)$
h. $M, n \models < \phi$ iff $\exists n'(R_{<}(n, n') \wedge M, n' \models \phi)$
i. $M, n \models > \phi$ iff $\exists n'(R_{>}(n, n') \wedge M, n' \models \phi)$

The clauses for the defined constants s and t are $M, n \models s_0$ iff $n = root_S$ and $M, n \models t$ iff $n \in \Theta_S$ from which it follows that they are true only at the root node and at terminal nodes, respectively.

The wffs of L^F are defined as follows. The set L^F_{wfff} is the smallest set such that (i) $a \in L^F_{wff}$ for each $a \in A$ (i.e., the set of attributes functions as the set of propositional variables for L^F, (ii) if ϕ, $\psi \in L^F_{wff}$, then $\neg\phi, \phi \wedge \psi \in L^F_{wff}$ and (iii) if $\phi \in L^F_{wff}$, then $< f > \phi \in L^F_{wff}$ for each $f \in F$, i.e., the elements of F function as modal operators. The choice of both A and F depends on the application as well as on the linguistic theory that is used. For the present purpose, A will be assumed to contain at least s, np, vp and vp' which are used to express purely categorial information about nodes, nom, dat and acc, which are the values of the feature $CASE$ as well as pl and sg, that are admissible values for the NUM feature. Thus, A can be taken as the union of the label sets $Cat = \{s, np, vp, vp'\}$, $Case = \{nom, dat, acc\}$ and $Num = \{sg, pl\}$. The set F contains at least the features $CASE$ and $NUM(BER)$. An example of a wff of L^F is $np \wedge < CASE > acc \wedge < NUM > sg$.

Elements of L^F_{wff} are interpreted on feature structures **FS**. The satisfaction relation is given in (6).

(6) a. $M, w \models a$ iff $w \in Q_a$ for $a \in A$
b. $M, w \models \neg\phi$ iff $M, w \not\models \phi$
c. $M, w \models \phi \wedge \psi$ iff $M, w \models \phi$ and $M, w \models \psi$
d. $M, w \models < f > \phi$ iff $\exists w'(R_f(w, w') \wedge M, w' \models \phi)$

The injective functional relation R_{z_f} with domain N_S assigns to each node $n \in N_S$ a feature structure $R_{z_f}(n) = \mathbf{FS}_n$. The image of R_{z_f} is the set $\{\mathbf{FS}_n \mid n \in N_S\}$ which is an element of the LDG structure defined above. The composition of R_{z_f} with the function γ assigns to each node $n \in N_S$ the root w_{0_n} of the feature structure $\mathbf{FS}_n$ assigned to n by R_{z_f}: $[\gamma \circ R_{z_f}](n) = w_{0_n}$.

In order to be able to talk about LDG structures for verbs, i.e. about the combined ontology, it is necessary to combine the languages with each other. This means that it must be possible to express in at least one language L constraints that are evaluated with respect to elements of structures with respect to which formulas from L are not evaluated. The languages will be combined at the level of atomic wffs of L^{T_S}. Instead of building L^{T_S} on top of a set VAR of propositional variables, it is built in terms of a set of structured atomic formulas, namely the wffs of $L^{T_{TS}}$ and L^F. In contrast to Blackburn et al. (1993), the wffs ϕ of $L^{T_{TS}}$ and L^F are not directly taken as wffs of L^{T_S}. Rather, the formulas will be of the form $< z_{ts} > \phi$ and $< z_f > \psi$ for ϕ a wff of $L^{T_{TS}}$ and ψ a wff of L^F. Thus, VAR is taken to be the union of the sets $\{< z_{ts} > \psi \mid \psi \in L^{T_{TS}}_{wff}\}$ and $\{< z_f > \phi \mid \phi \in L^F_{wff}\}$ such that syntactically the propositional variables $p \in VAR$ are replaced by formulas of the form $< \alpha > \phi$ as defined above. The modalities z_{ts} and z_f are interpreted in the LDG structure by one of the binary relations that link the tree structure $\mathbf{T}_S$ to the tree structure $\mathbf{T}_{TS}$ and one of the feature structures $\mathbf{FS}_n$, respectively. The base clause in the definition of satisfaction for z_f is given in (7).

(7) $M, n \models < z_f > \phi$ iff $\exists \mathbf{FS}[R_{z_f}(n, \mathbf{FS})\ (= \mathbf{FS}_n) \wedge M, \gamma(\mathbf{FS}) \models \phi]$

Following Blackburn et al. (1993) and Blackburn/Meyer-Viol (1997), the resulting language, $L^{T_S}(L^{T_{TS}}, L^F)$, will be called the language L^{T_S} *layered over* the languages $L^{T_{TS}}$ and L^F. Thus, a language with three 'layers' is used. The top layer are the formulas from L^{T_S} as defined above. They move us around the syntactic tree and are used to talk about structural (or configurational) properties of the tree. The base

layer are formulas of the form $< \alpha > \phi$ with $\alpha = z_{ts}$ or $\alpha = z_f$. Formulas $< z_f > \phi$ with $\phi \in L^F_{wff}$ express both syntactic and morphological information about a node n: what is the syntactic category of n?; what are the values of features like $CASE$ and $NUM(BER)$? They are used to talk about the 'internal' structure of nodes.
In order to be linguistically admissible, a feature decorated tree must satisfy a number of constraints of how atomic information is distributed. One prominent example of a constraint are phrase structure rules which constrain the distribution of purely categorial information. A typical example is $S \rightarrow NP\ VP$ which says that a local tree with root node of sort s is admissible if it immediately dominates a left node of sort vp and if it immediately dominates a right node of sort np. Formally, they impose restrictions on $R_{\succ_1}$ and $R_{\succ_2}$, i.e. the 'immediately left (right) dominates' relation. In $L^{T_S}(L^{T_{TS}}, L^F)$ this rule is expressed by the $L^{T_S}(L^F)$ formula $< z_F > (s \rightarrow \downarrow^1 np \wedge \ \downarrow^2 vp)$ $(= < z_F > s \rightarrow < z_F > \downarrow^1 np \wedge \ < z_F > \downarrow^2 vp$. This constraint concerns the distribution of purely categorial information and can be given the form $< z_f > (\phi \rightarrow \psi)$ for $\phi, \psi \in L^F_{wff}$. The structural information, expressed by pure L^{T_S} formulas, plays no role. The constraints on the relation between the syntactic and the morphological component imposed by LDG are of the form $< z_f > p \wedge \ t \wedge \alpha(< z_f > p \wedge t)$ $(p \in Cat,\ \alpha \in \{<, >\})$, i.e., the antecedent ϕ is not a pure L^F formula but expresses both purely categorial information and structural information about the position of the node in the tree in relation to its categorial classification. Antecedents of this form require that terminal nodes n $(n \in \Theta_S)$ of sort p satisfy the information expressed by the succedent ψ $(= < z_f > \phi)$, depending on their position among those elements of Θ_S that are also of sort p. An antecedent of the form $< z_f > p \wedge \ t \wedge \alpha$ for $\alpha = \ < (< z_f > p \wedge t)$ or $\alpha = \ > (< z_f > p \wedge t)$ expresses information about the order of arguments of the verb that are of sort p. As the constraint on the categorial sort of the terminal node expresses purely syntactic information, it follows that particular syntactic information at one place of the tree $\mathbf{T}_S$ requires some information to be present at that node too. In LDG this information concerns the value of the $CASE$ feature and therefore morphological information. For a three place verb like 'give' the constraints are given in (8).

(8) a. $< z_f > np \ \wedge t \wedge \neg < (< z_f > np_t \wedge t) \rightarrow < z_f >< CASE > nom$

b. $< z_f > np \wedge t \wedge \ < (< z_f > np \wedge t) \wedge \ > (< z_f > np \wedge t) \rightarrow < z_f >< CASE > dat$

c. $< z_f > np \wedge t \ \wedge \neg > (< z_f > np \wedge t) \wedge \ < (< z_f > np \wedge t) \rightarrow < z_f >< CASE > acc$

The constraints in (8) only hold if the argument positions are structural, otherwise there are exceptions to the constraints in (8b) and (8c). An example for (8b) are verbs with double accusative like 'lehren' ('teach'), 'Ich lehrte einen Studenten Semantik' ('I taught a student semantics'), where the intermediate argument is assigned accusative and not dative case, as predicted by (8b). The constraints in (8) must then be made conditional on the fact that additional information that is introduced by the verb does not hold. If this information is present, other constraints are effective and linking is no longer structural (see Naumann/Latrouite (2000) for details).

4 The SF-and the TS-Level

As was said in section (2), in LDG the SF of a verb v is cast in terms of an underlying event ontology. Yet, no formal definitions of the atomic SF-predicates and the connective & are given. The formalization of the SF-level that will be presented is based on *Dynamic Event Semantics* (DES). DES is based on the intuition that non-stative verbs express changes, Naumann (1999). A change is either an object (event) or a transformation of states (TS). Changes as objects bring about results by transforming a state s at which a result does not hold into a state s' where the result holds. This double perspective on the intuitive notion of a change is modeled in DES by having both an eventuality-structure $\mathbf{E}$ with an underlying domain E of events and a transition-structure $\mathbf{S}$ with an underlying domain S of states that is linearly ordered by $<_S$. Each event $e \in E$ is assigned its execution sequence, that is a convex subset of S. The domain E is sorted by the label set $VERB$: for $v \in VERB$, $P_v \subseteq E$ is the set of all events of type v, e.g. the set of all eating events if $v = eat$.
An event e can bring about more than one result (or TS) such that the relationship between an event and a TS is not one-one but one-many. P_v determines for each of its elements a set $Res(P_v, e)$ of results that e can possibly bring about. These results can be distinguished by the way they are evaluated on the execution sequence of e: (i) a result Q is *minimal* if it holds at all states of the execution sequence of e that are end points of initial stages e' of e that are of type P_v and (ii) a result Q is *maximal* if it only holds at the end point of e (if it holds at all on e's execution sequence). Results that are both minimal and maximal are called *w-maximal*, whereas results that are minimal without being maximal are called *w-minimal* and those that are maximal but not minimal *s-maximal*.
For an event of type 'John eat a fish' examples are: w-minimal: part of the fish is in John's stomach; s-maximal: (the mass of) the fish is in John's stomach; the fish does not exist. These results are (temporally) ordered: first the w-minimal results are brought about and then the s-maximal ones. It is therefore possible to define an ordering $\leq_v$ on $Res(P_v, e)$: $Q \leq_v Q'$ if Q is brought about not later than Q' on the execution sequence of e. Two results Q and Q' that are indistinguishable according to $\leq_v$ can be distinguished by implication relations. This gives rise to the more fine-grained ordering $\leq_v^*$: $Q_1 \leq_v^* Q_2$ iff $Q_1 <_v Q_2 \vee [Q_1 =_v Q_2 \wedge Q_1 \subseteq Q_2]$. Thus, for results that are temporally equivalent, implied results are 'greater' than those which imply them. Implied results are indicated by the subscript 2 whereas implying ones by the subscript 1. For the examples above using $\leq_{eat}^*$ yields the ordering w-minimal $\leq_{eat}^*$ s-maximal$_1$ $\leq_{eat}^*$ s-maximal$_2$.
The set $Res(P_v, e)$ can be partitoned by the equivalence relation 'is $\approx_v^*$-equivalent to': $[Q_T]_{v,e} = \{Q \mid Q \in Res(P_v, e) \wedge T(P_v, Q, e)\}$ for T a type of result. The ordering $\leq_v^*$ induces a corresponding ordering on these equivalence classes. At the level of v, the equivalence classes are defined as follows: $[Q_T]_v = \{Q \mid \exists e[e \in P_v \wedge Q \in [Q_T]_{v,e}]\}$.
Verb-classes are defined in terms of types of results. E.g., in English Accomplishment-verbs like 'eat' determine all three types of results whereas activity-verbs like 'push' only determine a w-minimal result. Transfer-verbs like 'give' determine only w-maximal results. The results assigned to an event $e \in P_v$ are always brought about with respect to objects that participate in e. E.g., if e is of type 'John eat a fish', the results are brought about with respect to John and the fish. John is assigned both the w-minimal result (part of the fish is in his stomach) and the s-maximal$_1$ one ((the

whole mass of) the fish is in his stomach). These results are also assigned to the fish which is, in addition, assigned the s-maximal$_2$ result (the fish no longer exists). For an event e of type 'John give the book to Mary' there are two w-maximal results: 'Mary does have the book' (w-max$_1$) and 'John does not have the book' (w-max$_2$) which yields the assignment: w-max$_1$: Mary, the book; w-max$_2$: John, the book. The relationship between an event $e \in P_v$, an object d and a result Q that e can bring about with respect to d is captured by a function Δ_v: $\Delta_v(e)(d)(Q)$ holds just in case Q is a result that e can possibly bring about with respect to d.

The Level of TS To each result Q from $Res(P_v, e)$ there corresponds a subset of e's participants. These sets can be defined using Δ_v and the equivalence classes defined by $O_{i_{v,e}} = \{d \mid \exists Q[\Delta_v(e)(d)(Q) \wedge Q \in [Q_i]_{v,e}]\}$ ($[Q_i]_{v,e}$) is the i-th equivalence class). The $O_{i_{v,e}}$ need not be singletons, as the examples above have shown. Yet, as shown in Naumann (2000), each participant of e can be uniquely singled out in terms of a particular result Q from $Res(P_v, e)$ plus a maximality or minimality condition with respect to Q. It is therefore possible to define a correspondence between participants d of e and the set $Res(P_v, e)$ ordered by $\leq_v^*$. From the temporal ordering of the results it follows that in addition it is possible to decompose e in such a way that each element of $Res(P_v, e)$ is assigned to exactly one element of the decomposition. The decomposition is defined in terms of the composition relation C. For a given verb v, let j be the number of results $Q \in Res(P_v, e)$ that are not $\approx_v^*$-equivalent. Then, for each $e \in P_v$ a decomposition T_j is defined. For $j = 3$ the definition is $T_3(e) = \{\{e_1, e_2, e_3\} \mid \exists e_{21} \exists e_{321}[C(e_{21}, e_1, e_2) \wedge C(e_{321}, e_{21}, e_3) \wedge e = e_{321}]\}$. In terms of T_3 the following equivalence classes are defined.

(9) a. $[e_1]_e = \{e_1 \mid \exists T_j[T_j(e) \wedge e_1 \in T_j(e) \wedge \forall e'[e' \in T_j(e) \to e_1 \leq_E e']]\}$
b. $[e_2]_e = \{e_2 \mid \exists T_j[T_j(e) \wedge e_2 \in T_j(e) \wedge \exists e'[e' \in T_j(e) \wedge e' \leq_E e_2] \wedge \exists e'[e' \in T_j(e) \wedge e_2 \leq_E e']]\}$
c. $[e_3]_e = \{e_3 \mid \exists T_j[T_j(e) \wedge e_3 \in T_j(e) \wedge \forall e'[e' \in T_j(e) \to e' \leq_E e_3]]\}$

The relation $\leq_E$ is defined as: $e \leq_E e'$ iff $\omega(e) \leq_S \alpha(e')$. As the elements of an T_j-decomposition of an $e \in P_v$ are linearly ordered, $T_j(e)$ can be taken as an n-tuple $< e_1 ... e_j >$ such that $\omega(e_i) = \alpha(e_{i+1})$ for $1 \leq i < j$; $\alpha(e) = \alpha(e_1)$ and $\omega(e) = \omega(e_j)$. Furthermore, to each such n-tuple there exists a second n-tuple $< e_1^* ... e_j^* >$ such that each e_i^* is an initial stage of e with e_i as final stage (suffix). The e_i^* are called the *initial extensions* of the e_i. Together, they are a T_j^*-decomposition of e. For T_j^*-triples a precedence ordering is defined by $e_i^* \leq e_{i'}^*$ iff $R_B(e_i, e_{i'})$. The definitions of a T_j decomposition and of the corresponding equivalence classes are lifted to the level of P_v as follows: $T_{j,v} = \{E' \mid \exists e[e \in P_v \wedge E' = T_j(e)]\}$ and $[e_i]_v = \{e \mid \exists e' \exists T[e' \in P_v \wedge T(e') \wedge e \in [e_i]_{e'}]\}$.
Each argument position α of the TS of a verb v uniquely corresponds to a pair $< [e_i]_v, [Q_i]_v >$ in the sense that if an $n+1$ tuple $< d_1, ..., d_n, e >$ belongs to the interpretation of the verb v, then there is a T_j decomposition $< e_1, ..., e_j >$ of e with $j = n$ such that (i) $e_i \in [e_i]_v$, (ii) the result $Q \in Res(P_v, e)$ corresponding to e_i is an element of $[Q_i]_v$ and (iii) Q is the result characterizing d_{n-i+1} relative to e. The argument positions determined by a verb v can therefore be identified with the corresponding pairs $< [e_i]_v, [Q_i]_v >$. κ is a function that assigns to P_v the n-tuple of

pairs $< [e_i]_v, [Q_i]_v >$. The TS of v can then be defined as a tree the leafs of which are the pairs $< [e_i]_v, [Q_i]_v >$.
Note that the relationship between the $[e_i]_v$ and the $[Q_i]_v$ need not be a one-one correspondence because equivalence classes $[Q_i]_v$ and $[Q_j]_v$ the elements of which are $\approx_v$- but not $\approx_v^*$-equivalent are assigned the same equivalence class $[e_k]_v$. The nodes N_{TS} of $\mathbf{T}_{TS}$ are pairs of equivalence classes of events and results. The non-terminal nodes are generalizations of the terminal nodes. One constraint that the set of terminal nodes Θ_{TS} must satisfy is that their order is the inverse of the order determined by κ. This constraint is defined in terms of the leaf relation $R_{leaf_{TS}}$.

(10) a. $(\alpha, \sigma) \in R_{leaf_{TS}}$ iff (i) $\sigma = < p_1, ..., p_n >$, (ii) $R_{\succ^*_{TS}}(\alpha, p_i)$, (iii) $p_i \in \Theta_{TS}$ for $1 \leq i \leq n$ and (iv) $R_{>}(p_i, p_j)$ for $1 \leq i < j \leq n$
b. if $\kappa(P_v) = < p_1, ..., p_j >$, then $R_{leaf_{ts}}(root_{TS}) = < p_j, ..., p_1 >$

As language for talking about $\mathbf{T}_{TS}$ L^{T_S} can be chosen, i.e. $L^{T_S}(VAR)$ without the base clauses. That means that there is no atomic, i.e. non-structural, information associated with the nodes of $\mathbf{T}_{TS}$. This is a consequence of the fact that in the present context only structural linking is considered for which only the purely structural properties of TS are relevant (see section (2) for this distinction and Naumann/Latrouite (2000) for an alternative formalization of the notion of TS in LDG in which both structural and non-structural linking are analyzed). The strategy of how L^{T_S} and $L^{T_{TS}}$ are combined was already described above. The base clause is (11).

(11) $M, n_s \models < z_{ts} > \phi$ iff $\exists n_{ts}[R_{z_{ts}}(n_s, n_{ts}) \wedge M, n_{ts} \models \phi]$

The relation between $\mathbf{T}_S$ and $\mathbf{T}_{TS}$ is defined in terms of R_{leaf_S}, defined in (12a), and $R_{leaf_{TS}}$. Recall from section (2) that in LDG the default word order (syntactic level) must be the inverse of the argument position in TS which are realized by the argument NPs. In the present context this constraint amounts to the requirement that given the k-tuple of terminal nodes $< n_1, ..., n_k >$ of sort np that is the value of R_{leaf_S}, each n_i belonging to this tuple is mapped to the $k - i + 1$-th element of the k-tuple $< p_1, ..., p_k > (= R_{leaf_{TS}})$ of leafs of $\mathbf{T}_{TS}$.
The mapping is defined as the partial function $R_{z_{ts}}$ from N_S to Θ_{TS}, (12b). As the value of $R_{leaf_{TS}}$ is the inverse of $\kappa(P_v)$, the mapping can equally be defined in terms of κ, (12c). This way of defining the mapping will simplify expressing the constraint in L^{T_S}.

(12) a. $(\alpha, \sigma) \in R_{leaf_S}$ iff (i) $\sigma = < n_1, ..., n_m >$, (ii) $R_{\succ^*_S}(\alpha, n_i)$, (iii) $n_i \in \Theta_S$, (iv) $n_i \in \gamma(R_{z_F}(n_i))$ for $1 \leq i \leq m$ and (iv) $R_{<_S}(n_i, n_j)$ for $1 \leq i < j \leq m$
b. $R_{z_{ts}}(n, p)$ iff $n = \pi^i(R_{leaf_S}(root_S))$ and $p = \pi^{k-i+1}(R_{leaf_{TS}}(root_{TS}))$ for some i such that $1 \leq i \leq k$ and $k =$ the length of $R_{leaf_S}(root_S) = R_{leaf_{TS}}(root_{TS})$
c. $R_{z_{ts}}(n, p)$ iff $n = \pi^i(R_{leaf_S}(root_S))$ and $p = \pi^i(\kappa(P_v))$ for some i such that $1 \leq i \leq k$ and $k =$ the length of $R_{leaf_S}(root_S)$

$R_{z_{ts}}$ differs from R_{z_f} in the way conditions are imposed on the value of the function for a given argument. Whereas R_{z_f} constraints the value of a particular feature, namely the $CASE$ feature, no corresponding condition is imposed by $R_{z_{ts}}$. It constrains only the position of the value and does not impose any requirement on what has to hold at that value. This is reflected in the way the constraint is expressed in L^T: $< z_f > np \wedge t \wedge \alpha \rightarrow < z_{ts} > (\alpha \wedge t)$, where α is a formula expressing information about the relative order of the terminal node at which the whole formula is required to be true among the terminal nodes that are of sort np too, similarly to the constraints expressed in (8) ($\alpha =. \neg < (< z_f > np_t \wedge t)$ or $\alpha =< (< z_f > np \ \wedge t) \wedge \ > (< z_f > np \wedge t)$ or $\alpha = \neg > (< z_f > np \wedge t) \wedge \ < (< z_f > np \wedge t)$). This constraint imposes the constraint that a terminal node of sort np which is at a particular position among the terminal nodes of the same sort (expressed by a) is mapped to a terminal node in the tree spawned by $\kappa(P_v)$ that has the same position in relation to the terminal nodes of $\kappa(P_v)$. From the fact that the leafs of $\kappa(P_v)$ are inverse order to those of $\mathbf{T}_{TS}$ this amounts to the requirement that the default word order determined by v is the inverse of the order of arguments in TS. Note that the antecedents of the constraints in (8) and of the above scheme are identical. Thus, if a constraint in (8) and the corresponding instantiation of the above scheme are required to hold at a node n of $\mathbf{T}_S$, one gets the equivalence $M, [\gamma \circ R_{z_f}](n) \models \ < CASE > \beta$ iff $M, R_{z_{ts}}(n) \models \alpha$ (where β is a variable over atomic values for which the $CASE$ feature is defined). Linguistically, this equivalence requires a position in TS (expressed by α) to be related to a particular assignment of case to an argument-NP realizing this position.

The Level of SF If an event $e \in P_v$ occurs that is denoted by v in the lexicon, e can be decomposed into a $T_j = \{e_1, ..., e_j\}$, for $j =$ the number of results from $Res(P_v, e)$ that are not $\approx_v$-equivalent, such that e_1 brings about the results from $[Q_1]_e$ and no results from equivalence classes following $[Q_1]_e$, e_2 brings about the results from $[Q_2]_e$ and no result from a $[Q_k]_e$, $2 < k \leq j$, that follows $[Q_2]_2$ in the ordering $\leq_v$ (such that at the end of e_2 Q_1 and Q_2 have been brought about) and so on until finally the results from $[Q_j]_e$ are brought about by e_j. Furthermore, as shown above, to each $e_i \in T_j(e)$ there corresponds the largest prefix e' that is an initial stage of e and of which e_i is a final stage. Recall that the SF of a verb v is of the form $[\alpha_1 \ \& \ ... \ \& \ \alpha_n](e)$. Each SF predicate α_i can be interpreted with respect to the initial extension of the i-th event in $T_j(e)$. The α_i can be taken to express that the results from $[Q_i]_e$ corresponding to e_i is brought about.
A suitable formalism for expressing these relationships is (an extension of) *Dynamic Modal Arrow Logic* (DMAL), Naumann/Osswald (1999). DMAL combines Dynamic Modal Logic and Arrow Logic, van Benthem (1991). As shown in Naumann/Osswald (1999), the way a result from $[Q_i]_e$ is evaluated on the initial extension of e_i is captured by a dynamic mode, i.e. a function that maps pairs of sets of events and sets of states to sets of events. In DMAL a verb v is translated by formulas of the form $\pi_1 \ \& \ ... \ \& \ \pi_n$ where each π_i is of the form $DM(\sqsubseteq_v, \phi)$ or $DM(\sqsubseteq_v, \phi) \cap DM'(\sqsubseteq_v, \psi)$ for DM, DM' a dynamic mode. The satisfaction clause for & is (13) (σ is a finite sequence of events; σ_ϵ is the empty sequence).

(13) $M, e, \sigma \models \pi \ \& \ \pi'$ iff there is an e_1 such that (i) $R_B(e_1, e)$; (ii) $R_B(e_1, last(\sigma)) \vee \sigma = \sigma_\epsilon$; (iii) $M, e_1, \sigma \models \pi$ and (iv) $M, e, < \sigma \mid e_1 > \models \pi'$

(13) is an instance of a translation scheme because the second argument of the DM are dependent on a particular event $e \in P_v$. The translation of v in DMAL is therefore not unique. Yet, for a given $e \in P_v$ one has the following. Let $\pi_1 \ \& \ ... \ \& \ \pi_n$ be a DMAL translation of v. Then $M, e, \sigma \models \pi_1 \ \& \ ... \ \& \ \pi_n$ iff there is a T_n decomposition $< e_1, ..., e_n >$ of e such that (i) $R_B(e, e_i)$, $1 \leq i \leq n$, (ii) $R_B(e_i, e_j)$, $1 \leq i < j \leq n$, and (iii) $M, e_i, < \sigma, < e_1, ..., e_{i-1} >> \models \pi_i$. The tuple $< e_1, ..., e_n >$ need not be

unique. Rather, they are a particular T_n decomposition of e. What is common to all T_n decompositions that make π_1 & ... & π_n true is captured in terms of restrictions of the equivalence classes $[e_i]_{v,e}$ induced by T_n according to what has been said at the beginning of this section: $[e_i^*]_{v,e} = \{e' \mid e' \in [e_i]_{v,e} \wedge \forall Q[Q \in [Q_i]_{v,e} \rightarrow [\omega(e') \in Q \wedge . \forall Q'[Q' \in Res(P_v, e') \wedge Q <_v Q' \rightarrow \omega(e') \notin Q']]]\}$. The $[e_i^*]_{v,e}$ are the leafs of a tree which are labeled by formulas from L^{DMAL}. A language to talk about this tree is $L^{DMAL}(L^{T_{TS}})$. This language is L^{DMAL} augmented by formulas of the form $< z'_{ts} > \phi$ for ϕ an $L^{T_{TS}}$ well-formed formula. The atomic information associated with the nodes are subformulas of π_1 & ... & π_n. For the i-th leaf it is π_i. The trees T_{TS} and $\{T_{SF_e} \mid e \in R_v\}$ are related by a function R_{SF} with domain the nodes of the trees in $\{T_{SF_e} \mid e \in R_v\}$, i.e. $R_v \cup \{e \mid \exists e'[e' \in R_v \wedge R_B(e, e')]\}$ ($R_v = \{e \mid e \in [\![Tr(v)]\!]\}$ is the set of all events $e \in P_v$ denoted by v in the lexicon; $Tr(v)$ is the set of translations of v) and range N_{TS} that assigns to each e in the domain a node in T_{TS}.

5 Conclusion

In LDG a verb is analyzed at the grammatical levels of Semantic Form, Theta-Structure and the syntactic-morphological level. In the formalization presented in this paper each level is modeled by a mathematical structure. Constraints between the different levels that restrict the distribution of information are defined in terms of mappings that relate the domains of the structures with each other. The constraints are formulated by *fibring* or *layering* the different languages. The base clauses are modal formulas of the form $< \alpha > \phi$, where α is interpreted as the relation R_α that links elements of two domains. E.g., the constraint between the syntactic and the morphological level concerns the (default) constituent order and the assignment of case. For structural linking, particular positions in the order of constituents are related to particular cases. The (default) constituent order is represented by (the linear ordering of) the terminal nodes of the tree structure $\mathbf{T}_S$. The morphological (case) information is encoded in the feature structure corresponding to a terminal node (of category np). The constraint therefore requires that if a terminal node n of category np satisfies ϕ, expressing information about the position of n in the constituent order, then the corresponding feature structure must satisfy ψ, expressing information about the case of the constituent: $M, n \models \phi \Rightarrow M, \gamma[R_{z_f}(n)] \models \psi$, where R_{z_f} maps an $n \in N_S$ to its corresponding feature structure and $\gamma(\mathbf{FN})$ is the root of $\mathbf{FN}$.

References

[1] Blackburn, P. et al. (1993). Talking about Trees, in: *Proc. 6th Conference of the European Chapter of the ACL*, pp. 21-29

[2] Blackburn, P./Meyer-Viol, W. (1997). Modal Logic and Model-Theoretic Syntax, in: M. de Rijke (ed.): *Advances in Intensional Logic*, Kluwer Academic Publishers, pp. 29-60

[3] Blackburn, P./de Rijke, M. (1997). Zooming In, Zooming Out, *JoLLI*, vol.6, pp. 5-31

[4] Naumann, R. (1999). A Dynamic Approach to Aspect: Verbs as Programs, to appear in *Journal of Semantics*

[5] Naumann, R. (2000). Events and Objects, to appear *Proc. 7th Symposium on Logic and Language*, Budapest, Hungary

[6] Naumann, R./Latrouite A. (2000). Linking and the Dynamic Structure of Verbs: Underspecification at the morphological-semantic Interface, Obrst, L. (ed.): *Proceedings of the Workshop on Semantic Approximation, Vagueness and Granularity*, AAAI Publications

[7] Naumann, R./Osswald, R. (1999). A Dynamic Modal Arrow Logic for the Analysis of Aspectual Phenomena in Natural Language, to appear in *Proceedings 3rd International Tbilisi Symposium on Logic, Language and Computation*

[8] Pustejovsky, J. (1991). The generative Lexicon, *Computational Linguistics*, vol.17, pp.409-441

[9] van Benthem, J. (1991). *Language in Action*, Amsterdam, North-Holland

[10] Wunderlich, D. (1997). CAUSE and the Structure of Verbs, *Linguistic Inquiry*, vol. 28, pp.27-68

A New Spectrum of Recursive Models

ANDRE NIES, *Department of Mathematics, The University of Chicago, 5734 S. University Avenue, Chicago IL 60637, USA.*
E-mail: nies@math.uchicago.edu

Abstract

We describe a strongly minimal theory S in an effective language such that, in the chain of countable models of S, only the second model has a computable presentation. Thus there is a spectrum of an ω_1–categorical theory which is neither upward nor downward closed. We also give an upper bound on the complexity of spectra.

Keywords: Strongly minimal, ω_1–categorical, spectrum of recursive models

1 Introduction

Our main purpose is to find a strongly minimal theory in an effective language whose spectrum of recursive models is the set $\{1\}$. We rely on some concepts in Khoussainov, Nies, and Shore [2], reviewed here briefly. Baldwin and Lachlan [1] showed that the countable models of an ω_1–categorical theory T form an $\omega+1$–chain $M_0(T) \prec M_1(T) \prec \ldots \prec M_\omega(T)$ under elementary embeddings. In [2], we defined the spectrum of computable models of T,

$$\mathrm{SRM}(T) = \{i \leq \omega : M_i(T) \text{ has a computable presentation}\}.$$

We gave an example of an ω_1–categorical (in fact, strongly minimal) theory T such that $\mathrm{SRM}(T) = (\omega - \{0\}) \cup \{\omega\}$. Kudeiberganov [3], extending a result of Goncharov, proved that, for each $n \in \omega$, $n \geq 1$, there is an ω_1–categorical theory T such that $\mathrm{SRM}(T) = \{0, \ldots, n-1\}$. Here, in a priority construction, we combine the techniques used to prove the two results and obtain a strongly minimal theory T such that $\mathrm{SRM}(T) = \{1\}$. Thus, only $M_1(T)$ has a computable presentation (which we build in the priority construction).

The ultimate goal of these investigations is to describe all possible spectra of ω_1–categorical theories. In a sense, our example is the most complicated one found so far, since all the previous spectra were upward closed or downward closed in $\omega+1$. Before we proceed to the main result, we give an upper bound on the complexity of spectra. Many ω_1–categorical theories are model complete (for instance, ACF_0, or, more generally, each ω_1–categorical theory axiomatizable by Π_2–formulas, by Lindström's test), so we also give a tighter upper bound for such theories.

Proposition 1.1
Suppose T is ω_1–categorical theory in an effective language. Then
(i) $\mathrm{SRM}(T) \in \Sigma^0_3(\emptyset^\omega)$
(ii) If T is model complete, then $\mathrm{SRM}(T) \in \Sigma^0_4$.

PROOF. Suppose $\beta(x)$ is a strongly minimal formula for T in the sense of [1]. Choose an effective numbering of the set D of atomic relations and negations of atomic relations in the given effective language over the domain $\mathbb{N}$ (typical elements of D are $fn = fgm$ and $\neg Rnm$, where $n, m \in \mathbb{N}$, f, g are unary function symbols and R is a binary relation symbol in our language). If we view a c.e. set W as a subset of D, then W gives rise to a presentation of a computable model, provided that exactly one of an atomic relation or its negation is in W, and, if the language contains an equality symbol $\approx$, then $\{n, m : n \approx m \in W\}$ is an equivalence relation compatible with W. The numbers e such that W_e determines a presentation form a Π^0_2–set P. For $e \in P$, this computable presentation is denoted by $\mathcal{A}_e$.
In the following, "a.i." stands for "algebraically independent" and, for a structure $\mathcal{A}$ in our language, $\beta(\mathcal{A})$ denotes $\{a \in \mathcal{A} : \mathcal{A} \models \beta(a)\}$. Let S_k be the group of permutations of $\{1, \dots, k\}$.
To prove (i), we can suppose that $T \leq_T \emptyset^\omega$, otherwise $\mathrm{SRM}(T) = \emptyset$. Now $n \in \mathrm{SRM}(T) \Leftrightarrow \exists e \in P$

$$\mathcal{A}_e \models T \ [\Pi^0_1(\emptyset^\omega)] \ \& \tag{1.1}$$

$$\exists a_1, \dots, a_n \in \beta(\mathcal{A}_e)[(a_1, \dots, a_n) \text{ a.i.}] \ \& \tag{1.2}$$

$$\neg \exists a_1, \dots, a_{n+1} \in \beta(\mathcal{A}_e)[a_1, \dots, a_{n+1} \text{ a.i.}] \tag{1.3}$$

Also, in $\mathcal{A}_e$, $c_1, \dots, c_k$ are a.i. iff for all formulas $\forall \varphi(x_1, \dots, x_k)$

$$\forall \pi \in S_k[\mathcal{A}_e \models \varphi(c_{\pi(1)}, \dots, c_{\pi(k)}) \Rightarrow \exists^\infty c \ \mathcal{A}_e \models \varphi(c_{\pi(1)}, \dots, c_{\pi(k-1)}, c)],$$

which is $\Pi^0_1(\emptyset^\omega)$. Therefore (1.2) is $\Sigma^0_2(\emptyset^\omega)$, (1.3) is $\Pi^0_2(\emptyset^\omega)$ and the whole expression is $\Sigma^0_3(\emptyset^\omega)$, as desired.
For (ii), if T is model complete, then each formula α is equivalent under T to a Σ_1–formula σ with the same free variables. Moreover, by a recursive procedure using $T \cap \Pi_2$ as an oracle (the set of Π_2–sentences in T), we can determine such a σ from α. If T has a recursive model, then $T \cap \Sigma_1$ is c.e. and $T \cap \Pi_2$ is Π^0_2. Thus $T \leq_T \emptyset''$. Now, in the expression above, $\mathcal{A}_e \models T$ becomes Π^0_3. Moreover, since we can assume that all formulas involved are Σ_1, "$c_1, \dots, c_k$ a. i." becomes Π^0_2, (1.2) becomes Σ^0_3 and (1.3) Π^0_3. ∎

2 $\{1\}$ is a spectrum

THEOREM 2.1
There is a strongly minimal (and hence ω_1–categorical) theory T in an effective language such that $M_i(T)$ $(i \leq \omega)$ has a computable presentation iff $i = 1$.

PROOF. We use a language consisting of binary relations P_k $(k \geq 0)$ called *edge relations* and further relations L_e $(e \geq 0)$. T contains axioms saying that the relations do not depend on the order of the elements, and only can hold for distinct elements. Let $\mathcal{L}_P$ be first-order language over $\{P_k : k \geq 0\}$. The models of T restricted to $\mathcal{L}_P$ are, with a small notational change, as in [2]. They consist of a disjoint union of components C_i, D. C_0 is a singleton, and C_{n+1} is the union of two copies of C_n, where elements in different subcomponents are connected via a P_n–edge. We call the models C_i *complexes of dimension* i, or for short, *i-complexes*, which replace the

i–cubes in [2] to simplify notation. There are natural embeddings of an i–complex into an $i+1$–complex. The ∞–complex D is the union of a chain of an i–complex for each finite i.

We determine T by describing a recursive presentation of $M_1(T)$. However, as in [2], $T \cap \mathcal{L}_P$ can be axiomatized by saying for which $n \in \omega$ an n–complex exists, and that there is at most one for each n. As in [2], for an infinite set $S \subseteq \omega$, let $\mathcal{A}_S = \bigcup_{n \in S} C_n$ be the $\mathcal{L}_P$–structure consisting of exactly one n–complex whenever $n \in S$. Then $T \cap \mathcal{L}_p = \mathrm{Th}(\mathcal{A}_S)$ is ω_1–categorical, where $M_i(T \cap \mathcal{L}_p)$ $(i \leq \omega)$ consists of $\mathcal{A}_S$ and an ∞–complex for each $j < i$.

An axiomatization for the theory in the full language is obtained by specifying, in addition, first–order definitions for the relations L_e. This is needed to show that the full theory is ω_1–categorical. Actually, $T \cap \mathcal{L}_P$, and hence T, are strongly minimal, as the following proposition shows. ∎

Proposition 2.2

For each infinite $S \subseteq \omega$, $\mathrm{Th}(\mathcal{A}_S)$ is strongly minimal.

Proof. Suppose M is a countable model of $\mathrm{Th}(\mathcal{A}_S)$ and $D \subseteq M$ is definable from parameters $a_0, \ldots, a_{n-1} \in M$ using edge relations among $P_0, \ldots, P_k$, $k \in S$, with the intent of showing that D is finite or cofinite. Now $\widetilde{M} = M \restriction \{P_0, \ldots, P_k\}$ consists of at most k complexes of dimension $< k$ and infinitely many k–complexes. Let F be the union of the complexes of dimension $< k$ and all complexes containing some a_i. Then F is finite, and if $c, d \in M - F$, there is an automorphism of $\widetilde{M}$ taking c to d and fixing each parameter used to define D. Thus, if $D \not\subseteq F$, then $D \cup F = M$. ∎

We now describe the construction of a computable presentation for $M_1(T)$. The ∞-complex will be the complex containing 0 in this particular representation.

The construction is in stages. Each stage s has finitely many substages, denoted by the letters τ, σ, which are numbered 0,1,2, ... through the whole construction, independently of s. $M_{1,\tau}(T)$ is the model obtained by the end of stage τ and has as a domain an initial segment $[0, u) \subseteq \mathbb{N}$, $u \geq \tau$. At the end of any substage τ, D will denote the current complex containing 0. If x is already in the domain, $\dim_\tau(x)$ denotes the dimension of the complex x is in at stage τ (so that $\dim_\tau(0)$ is the current dimension of D). The distance of x, y in the domain of $M_{1,\tau}(T)$ is

$$\begin{aligned} d_\tau(x,y) &= 0 \text{ if } x = y \\ d_\tau(x,y) &= k \text{ if } P_{k-1}xy \ (k \text{ is unique}) \\ d_\tau(x,y) &= \infty \text{ if there is no such } k. \end{aligned}$$

A complex C_r which exists at substage τ will be isomorphic to the "ball" $\{x \in D : d_\tau(0,x) \leq r\}$.

During the construction, we may do one of the following: (a) add a new m–complex (whose domain consists of the least numbers not used before), or (b) merge an existing complex C_r into D, using a procedure *Merge*(C_r), which chooses k large, first expands C_r, D to complexes D', D'' of dimension $k-1$ and then connects all elements of D' with all elements of D'' via P_{k-1}. Thus, $\dim_\tau(x)$ can change at most once from a constant value to "unbounded", while $d_\tau(x,y)$ may change once from ∞ to a finite value. We denote the limit value of $\dim_\tau(x)$ by $\dim(x)$ and the limit value of $d_\tau(x,y)$ by $d(x,y)$.

We recall a further definition from [2].

DEFINITION 2.3
A function f is **limitwise monotonic** if there exists a recursive function $\varphi(x,t)$ such that $\varphi(x,t) \leq \varphi(x,t+1)$ for all $x,t \in \omega$, $\lim_t \varphi(x,t)$ exists for every $x \in \omega$ and $f(x) = \lim_t \varphi(x,t)$.

Let S be the set of dimensions of finite complexes in any model of T. In [2, Lemma 2.2] we show that, if the prime model $\mathcal{A}_S$ is recursive, then the set S is the range of a limitwise monotonic function.
Let $\varphi_e(x,t)$, $e \in \omega$, be a uniform enumeration of all partial recursive functions φ such that for all $t' \geq t$ if $\varphi(x,t')$ is defined, then $\varphi(x,t)$ is defined and $\varphi(x,t) \leq \varphi(x,t')$. To ensure $M_0(T) \restriction \mathcal{L}_P$ (and hence $M_0(T)$) has no computable presentation, we satisfy requirements N_i which imply that S is not the range of a limitwise monotonic function given by φ_i.

$$N_i : \exists x,t \ \varphi_i(x,t) \text{ undefined } \vee \ \exists x \ \lim_t \varphi_i(x,t) \notin S.$$

The last disjunct may be achieved by ensuring $\lim_t \varphi_i(x,t) = \infty$.
An N_i-strategy has a parameter $m = m(N_i)$, whose values are chosen in a decreasing way in the interval $[g(i), g(i+1))$, where $g(i) = \sum_{j<i} h(j)$ and $h(j)$ is a computable function bounding the possible number of injuries to the requirement N_j (see Lemma 2.5 below). It has also parameters x,t. All parameters may be undefined.
The N_i–strategy is as in the proof of the recursion theoretic lemma [2, Lemma 2.1], but here it is incorporated into the priority construction of a presentation for $M_1(T)$. First add an m-complex, for an approriate m. The "opponent" now has to provide x,t such that $\varphi_i(x,t) = m$. As a response, use x to drive the limit $\lim_{t'} \varphi_i(x,t')$ to infinity. To do so, remove an m'-complex whenever $\varphi_i(x,t') = m'$ for $t' > t$. (The m'–complex was created by N_i itself, in which case $m' = m$, or by a lower priority N-strategy still waiting for the opponent's first move, which is now injured.) In some more detail, the N_i–strategy is the following. If any of the cases below applies, take the corresponding action.

(N1) All parameters are undefined, and $g(i+1) \leq s$.
Action. Let m be the largest unused number in $[g(i), g(i+1))$. Perform the procedure *Expand*$(\emptyset, m)$, which creates a new complex of dimension m.

(N2) m is defined, but x,t are undefined, and now $\varphi_{i,s}(x,t) = m$ for some $x,t < s$.
Action. Choose x,t as values for the parameters. Call the procedure *Merge*(C_m), which puts C_m into D and thereby removes m from the list of possible values for $\lim_{t'} \varphi_i(x,t')$.

(N3) x,t are defined and now $\varphi_i(x,t') = m'$ for some $t' > t$, where currently an m'-complex $\neq D$ exists.
Action. Perform *Merge*$(C_{m'})$.

The requirements R_e code K into any presentation of a model $M_i(T)$, $i \geq 2$. By meeting the following requirements, we ensure that, if $e \notin K$, L_e is empty in each model of T, and if $e \in K$ then $L_e uv$ holds for any two algebraically independent elements of a model of T.
$R_e : \ e \in K \ \Rightarrow \ \exists n$

$$\forall x,y[dim(x) < g(e+1) \vee dim(y) < g(e+1) \vee d(x,y) < n \vee L_e xy]. \tag{2.1}$$

Since (2.1) can be expressed in a first-order way, and only the last alternative can occur for algebraically independent u, v, meeting all the requirements R_e is sufficient for the coding of K.

The R_e– strategy has a single parameter n_e, which is defined first at a stage s when $e \in K_s$ and may be made undefined finitely often by higher priority N–type strategies. The limit value will provide the witness n for (2.1). The R_e–strategy tries to ensure $L_e uv$ whenever n_e is defined, $\dim(u), \dim(v) \geq g(e)$ and $d(u,v) \geq n_e$. The priority ordering of the requirements is $N_0 \prec R_0 \prec N_1 \prec R_1 \prec \ldots$ Both types of requirements are *reset* by making all their parameters undefined.

Suppose an N-strategy wants to merge a complex $C_{m'}$ created by an N'–strategy into D, so that $N \prec N'$. This conflicts with the R_e–strategy in case we did not declare $L_e xy$ for all $x \in C_{m'}, y \in D$, since we will use an edge relation P_u, $u \geq n_e$ to connect x, y. It is too late now to add $L_e xy$, since we want a computable presentation of $M_1(T)$. This conflict is solved as follows: for all the requirements N' such that $R_e \prec N'$ and $C_{m(N')}$ exists, when e is enumerated into K, R_e first merges $C_{m(N')}$ into D. Only then R_e defines the first value of n_e, larger than all indices of edge relations used so far. If $N' \prec R_e$, before merging $C_{m(N')}$ into D we make n_e undefined, and R_e redefines it with large value after the merging took place.

The effect of R_e on the theory is described by a set F_e, which is cofinite if $e \in K$ and empty otherwise. Let $F_{e,0} = \emptyset$ and $F_{e,\tau} =$

$$F_{e,\tau-1} \cup \{k : n_e \text{ defined at substage } \tau \;\&\; k \geq n_e \;\&\; P_{k-1} \text{ first used at } \tau\}. \tag{2.2}$$

Let $F_e = \bigcup_\tau F_{e,\tau}$. We will verify that

$$M_1 \models L_e xy \Leftrightarrow \dim(x), \dim(y) \geq g(e+1) \;\&\; d(x,y) \in F_e \cup \{\infty\}, \tag{2.3}$$

which gives the desired first–order definition of L_e from finitely many edge relations. During a substage τ of the construction, we add $L_e xy$ to the presentation if y is a new element, $e \in K_{s-1}$ and (2.3) holds at that stage. Then presentation is computable. We need to verify that $L_e^{M_1(T)}$ actually satisfies (2.3), despite possible changes of $\dim(x), \dim(y)$ and $d(x,y)$ after τ.

We describe the procedures and the construction in detail. Whenever a procedure adds new numbers to the domain, they are chosen minimal in $\mathbb{N}$.

Expand(C_u, k) has as an input a u-complex C_u (recall that C_u is isomorphic to $\{x : d(x,0) \leq u\}$). It expands C_u to a k-complex by adding new elements and the appropriate P_{k-1}–relations between elements. We also include as a special case *Expand*$(\emptyset, k)$, which creates a new complex of dimension k. This is counted as one substage.

Merge(C_r) assumes that there is a complex C_r, $r = m(N)$ for some (unique) N. It merges C_r and D, but in a way that the overall goal that L_e be definable by (2.3) can be achieved. Let $N' \succ N$ be the requirement of highest priority such that $m(N')$ is defined. Recursively, call *Merge*$(C_{m(N')})$, using finitely many substages (if N' fails to exist, this step is vacuous). Next, in a single substage τ, perform the following:

1. Reset all the R-type requirements $\succ N$.
2. Choose k large and call *Expand*$(C_{r,\tau-1}, k-1)$, producing a complex D', and call *Expand*$(D_{\tau-1}, k-1)$ producing a complex D''.

3. Connect D', D'' by adding symmetric edges $P_{k-1}xy$, whenever x is in one and y in the other. This yields D_τ. Reset the requirement N.

The construction.
Stage 0. Let $D = \{0\}$, $\tau = 0$.
Stage $s > 0$. In the following, increase τ by one after each substage. Declare $L_e xy$ whenever an element y is added at a substage τ to the domain M_1 such that $e \in K_{s-1}$, $d_\tau(x,y) \in F_{e,\tau} \cup \{\infty\}$ and $\dim_\tau(x), \dim_\tau(y) \geq g(e+1)$.

a) Pick the requirement of highest priority U (if there is any) for which one of the following applies, and carry out the corresponding action,
   1. U is N_i, all parameters of N_i are undefined, and $g(i+1) \leq s$. *Action.* (N1) above.
   2. U is R_e, n_e was not defined up to now and $e \in K_s$. *Action.* Let $N \succ R_e$ be the N-type requirement of highest priority such that $m(N)$ is defined. If N exists, perform *Merge*$(C_{m(N)})$. Next, pick a large number $\geq g(e+1)$ as n_e.

b) If some u-complex exists and some N-type requirement desires to merge it via (N2) or (N3), perform *Merge*(C_u) for the minimal such u.

c) To ensure that $dim(D) \geq s$ at the end of stage s, call *Expand*$(D_{\tau-1}, \dim_\tau(0)+1)$

d) For all R_e such that n_e is now undefined but was defined before, redefine n_e with a large value.

The verification. We write M_i instead of $M_i(T)$.

LEMMA 2.4
The model M_1 is recursive.

PROOF. We want to test whether $M_1 \models Rxy$, where $x, y \in \mathbb{N}$ and R is a relation symbol from our language. We can suppose that $x < y$ and $y \in \mathrm{dom}(M_{1,\tau}) - \mathrm{dom}(M_{1,\tau-1})$ so that y is added at a substage τ of a stage s.
1. If R is P_k, then we distinguish two cases. If y is added by a procedure *Expand*, then $M_1 \models P_k xy \Leftrightarrow M_{1,\tau} \models P_k xy$. Otherwise $M_1 \models P_k xy$ because we connected D', D'' in a *Merge* procedure and $x \in D', y \in D''$, or we performed (c) of the construction during a stage $t \geq s$, in which case $k \geq s$ (since at each stage we introduce a new edge relation). Thus it suffices to check whether $M_{1,\max(k+1,s)} \models P_k xy$.
2. Now suppose R is L_e. Then, by the construction, $M_1 \models L_e xy \Leftrightarrow M_{1,\tau} \models L_e xy$ (since we determine whether $L_e xy$ holds when y is introduced). ∎

LEMMA 2.5
There is a computable function h such that N_i is reset at most $h(i)$ times. In particular, during the construction, there is always a sufficient supply of candidates for the parameter $m(N_i)$, and also R_i is reset only finitely many times.

PROOF. Let $h(0) = 0$. To determine $h(i+1)$, we observe that N_{i+1} can be reset at most two times before N_i is reset as well. For, if N_i is not reset, then either N_{i+1} was reset by R_i, which can only happen once, or during a *Merge* procedure for the sake of N_i. This means that before the merging, N_i has parameters x, t and $\varphi_i(x,t) = m(N_{i+1})$. By the way $m(N_{i+1})$ is chosen and since $\varphi_i(x,t)$ is nondecreasing in t, this can only happen once before $x(N_i)$ is changed.

Now, defining h recursively by $h(0) = 0$, $h(i+1) = 3h(i) + 3$, we obtain the desired bound. ∎

LEMMA 2.6
The requirements N_i are met. Hence $M_0(T)$ has no computable presentation.

PROOF. Suppose that N_i is not reset from stage s_0 on. Then N_i has permanently highest priority from s_0 on, and can therefore always fulfill its desire to create a complex. Since a complex $C_{m(N_j)}$, $N_i \prec N_j$ is merged into D whenever N_i desires, N_i is met. ∎

LEMMA 2.7
L_e is definable in all models of T by a Σ_1-formula in the restricted language $\mathcal{L}_P$, which depends only on e.

PROOF. Since $T = \mathrm{Th}(M_1)$ by definition, it suffices to work in M_1. Clearly, if $e \notin K$, then $L_e = \emptyset$. Now suppose $e \in K$. Then F_e is cofinite by Lemma 2.5. As discussed after (2.3), we want to prove that, for each $x, y \in \mathbb{N}$,

$$M_1 \models L_e xy \;\Leftrightarrow\; \dim(x), \dim(y) \geq g(e+1) \;\&\; d(x,y) \in F_e \cup \{\infty\}. \qquad (2.4)$$

This suffices, for $\dim(x) \geq g(e+1)$ can be expressed by a Σ_1–formula in $\mathcal{L}_P$, and, if $\mathbb{N} - F_e \subseteq \{0, \ldots, m-1\}$, $m > 1$, then $d(x,y) \in F_e \cup \{\infty\} \;\Leftrightarrow\; d(x,y) \geq m \Leftrightarrow \neg P_0 xy \;\&\; \ldots \;\&\; \neg P_{m-2} xy$.
As in Lemma 2.4, suppose that $x < y$ and $y \in \mathrm{dom}(M_{1,\tau}) - \mathrm{dom}(M_{1,\tau-1})$ (but note that possibly $\dim_\tau(y) < \dim_\tau(x)$). We denote by $\mathrm{Compl}_\sigma(z)$ the complex z is in by the end of substage σ.
For the direction from left to right, if $L_e xy$, then at the end of substage τ, the right hand side in (2.4) holds. Thus $\dim_\sigma(x), \dim_\sigma(y) \geq g(e+1)$ for all $\sigma \geq \tau$, since dimension is nondecreasing over substages. Moreover, if $d_\tau(x,y) \in F_{e,\tau}$ then $d(x,y) = d_\tau(x,y) \in F_e$. Suppose now that $d_\tau(x,y) = \infty$ (so that x, y are in different complexes at the end of τ), but $k = d(x,y)$ is finite. Then at some substage $\sigma > \tau$, $\mathrm{Compl}_\tau(x) = \mathrm{Compl}_{\sigma-1}(x) = C_{m(N)}$ is merged into D during a run of the *Merge* procedure, while $y \in D_{\sigma-1}$. (If, instead, y enters D while x is in D already, we argue similarly.) During σ, x is in a complex D' which is connected with $D'' \supseteq D_{\sigma-1}$ using P_{k-1}, where k is chosen large. Note that $R_e \prec N$, since $\dim_\sigma(x) \geq g(e+1)$. So, during the run of the *Merge* procedure, n_e is still defined at σ when we use P_{k-1} and $k \geq n_e$, hence $k \in F_e$.
Now suppose the right hand side in (2.4) holds. We show $L_e xy$.
1. If $\dim_\tau(x) \leq \dim_\tau(y)$ and $\dim_\tau(x) < g(e+1)$, then, at the end of substage τ, the numbers x, y are in different complexes, otherwise $d(x,y) = d_\tau(x,y) < g(e+1)$ while $\min(F_e) \geq g(e+1)$. Suppose at the end of substage τ, $x \in C_{m(N)}$, and $y \in D$ or $y \in C_{m(N')}$, where $N \prec N'$. Since $\dim(x) \geq g(e+1)$, at a stage $\sigma > \tau$ we merge $C_{m(N)} = \mathrm{Compl}_\tau(x) = \mathrm{Compl}_{\sigma-1}(x)$ into D while $y \in D_{\sigma-1}$. By the *Merge* procedure and since $N \prec R_e$, R_e was reset before we merged $\mathrm{Compl}_\tau(x)$. So we add a relation $P_{k-1}xy$ while n_e is undefined, whence $k = d(x,y) \notin F_e$, contradiction. If $\dim_\tau(y) \leq \dim_\tau(x)$ and $\dim_\tau(y) < g(e+1)$, we argue similarly. We can henceforth assume that $\dim_\tau(x), \dim_\tau(y) \geq g(e+1)$, so that, by the end of stage τ, x is in D or in some $C_{m(N)}$ for some $N \succ R_e$, and similarly for y.

2. If $d_\tau(x, y) = \infty$ and we do not declare $L_e xy$ at τ, then n_e is undefined at τ. If $e \in K_{s-1}$ then, by the end of stage $s-1$, n_e was defined, and we made n_e undefined at substages of s prior to τ. Then, while performing *Merge*, we would have merged the complexes x and y are in into D at a substage of s prior to σ, contrary to $d_\tau(x, y) = \infty$. If $e \notin K_{s-1}$, then, since $e \in K$, by (a.2) in the construction, we merge the distinct complexes $\mathrm{Compl}_\tau(x)$ and $\mathrm{Compl}_\tau(y)$ into D at some substage before we define n_e for the first time. So, again $d(x, y) \notin F_e$.
3. Finally suppose $d_\tau(x, y) = k < \infty$. Since $k \in F_e$ and P_{k-1} is used first at a substage $\leq \tau$, $k \in F_{e,\tau}$. Since y is added at τ, we declare $L_e xy$. ∎

Acknowledgment

The author would like to thanks Robert Soare and Alistair Lachlan for helpful discussions.

References

[1] J. Baldwin and A. Lachlan. On strongly minimal sets. *J. Symbolic Logic*, 36(1), 1971.

[2] B. Khoussainov, A. Nies, and R. A. Shore. Recursive models of theories with few models. *Notre Dame Journal of Formal Logic*, 38:165–178, 1997.

[3] K. Kudeiberganov. On constructive models of undecidable theories. *Siberian Math. Journ.*, 21(5), 1980.

Intuitionistic Finitely Many-Valued Sequent Calculi

EUGENIA REZNIK, *Institut Supérieur d'Electronique de Paris, 21 rue d'Assas, 75006 Paris, France. E-mail: eugenia.reznik@isep.fr*

PHILIPPE CURMIN, *Groupe Cyber Informatique, 115 bld Péreire, 75015 Paris, France. E-mail: philippe.curmin@logique.jussieu.fr*

Abstract

We present an intuitionistic sequent calculus for arbitrary finitely many-valued logics which has a positive semantics given by a natural extension of Kripke models. Our calculus has two types of rules: the *introduction* rules and the *pseudo-cut* rules. The introduction rules introduce formulas at all places on the left or on the right hand side of a sequent. They generalize the introduction rules of 2-places sequent calculus. The pseudo-cut rules allow to introduce formulas in the middle of a sequent. They may not exist in a 2-places sequent calculus. We show the soundness and completeness of our system and prove the cut-elimination theorem.

Keywords: finitely many-valued logics, sequent calculus, Kripke models, cut elimination

1 Introduction

This paper presents an intuitionistic sequent calculus for many-valued propositional logics. The study of such calculi is interesting from two points of view : first we want to get a general framework for an intuitionistic sequent calculus that would be sound and complete for some extension to m-valued case of Kripke models [4]. Such a framework helps answering the following question : among the ways allowing to obtain an intuitionistic system (like LJ [3]) from a classical one (like LK [3]) which one can easily be generalized to a m-valued case. Second, does a m-valued intuitionistic calculus have a nice computational interpretation ?
The intuitionistic many-places sequent calculi were already studied in [6] and [2]. In both cases a natural extension of Kripke models to the many-valued logics was proposed and a sound and complete sequent calculus was given for such models. However in both cases some additional restrictions were needed. In [2] the semantics used is the negative one, i.e. a sequent is satisfied by an interpretation iff for some place i there is a formula at this place that does not take the value v_i. In [6] to get a nice semantics the sequents are restricted to a special syntactical form where one formula can occur only at all places on the left or on the right hand side of the sequent, which means no "isolated" formula in the middle of a sequent. The introduction rules introduce formulas on every place left from or on every place right from any given place i and thus no introduction is possible in the middle of a sequent.
In this work we propose sequent calculi (with positive semantics i.e. where a sequent is satisfied iff there is a formula at a place i which takes the corresponding value v_i) that are sound and complete for the extension of Kripke models proposed in [6] and

which use the most general form of many-places sequents. These calculi are obtained from the intuitionistic calculi of [6] by adding a new rule called *pseudo-cut* which is the only one allowing to introduce formulas in the middle of a sequent. Such a rule may only exist for a m-places sequent calculus where $m > 2$. We prove the soundness and the completeness of these calculi and we show that they have the subformula property. Then we consider intuitionistic sequent calculi extended with a cut-rule and we prove the cut-elimination theorem for them.

Finally we study the following question : may the pseudo-cut rule be replaced by a single rule of introduction in the middle of a sequent ? In the classical sequent calculus the proofs with pseudo-cuts may be transformed to a special form where the active formulas of the pseudo-cuts are the conclusions of the left and right introductions. Thus the pseudo-cut rule may be replaced by a "direct" introduction of the formula in the middle of the sequent and we get a sequent calculus as in [1]. We show that in the intuitionistic case the use of pseudo-cuts may not be reduced to this special form, that would "emulate" a single rule of introduction in the middle of a sequent.

2 A generic intuitionistic sequent calculus

We describe here a generic intuitionistic sequent calculus system. The system is generic because it is designed for any m-valued propositional logic (with $m \geq 3$). It should be seen as a framework from which calculi for particular logics can be built. Let us call $\mathcal{V}$ the finite set $\{v_1; \dots ; v_m\}$ of all truth values.

2.1 Formulas and sequents

We consider many-places sequents as those presented in [5], [1].

The set $\mathcal{F}$ of propositional formulas is defined by induction both from an enumerable set $\mathcal{A}$ of atomic formulas (also called propositional variables) $a_1, \dots , a_n, \dots$ and from a finite set C of connectives.

Let c^r be any connective with arity r.

A sequent is a list of m *multi-sets* of formulas $\Gamma_1|\dots|\Gamma_i|\dots|\Gamma_m$. The positions $1 \dots m$ in this list are called the *places* of the sequent.

Note that the sequents are not restricted to a special form (as opposed to Rousseau's sequents [6]): a formula may occur in any set of places in a sequent.

<u>Some notations and definitions</u>:

Upper case Greek letters $\Gamma, \Delta, \Sigma, \Pi, \Omega, \dots$ are used to denote sequents and if Γ is a sequent then $\Gamma_1, \dots , \Gamma_m$ are the m multi-sets which compose it, thus $\Gamma = \Gamma_1|\dots|\Gamma_m$.

By Γ_i, A we denote the multi-set of formulas obtained by adding an occurrence of A in Γ_i. By A^* we denote any number (but at least one) of occurrences of the formula A.

Γ_i, Δ_j is the *union* of the multi-sets Γ_i and Δ_j.

If Γ and Π are both sequents then $\Gamma\Pi$ is their *union* which is the sequent where an occurrence of a formula A at any place i comes either from an occurrence of A in Γ or an occurrence of A in Π at the same place. $(\Gamma)^n$ denotes the union of n occurrences of the sequent Γ.

For any multi-set of formulas Γ_0 and any set of places I ($I \subseteq \{1; \dots ; m\}$) we use $|\Gamma_0|_I$ to denote the subsequent of Γ_0 whose occurrences of formulas are exactly those of Γ_0

at the places I. If the places in I are consecutive we use $|\Gamma_0|_{i,j}$ instead of $|\Gamma_0|_{\{i;\ldots;j\}}$. If I has only one element i, we abbreviate $|\Gamma_0|_{\{i\}}$ by $|\Gamma_0|_i$.

2.2 Deduction rules

The deduction rules are divided in four sets: axioms, introductions (*left introductions* and *right introductions*), structural rules and *pseudo-cuts*. We shall see that no rules but the *pseudo-cuts* may introduce a formula only in the "middle" of a sequent.
There is a *left introduction* rule (i.e. to all places $\leq i$) for every place $1 \leq i < m$ and a *right introduction* rule (i.e. to all places $\geq i$) for every place $1 < i \leq m$ and for every connective c^r.
These introduction rules are generated in a mechanical way from the truth table $f_c : \mathcal{V}^r \rightarrow \mathcal{V}$ of the connective c^r through the conjunctive normal forms ([5], [1]) of $f_c(x_1, \ldots, x_r) \leq v_i$ (for the left introduction rule to the places $\leq i$) and of $f_c(x_1, \ldots, x_r) \geq v_i$ (for the right introduction rule to the places $\geq i$).
Contractions and *weakenings* of formulas are done explicitly by using the structural rules.
We shall remark that our introduction and structural rules are the same as those of Rousseau's intuitionistic sequent calculus [6].
The sequent calculus is given by the following rules:

- Axioms are of the form $A|\ldots|A$.
- For any connective c^r and any truth value v_i there is a left introduction to all places from 1 to i:

  $$\frac{\Gamma^j\; \Pi^j(A_1, \ldots, A_r)\ (1 \leq j \leq n)}{\Gamma\; |c(A_1, \ldots, A_r)|_{1,i}}\; c\, Int : 1, \ldots, i$$

  Where:
  n is the number of premises of the rule
  Every sequent $\Pi^j(A_1, \ldots, A_r)$ is the *active* subsequent of the j^{th} premise.
  Γ is the union of all the contexts of the premises: $\Gamma = \Gamma^1 \cup \ldots \cup \Gamma^j \cup \ldots \cup \Gamma^n$.
- For any connective c^r and any truth value v_i there is a right introduction to all places from i to m:

  $$\frac{\Gamma^j\; \Pi^j(A_1, \ldots, A_r)\ (1 \leq j \leq n')}{\Gamma\; |c(A_1, \ldots, A_r)|_{i,m}}\; c\, Int : i, \ldots, m$$

  With the same conventions of notations as for the left introductions but with the following syntactical restriction on the context $\Gamma^j = \Gamma^j_1|\ldots|\Gamma^j_m$ of each premise:
  (RC)
  $$\mathcal{F}(\Gamma^j_m) \subseteq \ldots \subseteq \mathcal{F}(\Gamma^j_i) \subseteq \mathcal{F}(\Gamma^j_{i-1}) \subseteq \ldots \subseteq \mathcal{F}(\Gamma^j_1)$$

  Where $\mathcal{F}(\Gamma^j_i)$ denotes the set of formulas of Γ^j_i.
  As in [6] this restriction is the only syntactical difference between the intuitionistic and the classical system.

- For any place i there is a weakening at the place i:

$$\frac{\Gamma_1|\ldots|\Gamma_i|\ldots|\Gamma_m}{\Gamma_1|\ldots|\Gamma_i, A|\ldots|\Gamma_m}\ W : i$$

- For any place i there is a contraction at the place i:

$$\frac{\Gamma\ |A, A|_i}{\Gamma\ |A|_i}\ C : i$$

- For any place i (with $1 < i < m$) there is a *pseudo-cut* at the place i:

$$\frac{\Gamma\ |A|_{1,i} \quad \Delta\ |A|_{i,m}}{\Gamma\Delta\ |A|_i}\ pcut : i$$

  The pseudo-cut rule may only exist in a many-valued logic and is the only rule which allows the introduction of formulas in the middle of a sequent. It looks like the well-known cut rule of the two-valued sequent calculi, but the formula A can not completely disappear from the conclusion. As we shall see in the section **4.2** the *pseudo-cuts* are needed for the completeness of the system.

The system obviously has the *subformula property*, which we state as follows :

PROPOSITION 2.1
If a m-places sequent Γ is provable in our system then in any proof p of Γ all formulas are subformulas of formulas of Γ.

3 Semantics

We recall here the extension of Kripke's semantics to many-valued logics used by Rousseau in [6].
A Kripke model $\mathcal{K}$ for an m-valued logic is a structure $< K, \leq, Val >$ where :

- K is a set of nodes
- $\leq$ is a pre-order over K
- $Val = (v_k)_{k \in K}$ is a set of *valuations* $v_k : \mathcal{A} \to \mathcal{V}$ of atomic formulas such that :
  $\forall k, k' \in K$ if $k \leq k'$ then $\forall a \in \mathcal{A}\ v_k(a) \leq v_{k'}(a)$

Any valuation v_k for a node k of a model $\mathcal{K}$ is extended to an *interpretation* $v_k^* : \mathcal{F} \to \mathcal{V}$ of all formulas, by induction on the construction of formulas:

- $v_k^*(a) = v_k(a)$ for any atomic formula a
- $v_k^*(c(A_1, \ldots, A_r)) = inf_{k' \geq k}(f_c(v_{k'}^*(A_1), \ldots, f_c(v_{k'}^*(A_r))$ for any connective c^r.
  Where $f_c : \mathcal{V}^r \to \mathcal{V}$ is the truth table of c.

For any truth value $v_i \in \mathcal{V}$ we write $k \Vdash F^{v_i}$ iff $v^*(F) = v_i$, which means that F takes the value v_i at the node k.

The following property expresses that the interpretation of formulas is growing through the nodes of a model.

PROPOSITION 3.1
$\forall k, k' \in K$ if $k \leq k'$ then $\forall F \in \mathcal{F}\ v_k^*(F) \leq v_{k'}^*(F)$

3.1 Interpretation of a sequent in a Kripke model

The last step is to define how sequents are interpreted.
Let K be a m-valued Kripke model. For any m-places sequent Π we write $k \Vdash \Pi$ iff Π is *satisfied* by the node k of K, which means that there exists a place $1 \leq i \leq m$ and an occurrence of a formula at the place i such that $k \Vdash F^{v_i}$. We shall say that this occurrence of F is *satisfied* by the node k.
We write $\mathcal{K} \Vdash \Pi$ (i.e. Π is true in the model $\mathcal{K}$) iff $k \Vdash \Pi$ for every node $k \in K$.

4 Soundness and completeness

We now prove that our sequent calculus is *sound* and *complete* with respect to this semantics. These are properties needed for any proof theoretic system.

4.1 Soundness

The soundness statement expresses that all provable sequents are true, where the meaning of "true" depends on the chosen semantics.

THEOREM 4.1
If a m-places sequent Π is provable then it is true in all m-valued Kripke models.

PROOF. The proof is a straightforward adaptation from [6] and thus will not be detailed here. Note that the restriction (RC) is needed in the case of right introductions. ∎

4.2 Completeness

THEOREM 4.2
Every sequent Π which is true in all m-valued Kripke models is provable.

PROOF. Let us take any non provable sequent Π and find a Kripke model in which Π is false.
We proceed in two steps : first we add formulas to Π in order to get a non provable sequent in the form of Rousseau's sequents [6], then we apply the completeness theorem of Rousseau's sequent calculus [6].

1. We first transform the sequent Π in the following way :
   Let us call an *isolated* formula any formula which is neither at all places left from nor at all places right from a place i $(1 < i < m)$.
   We choose an isolated formula A in Π (if there is any).
   Let $|A|_{<i}$ be the sequent $A| \dots |A| \dots$ where A is at all places $< i$ and let $|A|_{>i}$ be the sequent $\dots |A| \dots |A$ where A is at all places $> i$.
   As we suppose that Π is not provable, we know that sequent unions $\Pi|A|_{<i}$ and $\Pi|A|_{>i}$ may not be both provable, because otherwise we would prove Π by applying a pseudo-cut rule.

   We then define a sequent Π^1 from Π as :
   $\Pi^1 = \Pi|A|_{<i}$ if $\Pi|A|_{<i}$ is not provable
   $\quad = \Pi|A|_{>i}$ otherwise.

By repeating this process with an isolated formula in Π_1 and so on, we finally get a finite sequence $\Pi^0, \ldots, \Pi^l$ of sequents such that :

- $\Pi^0 = \Pi$
- None of the sequents Π^j is provable.
- Π^j is a subsequent of Π^{j+1}.
- Π^{j+1} has one less isolated formula than Π^j.
- Π^l does not have any isolated formula.

2. We apply Rousseau's completeness theorem :

   Let $\Pi' = \Pi^l$.

   Π' is not provable and contains Π. As it has no isolated formula, it matches Rousseau's definition of many-places sequents [6].

   As Rousseau's intuitionistic sequent calculus is a subsystem of our calculus, we know that Π' is not provable in Rousseau's calculus.

   By Rousseau's completeness theorem [6] there is a Kripke model $\mathcal{K}_0$ in which Π' is false.

   As Π in included in Π', Π is false in $\mathcal{K}_0$ too.

∎

5 Cut elimination

The second part of this paper deals with the process of *cut elimination*. Proving cut elimination shows that our calculus is *analytic*, because cut-free proofs have the *subformula property*. Furthermore, the cut elimination process is the dynamics of the calculus and should correspond to some kind of computation in an algorithmic interpretation (hypothetical, so far) of our sequent calculus, as in the two values case [7].

5.1 Cuts

We first define a notion of *cut* for our m-places sequents and discuss the difference between cuts and pseudo-cuts.
A cut is a rule of the form :

$$\frac{\Gamma|A|_i \quad \Delta|A|_j}{\Gamma\Delta}\, cut_{i,j}$$

Where i, j are any two different places.

Let us first remark that the system extended with a general cut rule is equivalent to the initial system (**2.2**). However, pseudo-cuts allow the subformula property, but cuts obviously do not.
Second, for the extended system cut-elimination leads to replace some cut rules with pseudo-cuts. Typically a proof of the following form (in the case of 3-places sequents):

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma\, A|A|\_}\;Int:0,* \qquad \dfrac{\vdots}{\Delta\,\_|A|A}\;Int:*,1}{\Gamma\Delta\,\_|A,A|\_}\;cut:0,1}{\Gamma\Delta\,\_|A|\_}\;C:*$$

is transformed as :

$$\dfrac{\dfrac{\vdots}{\Gamma\, A|A|\_}\;Int:0,* \qquad \dfrac{\vdots}{\Delta\,\_|A|A}\;Int:*,1}{\Gamma\Delta\,\_|A|\_}\;pcut:*$$

We shall see in the following sections how the notion of pseudo-cuts is needed to represent rules which "cut formulas" but which can not be eliminated from proofs in general.

Finally, the use of pseudo-cuts may also not be reduced to the simulation of "middle introductions" (i.e. introductions to a place i $(1 < i < m)$), as we shall see in the section **6**.

<u>Generalized cuts</u>:

To simplify the proofs we use in the following section a more general notion of cut, based on disjoint sets of places. It is called *gcut* as "generalized cut" and defined as follows :

$$\dfrac{\Gamma\,|A|_I \quad \Delta\,|A|_J}{\Gamma\Delta}\;gcut:I,J$$

Where I, J are both non empty sets of places and $I \cap J = \emptyset$.

Proposition 5.1
The *gcut* rule is derivable.

Proof. We just use (usual) cuts and contractions to get the conclusion $\Gamma\Delta$ from $\Gamma\,|A|_I$ and $\Delta\,|A|_J$. The *gcut* is finally obtained through $n \times p$ cuts, where n, p are respectively the number of places in I, J. ∎

5.2 Weak normalization

Let MLJ (as "many-places LJ") be the many-places intuitionistic sequent calculus presented in the section **2.2** to which we add all cut rules.
We shall now prove the *weak normalization* property for MLJ, which we state as follows :

Theorem 5.2
For any proof p in MLJ there is a cut-free proof in MLJ with the same conclusion.

The proof of this result is mainly based on the following lemma, for the case where a cut is done immediately after two introductions (left and right) and the occurrences of formulas which are being cut are taken among the occurrences that were just introduced.

5.3 Cut elimination lemma

LEMMA 5.3
Let p be a proof which ends with a cut preceded by two introduction rules (one to the left and one to the right) such that only introduced occurrences are cut.

$$\dfrac{\dfrac{\begin{matrix}\vdots\, p_1^k \\ \Gamma^k\Pi^k(A_1,\ldots,A_r,B)\,(1\le k\le n)\end{matrix}}{\Gamma\,|A|_{1,i}}\,c\,Int:1,\ldots,i \qquad \dfrac{\begin{matrix}\vdots\, p_2^l \\ \Delta^l\Sigma^l(A_1,\ldots,A_r,B)\,(1\le l\le n')\end{matrix}}{\Delta\,|A|_{j,m}}\,c\,Int:j,\ldots,m}{\Gamma|A|_{1,i'-1}|A|_{i'+1,i}\Delta|A|_{j,j'-1}|A|_{j'+1,m}}\,cut:i',j'$$

Where c is any connective of arity $r+1$ $(r\ge 0)$ and the cut formula A is $c(A_1,\ldots,A_r,B)$. If all subproofs above both introduction rules are cut-free then there exists another proof p' with the same conclusion as p where the only cuts are done on the subformulas $A_1,\ldots,A_r,B$ of A.

Proving this lemma requires the following technical property (which is easily proven by applying $gcuts$):

PROPOSITION 5.4
Let Γ be any m-places sequent $\Gamma_1|\ldots|\Gamma_m$ and A be any formula.
Let $\Omega^1,\ldots,\Omega^i,\ldots,\Omega^m$ be the m sequents obtained from Γ by adding an occurrence of A respectively to all places but the place i (with $1\le i\le m$).
If $\Omega^1,\ldots,\Omega^m$ are provable then Γ is provable.

We now prove the main lemma **5.3**:

PROOF. To eliminate the cut on A in p we need to use the premises of both introduction rules in order to cut all active occurrences of the subformulas $A_1,\ldots,A_r,B$ of A.
As we keep the most general definition of the introduction rules, we have to invoke here the way these rules are built from the normal forms to state the following properties for the premises of both introductions of A :
For every r-tuple $(i_1,\ldots,i_r)$ of places among $\{1;\ldots;m\}$:

1. There exists exactly one premise $\Gamma^k\Pi^k(A_1,\ldots,A_r,B)$ of the left introduction and exactly one premise $\Delta^l\Sigma^l(A_1,\ldots,A_r,B)$ of the right introduction such that each formula A_s $(1\le s\le r)$ is at places in $\Pi^k(A_1,\ldots,A_r,B)$ and $\Sigma^l(A_1,\ldots,A_r,B)$ but the place i_s.
   Let us denote respectively by $\Pi^{(i_1,\cdots,i_r)}(A_1,\ldots,A_r,B)$ and $\Sigma^{(i_1,\cdots,i_r)}(A_1,\ldots,A_r,B)$ the active parts of both sequents.
   Let us denote by $\Pi^{(i_1,\cdots,i_r)}(A_1,\ldots,A_r)$ the subsequent of both $\Pi^{(i_1,\cdots,i_r)}(A_1,\ldots,A_r,B)$ and $\Sigma^{(i_1,\cdots,i_r)}(A_1,\ldots,A_r,B)$ where B has no occurrence.
   Then we have $\Pi^{(i_1,\cdots,i_r)}(A_1,\ldots,A_r,B)=\Pi^{(i_1,\cdots,i_r)}(A_1,\ldots,A_r)\,|B|_I$ and $\Sigma^{(i_1,\cdots,i_r)}(A_1,\ldots,A_r,B)=\Pi^{(i_1,\cdots,i_r)}(A_1,\ldots,A_r)\,|B|_{I'}$.
   Where I,I' are the sets of places where B has an occurrence in each sequent.
2. There is no place where B occurs both in $\Pi^{(i_1,\cdots,i_r)}(A_1,\ldots,A_r,B)$ and $\Sigma^{(i_1,\cdots,i_r)}(A_1,\ldots,A_r,B)$.

Both properties are trivial consequences of the way normal forms are built from the truth tables of each many-valued logic.
Property 1 requires that each introduction of A exactly has m^r premises. This is true if axioms are kept as premises. Thus we have $n = n' = m^r$.
Property 2 is true because both introductions are done on disjoint sets of places (we suppose $i < j$).

By property 2 we can first eliminate all active occurrences of B in the premises of both introductions. In the following we omit contexts in the sequents to simplify notations.
For each r-tuple $(i_1, \ldots, i_r)$ of places, we build a proof of $\Pi^{(i_1,\cdots,i_r)}(A_1, \ldots, A_r)$ by *gcut* on B from the premises $\Pi^{(i_1,\cdots,i_r)}(A_1, \ldots, A_r, B)$ and
$\Sigma^{(i_1,\cdots,i_r)}(A_1, \ldots, A_r, B)$.

Then property 1 and proposition **5.4** may be invoked to cut successively the formulas $A_r, \ldots, A_1$.
Indeed for each $r-1$-uplet $(i_1, \ldots, i_{r-1})$ we got by *gcuts* on B a proof of each of the following sequents:
$\Pi^{(i_1,\cdots,i_{r-1},1)}(A_1, \ldots, A_{r-1}, A_r), \ldots, \Gamma\Pi^{(i_1,\cdots,i_{r-1},m)}(A_1, \ldots, A_{r-1}, A_r)$.
By proposition **5.4** we eliminate by successive gcuts all active occurrences of A_r: thus we prove the m^{r-1} subsequents $Pi^{(i_1,\cdots,i_{r-1})}(A_1, \ldots, A_{r-1})$.
We repeat this process to eliminate successively all active occurrences of $A_{r-1}, \ldots, A_1$: if we add contractions in the context of the conclusion we finally get a proof of $\Gamma\Delta$. ∎

5.4 Notion of mix

To prove the cut elimination theorem we use the standard method ([3]) which consists in replacing the *cut* rule by the following rule, called *mix*, where all occurrences of A in each of the places i, j are being cut:

$$\frac{\Gamma\,|A^*|_i \quad \Delta\,|A^*|_j}{\Gamma\Delta}\, mix : i, j$$

Where $i \neq j$ and A has no occurrence at the place i in Γ and A has no occurrence at the place j in Δ. This rule is obviously derivable using cuts and contractions: we first contract all occurrences of A in $|A^*|_i$ and $|A^*|_j$, then we apply a $cut : i, j$. We can conversely derive the *cut* rule from the *mix* rule by applying weakenings to the conclusion of the *mix*.

Let us call MLJ' the sequent calculus obtained from MLJ by replacing *cuts* by *mixes*: MLJ and MLJ' are obviously equivalent.
We shall deduce weak normalization for MLJ from weak normalization for MLJ'.

5.5 Weak normalization of MLJ'

The theorem **5.2** is an immediate corollary of the following result for MLJ':

Theorem 5.5
For any proof p in MLJ' there is a proof p' with the same conclusion which is translated in MLJ as a cut-free proof.

Proof. Let p be any proof in MLJ'. We denote by $t(p)$ the size of p (i.e. the number of nodes in the proof tree). We call *degree* of a *mix* the length of the cut formula and $d(p)$ denotes the maximal degree of a *mix* in p. Let $n(p)$ be the number of *mixes* which degree is $d(p)$ in p.
The proof is done by induction on the 3-tuple $(d(p), n(p), t(p))$ (with the lexicographic ordering).
All cases on the last rule R of p have to be enumerated, but for lack of space we can not detail them here. ∎

6 A discussion about middle introductions

Let us consider here the particular case of MLJ for the 3-valued Lukasiewicz logic with the set of connectives $\{\neg; \to\}$. The 3 values are denoted $\{0; *; 1\}$.
We want to show that pseudo-cuts can not be reduced to a kind of "middle introduction" rule, that is to say a rule which would allow to introduce directly a formula in the middle place in the 3-places sequents. As we already remarked it, pseudo-cuts are the only means to prove sequents with "isolated" formulas (which are excluded by Rousseau's definition of sequents [6]).
Nevertheless we prove here that the use of pseudo-cuts can not be restricted to the following case, where the pseudo-cut immediately follows the left and the right introduction of the *pcut*-formula:

$$\dfrac{\dfrac{}{\Gamma_1, A|\Gamma_2, A|\Gamma_3}\,Int^-:0,* \qquad \dfrac{}{\Delta_1|\Delta_2, A|\Delta_3, A}\,Int^+:*,1}{\Gamma_1, \Delta_1|\Gamma_2, \Delta_2, A|\Gamma_3, \Delta_3}\,pcut$$

One may consider this succession of two introductions (respectively to the left and to the right) and a *pcut* as a "macro-rule" of introduction to the place $*$. Let us then call $macro - Int : *$ such a sequence of rules.
Let $MLJL$ (as "MLJ Light") be the subsystem of MLJ where it is forbidden to use a *pcut* anywhere but in a macro-introduction to the place $*$. In the following we prove that $MLJL$ is not complete for Lukasiewicz 3-valued logic, by showing a sequent which is provable in MLJ but not in $MLJL$.

6.1 Introduction rules for Lukasiewicz 3-valued logic

We recall here the introduction rules of MLJ (and thus of $MLJL$) for the connectives $\{\neg; \to\}$. These rules are built from simplified normal forms (such a simplification is possible too in the general case **2.2** as shown in [1]). Contexts are omitted to simplify notations.

1. Introduction rules for $\neg$:

$$\dfrac{\_|\_|A}{\neg A|\_|\_}\,Int^-:0 \qquad \dfrac{\_|A|A}{\neg A|\neg A|\_}\,Int^-:0,*$$

$$\dfrac{A|\_|\_}{\_|\_|\neg A}\,Int^+:1 \qquad \dfrac{A|A|\_}{\_|\neg A|\neg A}\,Int^+:*,1$$

2. Introduction rules for $\to$:

$$\frac{\_|\_|A \quad B|\_|\_}{A \to B|\_|\_}\; Int^- : 0$$

$$\frac{\_|A|A \quad B|B|\_ \quad B|\_|A}{A \to B|A \to B|\_}\; Int^- : 0, *$$

$$\frac{A|B|B \quad A|A|B}{\_|\_|A \to B}\; Int^+ : 1$$

$$\frac{A|A, B|B}{\_|A \to B|A \to B}\; Int^+ : *, 1$$

6.2 Remarkable sequents

Let us now consider 3-places sequents of the following form:
$B^m|(A \to B)^n, A^p|\neg\neg(A \to B)^q, A^r$ (1)
where A, B are atomic formulas and m, n, p, q, r are the number of occurrences of each formula in the sequent. These sequents have the following property:

PROPOSITION 6.1
If Π is a sequent of the form (1) and either $m = 0$ or $n = 0$ or $q = 0$ then Π is not provable in MLJ.

PROOF. In each of these 3 cases we easily find a Kripke model in which Π is false ... ∎

But if none of n, m, q is zero, then these sequents are all provable in MLJ.

PROPOSITION 6.2
Any sequent of the form (1) such that $n \neq 0, m \neq 0$ and $q \neq 0$ is provable in MLJ.

PROOF. We just prove the subsequent $B|(A \to B)|\neg\neg(A \to B)$.

$$\frac{\dfrac{\dfrac{A \to B|A \to B|A \to B}{\neg(A \to B), A \to B|A \to B|}\; Int^- : 0}{A \to B|A \to B|\neg\neg(A \to B)}\; Int^+ : 1 \qquad \dfrac{\dfrac{B|B|B}{A, B|A, B|B}\; W : 0, *}{B|A \to B|A \to B}\; Int^+ : *, 1}{B|A \to B|\neg\neg(A \to B)}\; pcut$$

∎

PROPOSITION 6.3
None of the sequents of the form (1) is provable in $MLJL$.

PROOF. Let Π be a sequent of the form (1). We reason by absurdity by supposing that there is a proof p of Π in $MLJL$.
We enumerate all possible forms for p: let R be the last rule in p which is neither a weakening nor a contraction. By property **6.1** above, we know that the conclusion of R is a sequent of the following form:
$B, B^{m_0}|A \to B, (A \to B)^{n_0}, A^{p_0}|\neg\neg(A \to B), \neg\neg(A \to B)^{q_0}, A^{r_0}$
Moreover: (2)

- R is not a left introduction of B, because B is atomic.

- R is not a right introduction of A, because A is atomic.
- R is not a right introduction of $\neg\neg(A \to B)$, because the restriction (RC) on the context would not be satisfied.

Thus the only possible rule for R is a *pcut*. As the only non atomic formula at the place $*$ is $A \to B$, R is a *pcut* on $A \to B$ and p has the following form:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ B^{m'_1}|(A\to B)^{n_1},A,A^{p_1}|\neg\neg(A\to B)^{q_1},A,A^{r_1}\quad\ldots\end{array}}{B^{m'},A\to B|A\to B,(A\to B)^{n_0},A^{p_0}|\neg\neg(A\to B),\neg\neg(A\to B)^{q_0},A^{r_0}}\,Int^- \qquad \dfrac{\begin{array}{c}B|B|B\\ \vdots\, W\\ B,B^{m''},A|A,B|B\end{array}}{B,B^{m''}|A\to B|A\to B}\,Int^+}{B,B^{m_0}|A\to B,(A\to B)^{n_0},A^{p_0}|\neg\neg(A\to B),\neg\neg(A\to B)^{q_0},A^{r_0}}\,pcut$$

Indeed:

- By the restriction (RC) on the contexts of the premises of Int^+, the occurrences of A and $A \to B$ at the place $*$ and the occurrences of A and $\neg\neg(A \to B$ at the place 1 can not be anywhere else than in the conclusion of Int^-.
- B must have at least one occurrence in the conclusion of Int^+ because the sequent $|A \to B|A \to B$ is not provable (if we give the value 1 to A and 0 to B the sequent is not satisfied).

Thus the premise of Int^+ is obtained by weakenings from the axiom $B|B|B$.

On the leftmost branch of p, we then go on with another sequent of the form (1), and the previous reasoning may be done again. Passing through all *pcuts* on the leftmost branch of the proof, we get a list $\Pi, \Pi_1, \ldots, \Pi_l$ o (f sequents of the form (1).
Let us consider now the first rule applied above Π_l which is neither a weakening nor a contraction: by (2) it must be a *pcut*, which is impossible. ∎

7 Conclusion

We presented an intuitionistic sequent calculus for finitely many-valued logics. To get a calculus with positive semantics and which is sound and complete for a natural extension of Kripke models, we had to introduce a new deduction rule (the *pseudo-cut*) which can not exist in the two-valued case. Nevertheless the restriction (RC) on the right introductions may be seen as a generalization of the restriction which defines a two-valued intuitionistic calculus from a classical calculus.
The next step in this work is to investigate the computational meaning (if there is any) of this calculus. In particular, what do pseudo-cuts mean from a computational point of view ?

References

[1] M. Baaz, C. Fermüller, R. Zach, Elimination of Cuts in First-order Finite-valued logics. Journal of Information Processing and Cybernetics, EIK29 (1993) 6, pp. 333-355.

[2] M. Baaz, C. Fermüller, Intuitionistic Counterparts of Finite-valued Logics. Logic Colloquium '95 (Abstracts of Contributed Papers), Haifa, August 1995.

[3] G. Gentzen, "Investigations into logical deduction", in *The Collected Work of Gerhard Gentzen*, M.E. Szabo (Ed.), North Holland, Amsterdam (1969).

[4] S. Kripke, Semantical analysis of intuitionistic logics. 1963. In Formal Systems and Recursive Functions, J.Crossley and M.Dummet, editors, pp.92-129, North-Holland, Amsterdam.

[5] G. Rousseau, Sequents in many valued logic I. 1967, Fund. Math., 60, 23-33.

[6] G. Rousseau, Sequents in many valued logic II. 1970, Fund. Math., 67, 125-131.

[7] P. Wadler, A Curry-Howard isomorphism for sequent calculus, private communication, 1993.

The Largest Cartesian Closed Category of Domains, Considered Constructively

DIETER SPREEN, *Fachbereich Mathematik, Theoretische Informatik, Universität Siegen, 57068 Siegen, Germany.*
E-mail: spreen@informatik.uni-siegen.de

Abstract

A conjecture of Smyth [10] is discussed which says that if D and $[D \to D]$ are effectively algebraic directed-complete partial orders with least element (cpo's), then D is an effectively strongly algebraic cpo, where it was left open what exactly is meant by an effectively algebraic and an effectively strongly algebraic cpo.
First, notions of an effectively strongly algebraic cpo and an effective SFP object are introduced. The effective SFP objects are just the constructive (computable) objects in the effectively given category [9] of indexed ω-algebraic cpo's.

THEOREM
Every effective SFP object is an effectively strongly algebraic cpo, and vice versa. Moreover, this equivalence holds effectively.

This shows that the given notion of an effective SFP object is stable. In effectivity considerations of ω-algebraic cpo's it is usual to require that the partial order be decidable on the compact elements. Here, we use a stronger assumption.

THEOREM
If D is an indexed ω-algebraic cpo that has a computable completeness test and $[D \to D]$ is an ω-algebraic cpo, then D is an effective SFP object.

An ω-algebraic cpo has a computable completeness test, if there is a procedure which decides for any two finite sets X and Y of compact cpo elements whether X is a complete set of upper bounds of Y. It is an open question whether this requirement can be weakened in the above result.

COROLLARY
The category of effective SFP objects and continuous maps is the largest Cartesian closed full subcategory of the category of ω-algebraic cpo's that have a computable completeness test.

Next, it is studied whether this result also holds in a constructive framework (or, to be more precise, in the framework of recursive mathematics), where one considers categories with constructive domains as objects, that is, domains consisting only of the constructive (computable) elements of an indexed ω-algebraic cpo, and computable maps as morphisms. The notions of a weakly indexed full subcategory and of being constructively Cartesian closed are introduced. The effectivity requirements in these definitions are very weak.

THEOREM
The category of constructive SFP domains is the largest constructively Cartesian closed weakly indexed full subcategory of the category of constructive domains that have a computable completeness test.

Constructive (effective) versions of domain-theoretic results are very important both for the foundations of computer science as well as for applications, since programming languages specify *computable* maps and *computable* (*effectively given*) data structures. Moreover, effective versions of classical results are good approximations to what can be proved constructively and such results have turned out to be at the heart of computer science, at least under the viewpoint of developing correct programs. In this respect the results of the paper are relevant to the workshop.

Keywords: Effectively given domains, SFP domains, largest Cartesian closed category of domains

1 Introduction

In his seminal paper [10] Smyth showed that the category **SFP** introduced by Plotkin [8] is the largest Cartesian closed category of domains, thus confirming a conjecture of Plotkin. In this paper we treat Plotkin's conjecture for the case of effectively given domains.

For various reasons one mostly uses the term *domain* to mean ω-algebraic directed-complete partial order with least element in studies of programming language semantics. Unfortunately, the class of domains is not closed under an important construction needed e.g. for the interpretation of higher-type procedures: the space $[D \to E]$ of continuous maps between two domains D and E must not be a domain again.

To circumvent this problem, people often restrict to the bounded-complete domains, the class of which is closed under the function space construction. However, also this class is not closed under all constructions needed in semantics: the Plotkin or convex powerdomain of a bounded-complete domain is not, in general, bounded-complete. Powerdomains are used for the interpretation of nondeterministic programs. Plotkin therefore introduced the larger class of SFP domains and showed that it is closed under the construction of his powerdomain as well as the function space. Moreover, he conjectured that if D and $[D \to D]$ are domains, then D is SFP. The conjecture, proved by Smyth, indicates that the category **SFP** of SFP domains and continuous maps is the largest category of domains closed under the constructions of interest.

The question which category has to be considered instead of **SFP**, if the term *domain* is allowed to mean some more general kind of directed-complete partial order, has extensively been studied by Jung [3, 4, 5, 1]. If instead of the space of continuous maps one confines to the space of stable maps, the corresponding problem has been dealt with by Amadio [2] and Zhang [12].

In his paper Smyth conjectured that with respect to a natural notion of effectively algebraic and of effectively strongly algebraic cpo the following statement be true: If D and $[D \to D]$ are effectively algebraic cpo's, then D is an effectively strongly algebraic cpo. The study of effectiveness is important in a theory of the foundations of programming. "One reason", said Smyth [9], "has to do with the systematic study of the power of specification techniques. We cannot require of a general purpose programming language that it be able to specify (define) all number-theoretic functions, but only (at most) those which are partial recursive. A corresponding distinction must be made for all the 'data types' which one may wish to handle. And the problem is not simply that of picking out the computable functions over a given data type; we have the problem of specifying the data types themselves, and thus of determining the 'computable', or effectively given, data types (i.e. the types which should in principle be specifiable)."

We first introduce the notions of an effectively strongly algebraic domain and of an effective SFP domain and show that their (effective) equivalence, which shows that we have obtained a stable effectivity notion for SFP domains.

Plotkin introduced SFP domains as colimits of ω-chains of finite domains with embeddings as connecting morphisms. Then he proved that they are exactly the strongly algebraic domains, that is, those domains for which for any finite set X of compact elements, the least set containing X and closed under the operation of taking all minimal upper bounds of subsets of X is finite. Here, we encode the finite domains and the embeddings between them in a canonical way and consider effective ω-chains. These are such that for a given natural number n one can compute both the index of the nth domains and the index of the nth embedding in the chain. Effective SFP domains are then defined to be colimits of such effective chains. An effectively strongly algebraic domain is a strongly algebraic domain which has an indexing of its compact elements such that for any finite set X of compact elements a canonical index of the set of its minimal upper bounds can be computed from a canonical index of X.

Effective SFP domains have also been studied by Kanda in his dissertation [7]. But whereas in the present paper the effective SFP domains are the constructive objects of the category of indexed domains (that is, domains which come with a fixed numbering of their compact elements), in the sense that each object can be constructed in an effective way from its finite parts, this is not the case in Kanda's treatment, as he does not code the finite domains by canonical or explicit indices, from which the domains can easily be recovered. Instead he codes finite domains in the same way as effectively given domains in general. This coding contains only partial information about the domain. (See also the remark of Smyth in [11, Section 5].)

In effectivity considerations of domains it is usual to require that the domain order be decidable on the compact elements. Here, we use a stronger requirement. A domain is said to have a computable completeness test if there is a procedure which decides for any two finite sets X and Y of compact elements whether X is a complete set of upper bounds of Y. We show that if D and $[D \to D]$ are domains such that D has a computable completeness test, then D is an effective SFP domain. It is not known, whether the condition of having a computable completeness test can be weakened in this result. As in Smyth [10] it follows that the category of effective SFP domains and continuous maps is the largest Cartesian closed full subcategory of the category of domains having a computable completeness test.

Next, it is studied whether this result also holds in a constructive framework, or, to be more precise, in the framework of recursive mathematics. Here, one considers categories with constructive domains as objects, that is, domains consisting only of the constructive (computable) elements of an indexed ω-algebraic cpo, and computable maps as morphisms. It is shown that the category of constructive SFP domains is the largest constructively Cartesian closed weakly indexed full subcategory of the category of constructive domains having a computable completeness test.

The effectivity requirements that have to be satisfied by the category are rather weak compared with the conditions considered by Kanda [6] and Smyth [9] in their approaches to effectiveness in categories. We only require that for any two objects the corresponding morphism set is indexed in such a way that the universality statement in the definition of a categorical product holds effectively.

The rest of this paper is organized as follows. In Section 2 basic definitions and results from domain theory are given. Section 3 is its effective counterpart. Here, the definition of an effectively given SFP domain is given and some properties are derived. Smyth's conjecture is treated in Section 4.

2 Domains

Let $(D, \sqsubseteq)$ be a partial order with smallest element $\bot$. For a subset S of D, $\downarrow S = \{ x \in D \mid (\exists y \in S) x \sqsubseteq y \}$ is the *lower set* generated by S. The subset S is called *compatible* if it has an upper bound. S is *directed*, if it is nonempty and every pair of elements in S has an upper bound in S. D is a *directed-complete* partial order (cpo) if every directed subset S of D has a least upper bound $\bigsqcup S$ in D, and D is *bounded-complete* if every compatible subset has a least upper bound in D.

An element x of a cpo D is *compact* if for any directed subset S of D the relation $x \sqsubseteq \bigsqcup S$ always implies the existence of an element $u \in S$ with $x \sqsubseteq u$. We write D^0 for the set of compact elements of D. If D^0 is countable and for every $y \in D$ the set $\downarrow\{y\} \cap D^0$ is directed and $y = \bigsqcup \downarrow\{y\} \cap D^0$, the cpo D is ω-*algebraic* or, as we prefer to say, a *domain*. A standard reference for domain theory is [1].

The *product* $D \times E$ of two cpo's D and E is the Cartesian product of the underlying sets ordered coordinatewise. Obviously, $D \times E$ is a domain again with $(D \times E)^0 = D^0 \times E^0$, if D and E are domains.

Definition 2.1

A map $F \colon D \to E$ between cpo's D and E is *continuous* if it is monotone and for any directed subset S of D,

$$F(\bigsqcup S) = \bigsqcup F(s).$$

Let $[D \to E]$ denote the set of all continuous maps from D to E. Endowed with the *pointwise order*, that is $F \sqsubseteq G$ if $F(x) \sqsubseteq G(x)$, for all $x \in D$, it is a cpo again, but in general it need not be a domain. This means that the category **DOM** of domains and continuous maps is not Cartesian closed. Therefore one considers subclasses of domains which have this property, when using domains in programming language semantics, e.g. SFP domains. To introduce this kind of domains we need the following definitions.

Definition 2.2

An *embedding/projection* (F, G) from a cpo D to a cpo E is a pair of maps $F \in [D \to E]$ and $G \in [E \to D]$ such that $G \circ F = \mathrm{Id}_D$, the identity map on D, and $F \circ G \sqsubseteq \mathrm{Id}_E$. The map F is called *embedding* and G *projection*.

Note that the map G is uniquely determined by F, and vice versa [11]. Therefore, we also write F^R instead of G. Embeddings are one-to-one and preserve compactness [8].

Lemma 2.3

Let D and E be domains and $F \colon D \to E$. Then F is an embedding if and only if there is a monotone and one-to-one map $F_0 \colon D^0 \to E^0$ such that for all $y \in E$ and all $u, u' \in D^0$, if $F_0(u), F_0(u') \sqsubseteq y$ then there exists some $\bar{u} \in D^0$ so that $u, u' \sqsubseteq \bar{u}$ and $F_0(\bar{u}) \sqsubseteq y$.

Suppose (F, F^R) is an embedding/projection from C to D and (G, G^R) is an embedding/projection from D to E. Then the composition of (F, F^R) and (G, G^R) is defined by

$$(G, G^R) \circ (F, F^R) = (G \circ F, F^R \circ G^R).$$

Let $\mathbf{DOM^e}$ denote the category of domains and embeddings.

By an ω-chain in $\mathbf{DOM}^\mathrm{e}$ we understand a diagram of the form $\mathcal{D} = D_0 \xrightarrow{F_0} D_1 \xrightarrow{F_1} \dots$ (that is, a functor from ω to $\mathbf{DOM}^\mathrm{e}$). As is well known, the category $\mathbf{DOM}^\mathrm{e}$ is ω-cocomplete: every ω-chain in $\mathbf{DOM}^\mathrm{e}$ has a colimit. Up to isomorphism this given by the set

$$D_\infty = \{\, x \in \Pi_{m\in\omega} D_m \mid (\forall m \in \omega) x_m = F_m^R(x_{m+1}) \,\}$$

endowed with the componentwise partial order, that is

$$x \sqsubseteq y \Leftrightarrow (\forall m \in \omega) x_m \sqsubseteq_{D_m} y_m.$$

Note that

$$D_\infty^0 = \{\, u \in D_\infty \mid (\exists m \in \omega) u_m \in D_m^0 \wedge (\forall n \geq m) u_{n+1} = F_n(u_n) \,\}.$$

Definition 2.4
An *SFP domain* is a colimit of an ω-chain in $\mathbf{DOM}^\mathrm{e}$, where all domains in the chain are finite.

In [8] Plotkin gave an alternative, purely order-theoretic characterization of SFP domains, which is quite useful in many cases.

Definition 2.5
Let D be a partial order, X be a subset of D and $\mathrm{UB}(X)$ be the set of all upper bounds of X.

1. An element x of D is a *minimal upper bound* of X if it is an upper bound of X and it is nor strictly greater than any other upper bound of X.
2. A subset Y of $\mathrm{UB}(X)$ is *complete* for X if whenever $x \in \mathrm{UB}(X)$, then $x \sqsupseteq y$ for some $y \in Y$.

Let $\mathcal{U}(X)$ be the set of minimal upper bounds of X. Then $\mathcal{U}(X)$ is contained in every subset Y of $\mathrm{UB}(X)$ that is complete for X. Define $\mathcal{U}^*(X)$ to be the least set containing X and closed under $\mathcal{U}$. Then a domain D is called *strongly algebraic* if for each finite subset X of D^0, $\mathcal{U}(X)$ is complete and $\mathcal{U}^*(X)$ is finite.

Theorem 2.6
A domain is an SFP domain if and only if it is strongly algebraic.

As is well known, any partially ordered D set may be viewed as a category. The objects of the category are the elements of D and the set of morphisms between two objects x and y is a one-point set precisely when $x \sqsubseteq y$ and the empty set otherwise. Under this view-point ω-chains correspond to infinite increasing sequences and the colimits of such chains to least upper bounds of the sequences. Having this analogy in mind it is natural to ask what are the compact or finitary objects of a category.

Definition 2.7
An object A of a category $\mathbf{K}$ is *finitary* in $\mathbf{K}$ provided that, for any ω-chain $\mathcal{K} = (V_n, F_n)_{n\in\omega}$ in $\mathbf{K}$ with colimit $\big(V, (G_n)_{n\in\omega}\big)$, the following holds: for any morphism $H \in \mathbf{K}[A, V]$, and for sufficiently large n, there is a unique morphism $K \in \mathbf{K}[A, V_n]$ such that $H = G_n \circ K$.

Obviously, the finitary objects in $\mathbf{DOM}^\mathrm{e}$ are just the finite domains.

3 Effectively given domains

In what follows, let $\langle\ ,\ \rangle : \omega^2 \to \omega$ be a recursive pairing function with corresponding projections π_1 and π_2 such that $\pi_i(\langle a_1, a_2\rangle) = a_i$, and let Δ be a standard coding of all finite subsets of natural numbers. We extend the pairing function in the usual way to an n-tuple encoding. Moreover, let $P^{(n)}$ $(R^{(n)})$ denote the set of all n-ary partial (total) recursive functions, and let W_i be the domain of the ith partial recursive function φ_i with respect to some Gödel numbering φ. We let $\varphi_i(a)\!\downarrow$ mean that the computation of $\varphi_i(a)$ stops and $\varphi_i(a)\!\downarrow \in C$ that it stops with value in C.

Let S be a nonempty set. A *(partial) numbering* $\natural$ of S is a partial map $\natural \colon \omega \rightharpoonup S$ (onto) with domain $\mathrm{dom}(\natural)$. The value of $\natural$ at $n \in \mathrm{dom}(\natural)$ is denoted, interchangeably, by $\natural_n$ and $\natural(n)$. The pair $(S, \natural)$ is called *numbered set*. Note that instead of numbering and numbered set, respectively, we also say *indexing* and *indexed set*.

Definition 3.1
Let $\natural$ and κ be numberings of the set S.

1. $\natural \le \kappa$, read $\natural$ is *reducible* to κ, if there is some function $g \in P^{(1)}$ with $\mathrm{dom}(\natural) \subseteq \mathrm{dom}(g)$, $g(\mathrm{dom}(\natural)) \subseteq \mathrm{dom}(\kappa)$, and $\natural_m = \kappa_{g(m)}$, for all $m \in \mathrm{dom}(\natural)$.
2. $\natural \equiv \kappa$, read $\natural$ is *equivalent* to κ, if $\natural \le \kappa$ and $\kappa \le \natural$.

A map $F \colon S \to S'$ from a numbered set $(S, \natural)$ to a numbered set $(S', \natural')$ is *effective* if there is some function $f \in P^{(1)}$ such that $f(i)\!\downarrow \in \mathrm{dom}(\natural')$ and $F(\natural_i) = \natural'_{f(i)}$, for all $i \in \mathrm{dom}(\natural)$.

The following definition is essentially due to Smyth [9].

Definition 3.2
An *effectively given* category is a category $\mathbf{K}$ together with a total indexing ϱ of its finitary objects and a total indexing ϑ of the morphisms between finitary objects such that the following conditions hold:

1. The set $\{\, \langle m, n\rangle \mid \varrho_m = \varrho_n \,\}$ is recursive.
2. The set $\{\, m \in \omega \mid \vartheta_m \text{ is an identity morphism} \,\}$ is recursive.
3. There are functions $d, c \in R^{(1)}$ such that $\varrho_{d(m)}$ and $\varrho_{c(m)}$, respectively, are the domain and codomain of ϑ_m.
4. There is a function $\mathrm{comp} \in P^{(2)}$ such that for all $m, n \in \omega$ for which the codomain of ϑ_m is the domain of ϑ_n, $\mathrm{comp}(m,n)\!\downarrow$ and $\vartheta_n \circ \vartheta_m = \vartheta_{\mathrm{comp}(m,n)}$.

An ω-chain $(A_m, F_m)_{m\in\omega}$ of finitary objects in $\mathbf{K}$ is *effective* if there is a function $t \in R^{(1)}$ such that $A_m = \varrho_{\pi_1(t(m))}$ and $F_m = \vartheta_{\pi_2(t(m))}$, for all $m \in \omega$. A *constructive* object A of $\mathbf{K}$ is then a colimit of an effective ω-chain of finitary objects in $\mathbf{K}$.

We have already seen that the finite domains are just the finitary objects of the category $\mathbf{DOM^e}$. The same holds if we confine to the subcategory $\mathbf{IDOM^{ce}}$ of indexed domains and computable embeddings. Here, an *indexed domain* (D, δ) is a domain D with a fixed total numbering δ of its compact elements.

Definition 3.3
Let (D, δ) and (E, ε) be indexed domains. A map $F \in [D \to E]$ is *computable* if the set $\{\, \langle i, j\rangle \mid \varepsilon_j \sqsubseteq F(\delta_i) \,\}$ is recursively enumerable (r.e.).

The numbering of the compact elements is used to impose certain effectivity requirements on these elements. A condition that we shall always use is the decidability of the domain order.

DEFINITION 3.4
A domain D with a total numbering δ of its compact elements is *effectively given* if the set $\{ \langle i,j \rangle \mid \delta_i \sqsubseteq \delta_j \}$ is recursive.

Note that an embedding from an effectively given domain into another effectively given domain is computable exactly if its restriction to the compact elements is effective. Moreover, a Gödel number of the function witnessing effectivity can be computed from an index of the r.e. set witnessing the computability of the embedding, and vice versa.
If (D,δ) and (E,ε) are indexed domains, define the numbering $\delta \times \varepsilon$ by $(\delta \times \varepsilon)_{\langle i,j \rangle} = (\delta_i, \varepsilon_j)$. Then $(D \times E, \delta \times \varepsilon)$ is an indexed domain again. If (D,δ) and (E,ε) are effectively given, the same is true for $(D \times E, \delta \times \varepsilon)$.
Domain elements of particular interest are those which can be approximated effectively.

DEFINITION 3.5
Let (D,δ) be an indexed domain. An element x of D is called *constructive* if the set $\{ i \in \omega \mid \delta_i \sqsubseteq x \}$ is r.e.

Observe that computable maps map constructive elements to constructive elements. We denote the set of constructive elements of D by D_c. With respect to the restriction of the domain order it is a partial order, which we call *constructive domain.*
Let us now introduce canonical indexings of the finite domains and the embeddings between finite domains. In order not to have to deal with isomorphic copies we consider only finite domains that have natural numbers as elements. For $m \in \omega$ set

$$E_m = \{ \langle m,a \rangle \mid a \in \pi_1(\Delta_m) \cup \pi_2(\Delta_m) \}$$

and order it by

$$\langle m,a \rangle \sqsubseteq_m \langle m,b \rangle \Leftrightarrow \langle a,b \rangle \in \Delta_m.$$

In case that E_m is a partial order with smallest element, all elements are compact. We enumerate them in the following way:

$$\eta_a^m = \begin{cases} \langle m,a \rangle & \text{if } \langle m,a \rangle \in E_m, \\ n_\perp & \text{otherwise,} \end{cases} \qquad (a \in \omega).$$

Here $n_\perp$ is the smallest element of E_m. Then (E_m, η^m) is an effectively given domain. Now, let $\zeta_m = (E_m, \eta^m)$, if E_m is a partial order with smallest element, and let $\zeta_m = (\{0\}, \{\langle 0,0 \rangle\}, \lambda a.0)$, otherwise. Moreover, for natural numbers $\langle m,i,n \rangle$ such that there is some embedding $F \in [E_m \to E_n]$ with $\Delta_i = \{ \langle a,b \rangle \mid \eta_b^n = F(\eta_a^m) \}$, define $\theta_{\langle m,i,n \rangle} = F$. In any other case set $\theta_{\langle m,i,n \rangle} = \mathrm{Id}_{\{0\}}$. Then ζ and θ, respectively, are numberings of the finite domains and the embeddings between these such that the category $\mathbf{IDOM^{ce}}$ is effectively given.

DEFINITION 3.6
An effective SFP domain is a colimit of an effective ω-chain in $\mathbf{IDOM^{ce}}$.

Let $\mathcal{D} = \big((D_m, \delta^m), F_m\big)_{m\in\omega}$ be an effective ω-chain in $\mathbf{IDOM^{ce}}$. Set $F_{mn} = F_{n-1} \circ \cdots \circ F_m$, for $m < n$, and $F_{mm} = \mathrm{Id}_{D_m}$. Moreover, let $\mathrm{in}_m \colon D_m \to D_\infty$, defined by

$$\mathrm{in}_m(x)(n) = \begin{cases} F_{mn}(x) & \text{if } m \le n, \\ F^R_{nm}(x) & \text{otherwise,} \end{cases}$$

for $x \in D_m$, be the canonical embedding of D_m into D_∞. For $\langle m, a\rangle \in \omega$ set $\delta^\infty_{\langle m,a\rangle} = \mathrm{in}_m(\delta^m_a)$. Then δ^∞ is an indexing of D^0_∞ such that D_∞ is effectively given.
If D is a colimit of $\mathcal{D}$ there is a computable isomorphism $H \in [D \to D_\infty]$. Isomorphisms are embeddings and as we have seen in Lemma 2.3, these are determined by their values on the compact elements. Moreover, they are computable just if their restriction to the computable elements is effective. Let φ_i and φ_j witness that the restrictions of H and H^{-1}, respectively, to D^0 and D^0_∞ are effective. Moreover, let φ_c witness that the ω-chain $\mathcal{D}$ is effective. Then $\langle i, j, c\rangle$ is an *index* of D. This defines a partial indexing σ of the effective SFP domains.
As we shall see next, Plotkin's order-theoretic characterization of the SFP domains also holds in the effective setting.

Definition 3.7
A domain D with total numbering δ of its compact elements is *effectively strongly algebraic* if it is strongly algebraic and the operation $\mathcal{U}$ is effective, that is, there is some function $g \in R^{(1)}$ such that $\mathcal{U}(\delta(\Delta_i)) = \delta(\Delta_{g(i)})$, for all $i \in \omega$.

Since $\delta_m \sqsubseteq \delta_n$ if and only if $\delta_n \in \mathcal{U}(\{\delta_m, \delta_n\})$, it follows that every effectively strongly algebraic domain (D, δ) is effectively given. If i is a Gödel number of the function g witnessing the effectivity of $\mathcal{U}$, then i is called an *index* of D. Let τ denote the indexing of the effectively strongly algebraic domains thus obtained.

Theorem 3.8
Every effective SFP domain is an effectively strongly algebraic domain, and vice versa. Moreover, this equivalence holds effectively, that is, $\sigma \equiv \tau$.

In the introduction it has already been mentioned that in his dissertation [7] Kanda studied SFP domains in an effective setting. But he does not work with canonical indexings of the finite domains and embeddings. As a consequence of this, the numbering of his effective SFP domains is weaker than the one used here. He has no equivalence between the numberings of the effective SFP domains and the effectively strongly algebraic domains, respectively, as above.
As is well known, the category **SFP** of SFP domains and continuous maps is Cartesian closed: the one-point domain $\{\perp\}$ is the terminal object, the domain product is the categorical product and the space of continuous maps between two SFP domains is the categorical exponent. Note that for two SFP domain D and E, $[D \to E]$ is an SFP domain again.

Definition 3.9
Let D and E be SFP domains. A finite subset T of $D^0 \times E^0$ is called *joinable* if

$$(\forall T' \subseteq T)[(\forall u \in \mathcal{U}^D(\mathrm{pr}_1(T')))(\exists v \in \mathcal{U}^E(\mathrm{pr}_2(T')))(u, v) \in T].$$

Here pr_i is the projection onto the ith component.

For elements $u \in D^0$ and $v \in E^0$ define the *step function* $(u \searrow v)\colon D \to E$ by

$$(u \searrow v)(x) = \begin{cases} v & \text{if } u \sqsubseteq x, \\ \bot & \text{otherwise,} \end{cases} \qquad (x \in D).$$

Then the compact elements of $[D \to E]$ are exactly the maps of the form $\bigsqcup \{ (u_i \searrow v_i) \mid i \in I \}$, where $u_i \in D^0$ and $v_i \in E^0$, for $i \in I$, so that $\{ (u_i, v_i) \mid i \in I \}$ is joinable.
If (D, δ) and (E, ε) are effective SFP domains, then it follows with Theorem 3.8 that the set $\{ i \in \omega \mid \{ (\delta_m, \varepsilon_n) \mid \langle m, n \rangle \in \Delta_i \} \text{ is joinable} \}$ is recursive. Thus we can define a numbering γ of $[D \to E]^0$ by setting

$$\gamma_i = \begin{cases} \bigsqcup \{ (\delta_m \searrow \varepsilon_n) \mid \langle m, n \rangle \in \Delta_i \} & \text{if } \{ (\delta_m, \varepsilon_n) \mid \langle m, n \rangle \in \Delta_i \} \text{ is joinable,} \\ (\bot_D \searrow \bot_E) & \text{otherwise,} \end{cases}$$

for $i \in \omega$. Then it is easily verified that $([D \to E], \gamma)$ is effectively given. It is even an effective SFP domain. In addition, we have the important property that an element F of $[D \to E]$ is constructive exactly if it is a computable map. Note that F is uniquely determined by its values on the computable elements.
Define a *constructive SFP domain* to be the constructive domain obtained from an effective SFP domain, then we achieve the following result.

THEOREM 3.10
The categories **ESFP** of effective SFP domains and continuous maps and **CSFP** of constructive SFP domains and computable maps are both Cartesian closed.

4 The conjecture

In his paper [10] Smyth conjectured that the proof of his Theorem 1 may be used to show that with respect to appropriate effectivity notions the following statement be true:

> If D and $[D \to D]$ are effectively algebraic domains, then D is an effectively strongly algebraic domain.

In effectivity considerations of domains these usually have to be effectively given. The effectivity requirement in the definition of effectively given domains is quite weak. It is not clear to us how the set of minimal least of upper bounds of a finite set of compact elements can be computed in the case of such domains. We therefore strengthen this condition.

DEFINITION 4.1
A domain D with a total numbering δ of its compact elements has a *computable completeness test* if the set

$$\{ \langle i, j \rangle \mid \delta(\Delta_j) \subseteq \mathrm{UB}(\delta(\Delta_i)) \wedge \delta(\Delta_j) \text{ is complete for } \delta(\Delta_i) \}$$

is recursive.

Since $\delta_i \sqsubseteq \delta_j$ if and only if $\{\delta_j\}$ is complete for $\{\delta_i, \delta_j\}$, every domain that has a computable completeness test is effectively given.

Let **EDOMCC** be the category of domains with a computable completeness test and continuous maps and **CDOMCC** the category of constructive domains obtained from domains with a computable completeness test and computable maps. Obviously, the object sets of both categories are closed under the construction of product domains, which is also the categorical product in these domains. Hence, both domains are Cartesian.

PROPOSITION 4.2

1. **ESFP** is a proper full subcategory of **EDOMCC**.
2. **CSFP** is a proper full subcategory of **CDOMCC**.

PROOF. Let (D, δ) be an effective SFP domain and note that

$$\begin{aligned}\delta(\Delta_j) \subseteq \mathrm{UB}(\delta(\Delta_i)) \wedge \delta(\Delta_j) &\text{ is complete for } \delta(\Delta_i) \\ &\Leftrightarrow \mathcal{U}(\delta(\Delta_i)) \subseteq \delta(\Delta_j) \wedge \delta(\Delta_j) \subseteq \mathrm{UB}(\delta(\Delta_i)).\end{aligned}$$

Since D is effectively strongly algebraic and hence also effectively given, the right hand side of this equivalence is recursive in i and j. Thus, D has a computable completeness test. ∎

Now, we can state and prove our version of Smyth's conjecture.

THEOREM 4.3
If D and $[D \to D]$ are domains such that D has a computable completeness test, then D is effectively strongly algebraic.

The second important result in Smyth's paper says that **SFP** is the largest Cartesian closed full subcategory of **DOM**. For the proof he needed the next result.

LEMMA 4.4
Let **K** be a full subcategory of the category **CPO** of cpo's and continuous maps. Then the following three statements hold:

1. If **K** has a terminal object T, then T is the one-point cpo.
2. If **K** has a terminal object and the product $A \times_{\mathbf{K}} B$ of objects A and B exists, then $A \times_{\mathbf{K}} B$ is isomorphic in **CPO** to the usual product $A \times B$.
3. If **K** has a terminal object and all products of pairs, and the exponent E^D of objects D and E exists, then E^D is isomorphic in **CPO** to the usual function space $[D \to E]$.

Now note that if D is a domain with a computable completeness test and D is isomorphic to a cpo E, then also E is a domain with a computable completeness test. With Theorem 4.3 we therefore obtain the following analogue of Smyth's result.

THEOREM 4.5
ESFP is the largest Cartesian closed full subcategory of **EDOMCC**.

In the rest of this section we deal with the question whether a similar statement is true with respect to **CSFP** and **CDOMCC**. As we shall see such a statement holds, but only in an effective categorical setting. Whereas in the definition of effective SFP domains we used rather strong, though very natural, effectivity requirements, we shall now employ only very weak conditions.

Definition 4.6
Let $\mathbf{K}$ be a category and for any two objects A and B, $\alpha^{A,B}$ be a partial indexing of the morphism set $\mathbf{K}[A, B]$. Then $(\mathbf{K}, (\alpha^{A,B})_{A,B\in \mathrm{Ob}_{\mathbf{K}}})$ is called *weakly indexed.*

Definition 4.7
Let $(\mathbf{K}, (\alpha^{A,B})_{A,B\in \mathrm{Ob}_{\mathbf{K}}}), (\mathbf{K}', (\beta^{A,B})_{A,B\in \mathrm{Ob}_{\mathbf{K}'}})$ be weakly indexed categories and A, B be objects of $\mathbf{K}$.

1. The categorical product $(A \times B, \mathrm{pr}_A, \mathrm{pr}_B)$ of A and B is *constructive* if for any object C of $\mathbf{K}$ there is a function $\mathrm{prod}_C \in P^{(2)}$ such that for all $a \in \mathrm{dom}(\alpha^{C,A})$ and all $b \in \mathrm{dom}(\alpha^{C,B})$, $\alpha^{C,A\times B}_{\mathrm{prod}_C(a,b)}$ is the unique morphism in $\mathbf{K}[C, A \times B]$ with
$$\alpha^{C,A}_a = \mathrm{pr}_A \circ \alpha^{C,A\times B}_{\mathrm{prod}_C(a,b)} \quad \text{and} \quad \alpha^{C,B}_b = \mathrm{pr}_B \circ \alpha^{C,A\times B}_{\mathrm{prod}_C(a,b)}.$$
2. $(\mathbf{K}, (\alpha^{A,B})_{A,B\in \mathrm{Ob}_{\mathbf{K}}})$ is a *full subcategory* of $(\mathbf{K}', (\beta^{A,B})_{A,B\in \mathrm{Ob}_{\mathbf{K}'}})$ if $\mathbf{K}$ is a full subcategory of $\mathbf{K}'$ and for all objects A, B of $\mathbf{K}$, $\alpha^{A,B} \equiv \beta^{A,B}$.

Definition 4.8
A weakly indexed category $\mathbf{K}$ is *constructively Cartesian closed* if $\mathbf{K}$ contains a terminal object and for every pair of objects there is a constructive categorical product and a categorical exponent.

We now have to verify that **CDOMCC** and **CSFP** are weakly effective categories. Let (D, δ) and (E, ε), respectively, be both effective SFP domains or both domains that have a computable completeness. If $F\colon D \to E$ is computable and $W_i = \{ \langle m, n\rangle \mid \varepsilon_n \sqsubseteq F(\delta_m) \}$, then we call i an *index* of the restriction of F to D_c. This defines partial indexings of $\mathbf{CSFP}[D, E]$ and $\mathbf{CDOMCC}[D, E]$, respectively.

Proposition 4.9
1. The category **CDOMCC** is weakly indexed.
2. The category **CSFP** is a constructively Cartesian closed weakly indexed full subcategory of **CDOMCC**.

In the framework of weakly effective categories Lemma 4.4 can be strengthened.

Lemma 4.10
Let $\mathbf{K}$ be a weakly indexed full subcategory of **CDOMCC**. Then the following three statements hold:

1. If $\mathbf{K}$ has a terminal object T, then T is the one-point cpo.
2. Let $\mathbf{K}$ have a terminal object. If the product $A \times_{\mathbf{K}} B$ of objects A and B exists, then $A \times_{\mathbf{K}} B$ is isomorphic in **CDOMCC** to the usual product $A \times B$.
3. Let $\mathbf{K}$ have a terminal object and all products of pairs. If the exponent E^D of objects D and E exists, and $\overline{E^D}$, $\bar{D}$ and $\bar{E}$, respectively, are objects in **EDOMCC** with $E^D = (\overline{E^D})_c$, $D = \bar{D}_c$ and $E = \bar{E}_c$, then $\overline{E^D}$ is isomorphic in **CPO** to the usual function space $[\bar{D} \to \bar{E}]$.

As above we now obtain our constructive analogue of Smyth's second result.

Theorem 4.11
CSFP is the largest constructively Cartesian closed weakly indexed full subcategory of **CDOMCC**.

References

[1] S. Abramsky and A. Jung, Domain theory, in: S. Abramsky, D. M. Gabbay and T. S. E. Maibaum, eds., *Handbook of Logic in Computer Science, Vol. 3, Semantic Structures*, Clarendon Press, Oxford, 1994, 1–168.

[2] R. Amadio, Bifinite domains: Stable case, in: D. H. Pitt, P.-L. Curien, S. Abramsky, A. M. Pitts, A. Poigné and D. E. Rydehard, eds., *Category Theory and Com. Sci.*, Lec. Notes in Comp. Sci. 530, Springer-Verlag, Berlin, 1991, 16–33.

[3] A. Jung, New results on hierarchies of domains, in: M. Main, A. Melton, M. Mislove and D. Schmidt, eds., *Mathematical Foundations of Programming Language Semantics*, Lec. Notes in Comp. Sci. 298, Springer-Verlag, Berlin, 1988, 303–310.

[4] A. Jung, Cartesian closed categories of domains, vol. 66 of CWI Tracts, Centrum voor Wiskunde en Informatica, Amsterdam, 1989.

[5] A. Jung, The classification of continuous domains, in: *Proc. Logic in Comp. Sci.*, IEEE Computer Society Press, 1990, 35–40.

[6] A. Kanda, Fully effective solution of recursive domain equations, in: J. Bečvář, ed., *Mathematical Foundations of Computer Science*, Lec. Notes in Comp. Sci. 74, Springer-Verlag, Berlin, 1979, 326–336.

[7] A. Kanda, Effective solutions of recursive domain equations, Ph. D. thesis, Univ. of Warwick, 1979.

[8] G. D. Plotkin, A powerdomain construction, *SIAM J. Comput.* 5 (1976) 452–488.

[9] M. B. Smyth, Computability in categories, in: J. de Bakker and J. van Leeuwen, eds., *Automata, Languages and Programming, Proc. Seventh Colloq.*, Lec. Notes in Comp. Sci. 85, Springer-Verlag, Berlin, 1980, 609–620.

[10] M. B. Smyth, The largest cartesian closed category of domains, *Theoret. Comput. Sci.* 27 (1983) 109–119.

[11] M. B. Smyth and G. D. Plotkin, The category-theoretic solution of recursive domain equations, *SIAM J. Comput.* 11 (1982) 761–783.

[12] G.-Q. Zhang, The largest cartesian closed category of stable domains, *Theoret. Comp. Sci.* 166 (1996) 203–219.

Towards Decision of Testing Equivalence for Petri Nets with Dense Time

IRINA VIRBITSKAITE, *Institute of Informatics Systems, Siberian Division of the Russian Academy of Sciences, Novosibirsk, Russia.*
E-mail: virb@iis.nsk.su [1]

Abstract

The intention of the paper is to extend the testing methodology to a model of Petri nets with a dense time domain. Alternative characterizations of timed testing relations are provided. We also show the decidability of timed testing for a subclass of the model under consideration. The key to our decision procedures is that timed testing equivalences can be reduced to appropriate symbolic bisimulations.

Keywords: time Petri nets, timed testing, symbolic bisimulation, decidability

1 Introduction

An important ingredient of every theory of concurrency is a notion of equivalence between processes. Typically, equivalences are used in the setting of specification and verification both to compare two distinct systems and to reduce the structure of a system.
The best known behavioural equivalence is bisimulation [8]. Two processes are assumed to be bisimilar, if the observer can not see any difference in their behaviours. The standard definitions of bisimulation equivalences and preorders are given in terms of recursively defined relations. The formulation is mathematically elegant and yields efficient algorithms for determining if finite-state processes are related. However, it is often the case that there is no smooth connection between the preorders and equivalences.
Another of the major equivalences of concurrency theory is that of testing, presented in [6]. Two processes are considered testing equivalent if there is no test that can distinguish them. This notion is intuitively appealing and has led to a well-developed mathematical theory of processes that ties together the equivalences and preorders. However, no characterization of these equivalences has led to an algorithmic solution for finite-state processes. Therefore testing decision procedures are based on reduction of testing to bisimulation [3].
Those equivalences were considered for formal system models without time delays. Recently, the demand for correctness analysis of real time systems, i.e. systems whose descriptions involve a quantitative notion of time, increases rapidly. Timed extensions of interleaving models have been investigated thoroughly in the last 5-10 years (see,

[1] This work is supported in part by the Russian Fund of Basic Research (Grant N 00-01-00898), and the Russian Ministry of High Education (Grant for Basic Research in Mathematics).

for example, [1]).

As a representive of a true concurrent approach, time Petri nets are considered in [7] to model real time systems over a dense time domain. A time Petri net is a Petri net whose transitions are labelled by two timing constraints that indicate their earliest and latest firing times. A state of a time Petri net is a set of its places containing tokens, equipped with a finite set of clocks corresponding to the transitions and recording the time elapsed since the transitions were enabled. A time Petri net progresses through a sequence of states in one of two ways: by firing transitions and letting a certain amount of time pass. The model appeared to us as a most simple and natural approach and, at the same time, it provably offers enhanced expressive capabilities w.r.t. several other timed extensions of Petri nets.

In this paper, we first seek to develop a framework for testing preorders and equivalences in the setting of time Petri nets to take into account processes' timing behaviour in addition to their degrees of relative nondeterminism and concurrency, and then try to apply the framework to characterize and, also, to decide timed testing equivalences reducing them to some variants of symbolic bisimulation.

There have been several motivations for this work. One has been given by the paper [10] which has provided alternative characterizations for testing preorders in the context of Petri net models with discrete time. A next origin of this study has been the paper [2], where both strong and weak timed bisimulations are shown to be decidable for dense time processes, by reducing the equivalences to the corresponding symbolic bisimulations. Finally, another origin has been the papers [4] and [9], which have treated timed testing for iterleaving models with discrete time and dense time, respectively. The latter paper has also provided a testing decision procedure that uses an untimed bisimulation between deterministic graphs built from mutually refined timer region graphs that are a finite abstraction of an operational semantics of the timed transition model considered. When we surveyed these results, we discovered that it was not the case for time Petri nets. Working further on the question we looked for a subclass of nets in which we could reduce timed testing relations to the corresponding variants of symbolic bisimulations.

The rest of the paper is organized as follows. The basic notions concerning time Petri nets are introduced in the next section. The definitions and alternative characterizations of timed testing and symbolic bisimulation are given in sections 3 and 4, respectively. In the following section, we introduce a subclass of deterministic time Petri nets, in which timed testing relations coincide with appropriate symbolic bisimulations. Decision procedures for the equivalences are further provided. Due to lack of the space, some of the proofs are omitted they can be found in a forthcoming paper.

2 Time Petri Nets

In this section, we define some terminology concerning time Petri nets [7]. Before introducing the basic concepts, we need to consider some auxiliary notations. Let Act be a finite set of visible actions and τ an internal action. Then $Act_\tau = Act \cup \{\tau\}$. Further, let $\mathbf{N}$ be the set of natural numbers, $\mathbf{R}^+$ the set of positive real numbers, and $\mathbf{R}_0^+$ the set of nonnegative real numbers. Define the set $Interv(\mathbf{R}_0^+) = \{(d_1, d_2), (d_1, d_2], [d_1, d_2), [d_1, d_2] \subset \mathbf{R}_0^+ \mid d_1, d_2 \in \mathbf{N}\}$.

We are now ready to introduce the concept of time Petri nets.

DEFINITION 2.1
A *time Petri net, labelled over* Act_τ, is a tuple $TN = (P, T, F, l, D, m_{TN})$, where P is a finite set of places; T is a finite set of transitions ($P \cap T = \emptyset$); $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation; $l : T \to Act_\tau$ is a labelling function; $D : T \to Interv(\mathbf{R}_0^+)$ is a timing function such that $D(t)$ is a closed interval from $Interv(\mathbf{R}_0^+)$ for all $t \in T$ with $l(t) \in Act$ [2]; $m_{TN} \subseteq P$ is the initial marking.

Fig. 1 shows an example of a time Petri net, where action labels and time intervals are drawn near to transitions. If no confusion arises, we will often omit place identities and use action labels rather transition identities to denote transitions.
For $t \in T$, ${}^\bullet t = \{p \in P \mid (p,t) \in F\}$ and $t^\bullet = \{p \in P \mid (t,p) \in F\}$ denote the *preset* and *postset* of t, respectively. To simplify the presentation, we assume that ${}^\bullet t \cap t^\bullet = \emptyset$ for every transition t. Let $\mathcal{TN}_\tau$ denote the set of all time Petri nets labelled over Act_τ.
A *marking* m of TN is any subset of P. A transition t is *enabled* in a marking m if ${}^\bullet t \subseteq m$ (all its input places have tokens in m), otherwise it is *disabled*. Let $enable(m)$ be the set of transitions, enabled in m. Let $\Gamma(TN) = [T \to \mathbf{R}_0^+]$ be the set of *time assignments* for transitions from T. Given $\natural \in \Gamma(TN)$ and $d \in \mathbf{R}_0^+$, we let $\natural + d$ denote the time assignment of the value $\natural(t) + d$ to each t from T.

TN_1 :

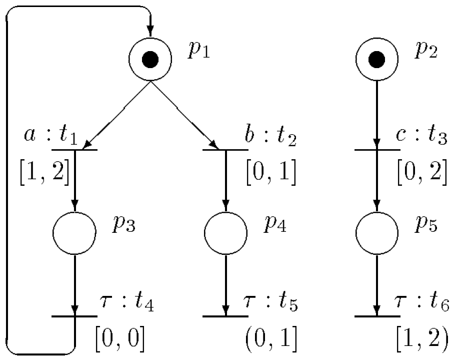


FIG. 1.

A *state* q of TN is a pair $<m, \natural>$, where m is a marking and $\natural \in \Gamma(TN)$. The *initial state* of TN is a pair $q_{TN} = <m_{TN}, \natural_{TN}>$, where $\natural_{TN}(t) = 0$ for all $t \in T$.
The states of TN change, if a time passes or if a transition fires. In a state $q = <m, \natural>$, a transition $t \in T$ *can fire* (denoted $q \xrightarrow{t}$), if $t \in enable(m)$ and $\natural(t) \in D(t)$. We shall write $q \xrightarrow{a}$, if $q \xrightarrow{t}$ and $l(t) = a$. The firing of t in q *leads to* the state $q' = <m', \natural'>$ (denoted $q \xrightarrow{t} q'$), if $m' = (m \setminus {}^\bullet t) \cup t^\bullet$, and forall $t' \in T$, $\natural'(t') = 0$, if $t' \in enable(m') \setminus enable(m)$, else $\natural'(t') = \natural(t')$. We shall write $q \xrightarrow{a} q'$, if $q \xrightarrow{t} q'$ and $l(t) = a$. In a state $q = <m, \natural>$, a time $d \in \mathbf{R}^+$ *can pass* if $\forall t \in enable(m)\ \exists d' \geq d \diamond \natural(t) + d' \in D(t)$. The passage of d in q *leads to* the state $q' = <m', \natural'>$ (denoted $q \xrightarrow{d} q'$), if $m' = m$, and $\natural'(t) = \natural(t) + d$ for all $t \in T$. A state q is *reachable* if either $q = q_{TN}$ or there exists a reachable state q' such that $q' \xrightarrow{x} q$ for some $x \in T \cup \mathbf{R}^+$. Let $RS(TN)$ denote the set of all reachable states of TN.

[2] Notice, τ-actions with open intervals are essential for alternative characterizing timed testing (see below).

A *run* r is a sequence of $q_{TN}, q_1, \ldots, q_{n-1}, q_n \in RS(TN)$ and $x_1, \ldots, x_n \in T \cup \mathbf{R}^+$ of the form: $r = q_{TN} \xrightarrow{x_1} q_1 \ \ldots \ q_{n-1} \xrightarrow{x_n} q_n$. We define the timing duration of a run r as follows: $time(r) = \sum_{1 \leq i \leq n,\ x_i \in \mathbf{R}^+} \ x_i$. Let $R(TN)$ denote the set of all runs in TN. For $t \in T$, we let $time(t) = \{time(r) \mid r = q_{TN} \xrightarrow{x_1} q_1 \ldots q_{n-1} \xrightarrow{x_n} q_n \in R(TN),\ q_n \xrightarrow{t}\}$ (the set of times at which a transition t can fire). As an illustration, we construct the following run r of the time Petri net TN_1 (see Fig. 1): $r = \langle\{p_1, p_2\}, \natural \equiv 0\rangle \overset{\delta_1=0.7}{\rightarrow} \langle\{p_1, p_2\}, \natural \equiv 0.7\rangle \overset{\delta_2=0.3}{\rightarrow} \langle\{p_1, p_2\}, \natural \equiv 1\rangle \overset{\delta_3=0}{\rightarrow} \langle\{p_1, p_5\}, \natural \equiv 1\rangle \overset{\delta_4=0}{\rightarrow} \langle\{p_4, p_5\}, \natural \equiv 1\rangle \overset{\delta_5=1}{\rightarrow} \langle\{p_4, p_5\}, \natural \equiv 2\rangle \overset{\delta_6=0}{\rightarrow} \langle\{p_5\}, \natural \equiv 2\rangle \overset{\delta_7=0}{\rightarrow} \langle\emptyset, \natural \equiv 2\rangle$. Moreover, we have $time(r) = 2$ and $time(t_6) = [1, 4)$.

We say that TN is *one-safe*, if for every $<m, \natural> \in RS(TN)$ and for every $t \in enable(m)$ it holds: $t^\bullet \cap m = \emptyset$. To guarantee that in any run of TN time is increasing beyond any bound, we need the following *progress condition*: for every set $\{t_1, t_2, \ldots, t_n\} \subseteq T$ (s.t. $\forall\, 1 \leq i < n \diamond t_i^\bullet \cap {}^\bullet t_{i+1} \neq \emptyset$ and $t_n^\bullet \cap {}^\bullet t_1 \neq \emptyset$) there exists $1 \leq j \leq n$ s.t. $0 \notin D(t_j)$. In the sequel, TN and TN' will always denote one-safe time Petri nets satisfying the progress condition.

The *weak leading relation* $\Rightarrow$ *on states of* TN is a relation defined by: $\overset{\epsilon}{\Rightarrow} \iff \overset{\tau}{\rightarrow}^*$ and $\overset{x}{\Rightarrow} \iff \overset{\epsilon}{\Rightarrow}\overset{x}{\rightarrow}\overset{\epsilon}{\Rightarrow}$, where $\overset{\tau}{\rightarrow}^*$ is the reflexive and transitive closure of $\overset{\tau}{\rightarrow}$, and $x \in Act \cup \mathbf{R}^+$. We consider the relation $\overset{d}{\Rightarrow}$ as possessing the time continuity property: $q \overset{d_1+d_2}{\Rightarrow} \iff q \overset{d_1}{\Rightarrow}\overset{d_2}{\Rightarrow}$ for some $d_1, d_2 \in \mathbf{R}^+$.

From now on, we shall use the following notions and notations. Let $Act(\mathbf{R}_0^+) = \{a(d) \mid a \in Act,\ d \in \mathbf{R}_0^+\}$ be the set of *timed actions* of Act over $\mathbf{R}_0^+$. Then $(Act(\mathbf{R}_0^+))^*$ is the set of finite *timed words* over $Act(\mathbf{R}_0^+)$. The function $\Delta : (Act(\mathbf{R}_0^+))^* \rightarrow \mathbf{R}_0^+$, measuring the *duration* of a timed word, is defined by: $\Delta(\epsilon) = 0$, $\Delta(w.a(d)) = \Delta(w) + d$. The domain for real-time languages is denoted by $Dom(Act,\ \mathbf{R}_0^+) = \{\langle w, d\rangle \mid w \in (Act(\mathbf{R}_0^+))^*,\ d \in \mathbf{R}_0^+,\ d \geq \Delta(w)\}$. The weak leading relation $\Rightarrow$ is extended to timed words from $(Act(\mathbf{R}_0^+))^*$ and $Dom(Act,\ \mathbf{R}_0^+)$ as follows. For given $d \in \mathbf{R}_0^+$, $d' \in \mathbf{R}^+$, $a \in Act$, and $w \in (Act(\mathbf{R}_0^+))^*$, we define: $q \overset{a}{\Rightarrow} q'$ iff $q \overset{a(0)}{\Rightarrow} q'$; $q \overset{d'}{\Rightarrow}\overset{a}{\Rightarrow} q'$ iff $q \overset{a(d')}{\Rightarrow} q'$; $q \overset{w}{\Rightarrow}\overset{a(d)}{\Rightarrow} q'$ iff $q \overset{w.a(d)}{\Longrightarrow} q'$; $q \overset{w}{\Rightarrow} q'$ iff $q \overset{\langle w,\ \Delta(w)\rangle}{\Longrightarrow} q'$; $q \overset{\langle w, d\rangle}{\Longrightarrow}\overset{d'}{\Rightarrow} q'$ iff $q \overset{\langle w,\ d+d'\rangle}{\Longrightarrow} q'$. The set $L(TN) = \{\langle w, d\rangle \in Dom(Act, \mathbf{R_0^+}) \mid \ q_{TN} \overset{\langle w, d\rangle}{\Longrightarrow}\}$ is the *language* of TN.

3 Timed Testing

In this section, we develop *timed testing* relations and provide their alternative characterizations in the setting of time Petri nets.

In general, testing preorders and equivalences relate processes on the basis of their behaviour in response to tests. One natural approach is taken by DeNicola and Hennessy in [6]: a test may be any process in which the set of actions contains a distinguished action ω, the *success* action; then, a process' behaviour in response to a test is given in terms of the *computations* that result when the test is 'applied' to the process. These intuitions may be formalized in the setting of time Petri nets as follows.

Let $\omega \notin Act_\tau$ and $Act_{\tau,\omega} = Act_\tau \cup \{\omega\}$. We shall use $\mathcal{TN}_{\tau,\omega}$ to denote the set of all time Petri nets labelled over $Act_{\tau,\omega}$ (the set of *tests*). For a test $TTN \in \mathcal{TN}_{\tau,\omega}$, let $\widetilde{q}_{TTN}$ be its *initial* state.

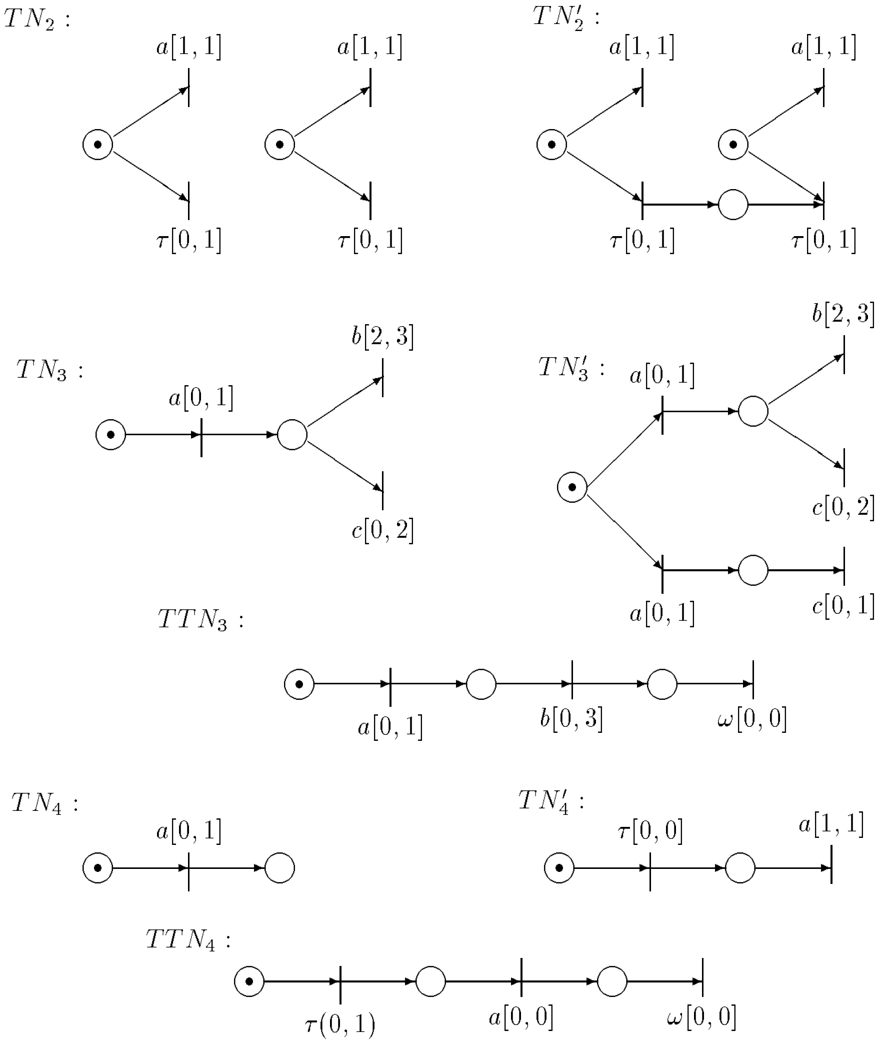


FIG. 2.

Let $TN \in \mathcal{TN}_\tau$, $TTN \in \mathcal{TN}_{\tau,\omega}$ and $q \in RS(TN)$, $\widetilde{q} \in RS(TTN)$. Then an expression $q||\widetilde{q}$ is called a *testing state*. A testing state is *successful*, if $\widetilde{q} \xrightarrow{\omega}$. The *leading relation on testing states* is defined by: $q||\widetilde{q} \xrightarrow{x} q'||\widetilde{q}'$, if $q \xrightarrow{x} q'$ and $\widetilde{q} \xrightarrow{x} \widetilde{q}'$ $(x \in Act \cup \mathbf{R}^+)$; $q||\widetilde{q} \xrightarrow{\tau} q'||\widetilde{q}$, if $q \xrightarrow{\tau} q'$; $q||\widetilde{q} \xrightarrow{\tau} q||\widetilde{q}'$, if $\widetilde{q} \xrightarrow{\tau} \widetilde{q}'$. The *weak leading relation on testing states* is defined in a similar way as on states of a time Petri net. A maximal sequence $\natural = q_0||\widetilde{q}_0 \xrightarrow{y_1} q_1||\widetilde{q}_1 \xrightarrow{y_2} \cdots \xrightarrow{y_n} q_n||\widetilde{q}_n$ $(y_i \in Act_\tau \cup \mathbf{R}^+,\ 1 \leq i \leq n,\ n \geq 0)$ is a *computation* of $q_0||\widetilde{q}_0$, if $\exists c \in \mathbf{R}^+\ \forall y_i \in \mathbf{R}^+ \diamond y_i \geq c$. We say that $y_1 y_2 \ldots y_n$ is the corresponding sequence of a computation $\natural$. A computation $\natural$ is called *successful*, if $q_i||\widetilde{q}_i$ is successful for some $0 \leq i \leq n$. We write q *may* $\widetilde{q}$, if there is a successful computation of $q||\widetilde{q}$; TN *may* TTN, if q_{TN} *may* $\widetilde{q}_{TTN}$. We write q *must* $\widetilde{q}$, if every

computation of $q \| \widetilde{q}$ is successful; TN *must* TTN, if q_{TN} *must* $\widetilde{q}_{TTN}$.
We are now ready to define the concepts of timed testing preorders and equivalences for time Petri nets TN and TN': $TN \leq_\alpha TN' \iff \forall\, TTN \in \mathcal{TN}_{\tau,\omega} \diamond TN\ \alpha\ TTN \Rightarrow TN'\ \alpha\ TTN$, where $\alpha \in \{may,\ must\}$; $TN \leq_{test} TN' \iff TN \leq_{may} TN'$ and $TN \leq_{must} TN'$; $TN \simeq_\alpha TN' \iff TN \leq_\alpha TN'$ and $TN' \leq_\alpha TN$, where $\alpha \in \{may,\ must,\ test\}$.
To illustrate the concepts introduced, consider the time Petri nets shown in Fig. 2. The *test*-equivalent nets TN_2 and TN_2' are first depicted. The next nets TN_3 and TN_3' are *may*-equivalent, but not *must*-equivalent. Consider the test TTN_3 (see Fig. 2). We have TN_3 *must* TTN_3, but $\neg(TN_3'\ must\ TTN_3)$, because there exists an unsuccessful computation of $M_{TN_3'} \| T_{TTN_3}$ with the corresponding sequence $a.1$. The non-*may*-equivalent nets TN_4 and TN_4' are finally shown. Consider the test TTN_4 (see Fig. 2). We have TN_4 *may* TTN_4, but $\neg(TN_4'\ may\ TTN_4)$, because every computation of $M_{TN_4'} \| T_{TTN_4}$ is unsuccessful.
In the following, we shall need some additional notions and notations. Let $q \in RS(TN)$ and $\langle w,\ d\rangle \in Dom(Act,\ \mathbf{R}_0^+)$. Then $S(q) = \{x \in Act \cup \mathbf{R}^+ \mid q \xrightarrow{x}\}$, and $Acc(TN,\ \langle w,\ d\rangle) = \{S(q') \mid q_{TN} \overset{\langle w,\ d\rangle}{\Longrightarrow} q',\ q' \not\xrightarrow{\tau}\}$ (timed acceptance set). Let $N, N' \subset 2^{Act \cup \mathbf{R}^+}$. Then $N \subset\subset N' \iff \forall S \in N\ \exists S' \in N' \diamond [(S' \mid_{Act} \subseteq S \mid_{Act}) \wedge (S \mid_{\mathbf{R}^+} = \emptyset \Rightarrow S' \mid_{\mathbf{R}^+} = \emptyset)]$; $N \equiv N' \iff N \subset\subset N' \ \wedge\ N' \subset\subset N$.
Timed testing can be characterized by means of timed languages and timed acceptance sets.

Lemma 3.1
Let $\langle w, d\rangle = \langle a_1(d_1) \ldots a_n(d_n), \sum_{i=1}^n d_i + d_{n+1}\rangle \in L(TN) (n \geq 0)$. Let $A_{\langle w,d\rangle} = \{\langle a_1(d_1') \ldots a_n(d_n'), \sum_{i=1}^n d_i' + d_{n+1}'\rangle \in Dom(Act \cup \mathbf{R}_0^+) \mid \forall 1 \leq k \leq l \leq n+1 \diamond \sum_{i=k}^l d_i' \in \wr \sum_{i=k}^l d_i \wr\}$. Then **(i)** $A_{\langle w,d\rangle} \subseteq L(TN)$, **(ii)** $\forall \langle w', d'\rangle \in A_{\langle w,d\rangle} \diamond Acc(TN, \langle w, d\rangle) \equiv Acc(TN, \langle w', d'\rangle)$. Here, for $d \in \mathbf{R}_0^+$, we let $\wr\, d\, \wr = [\ d, d\ ]$, if $d \in \mathbf{N}$, else $\wr\, d\, \wr = (\lfloor d \rfloor, \lceil d \rceil)$.

Proposition 3.2
(i) $TN \leq_{may} TN' \iff L(TN) \subseteq L(TN')$; **(ii)** $TN \leq_{must} TN' \iff \forall \langle w,\ d\rangle \in Dom(Act, \mathbf{R}_0^+) \diamond Acc(TN', \langle w, d\rangle) \subset\subset Acc(TN, \langle w, d\rangle)$.

4 Symbolic Bisimulation

In this section, we define and characterize some notions of symbolic bisimulation.
Central to defining the notions is a synchronization of different time Petri nets, gained by using the product of their states. Let $q \in RS(TN)$ and $q' \in RS(TN')$. Then a pair $q|q'$ is called an *extended state* of TN and TN'. We use $ES(TN, TN')$ to denote the set of all extended states of TN and TN'. Let $q|q',\ q_1|q_1' \in ES(TN, TN')$. Then the *leading relation on extended states* is defined as follows: $q|q' \xrightarrow{x} q_1|q_1'$, if $q \xrightarrow{x} q_1$ and $q' \xrightarrow{x} q_1'$ $(x \in Act \cup \mathbf{R}^+)$; $q|q' \xrightarrow{\tau} q_1|q_1'$, if either $q \xrightarrow{\tau} q_1$ and $q' = q_1'$ or $q' \xrightarrow{\tau} q_1'$ and $q = q_1$. The *weak leading relation on extended states* is defined and further extended to timed words from $Dom(Act, \mathbf{R}_0^+)$ in a similar way as on states of a time Petri net. Since time Petri nets are defined over a dense time domain, the number of their extended states is infinite. In order to get a discrete representation of the state-space of time Petri nets, we use the concept of regions (equivalence classes of states) [1]. Let $q = (m, \natural)|q' = (m', \natural')$, $q_1 = (m_1, \natural_1)|q_1' = (m_1', \natural_1') \in ES(TN, TN')$.

Then $q|q' \simeq q_1|q_1'$, if (1) $m = m_1$, $m' = m_1'$, (2) (i) for each $1 \leq i \leq n$ it holds: $\lfloor \natural|\natural'(t_i) \rfloor = \lfloor \natural_1|\natural_1'(t_i) \rfloor$ or $\natural|\natural'(t_i) + d \notin D(t_i)$ and $\natural_1|\natural_1'(t_i) + d_1 \notin D(t_i)$ for all $d, d_1 \in \mathbf{R}_0^+$; (ii) for each $1 \leq i, j \leq n$ s.t. $\natural|\natural'(t_i) + d \in D(t_i)$ for some $d \in \mathbf{R}_0^+$ and $\natural_1|\natural_1'(t_j) + d_1 \in D(t_j)$ for some $d_1 \in \mathbf{R}_0^+$ it holds: $\{\natural \mid \natural'(t_i)\} \leq \{\natural \mid \natural'(t_j)\} \iff \{\natural_1 \mid \natural_1'(t_j)\} \leq \{\natural_1 \mid \natural_1'(t_j)\}$, and $\{\natural \mid \natural'(t_i)\} = 0 \iff \{\natural_1 \mid \natural_1'(t_i)\} = 0$. Here, $n = \mid T_{TN} \mid + \mid T_{TN'} \mid$, $\natural|\natural'$ is the concatenation of vectors $\natural$ and $\natural'$, and for $d \in \mathbf{R}_0^+$, $\{d\}$ and $\lfloor d \rfloor$ denote its fractional and smallest integer parts, respectively. A set $R = [q|q'] = \{q_1|q_1' \in ES(TN, TN') \mid q|q' \simeq q_1|q_1'\}$ is called a *region* of TN and TN'.

Let R, R_1 be regions of TN and TN' such that $R \neq R_1$. Then the *leading relation on regions* is defined as follows: $R \xrightarrow{a} R_1$, if there exists $q|q' \in R$ and $q_1|q_1' \in R_1$ such that $q|q' \xrightarrow{a} q_1|q_1'$ ($a \in Act_\tau$); $R \xrightarrow{\chi} R_1$, if there exists $d \in \mathbf{R}^+$, $q|q' \in R$ and $q_1|q_1' \in R_1$ such that $q|q' \xrightarrow{d} q_1|q_1'$, and $q|q' \xrightarrow{d'} \widetilde{q}_1|\widetilde{q}_1' \in R \cup R_1$ for all $0 < d' \leq d$. We shall call a partition of $ES(TN, TN')$ into regions *stable* iff it holds: if $R \xrightarrow{a} R_1$, then $\forall q|q' \in R \diamond q|q' \xrightarrow{a} q_1|q_1'$ for some $q_1|q_1' \in R_1$ ($a \in Act_\tau$); and if $R \xrightarrow{\chi} R_1$, then $\forall q|q' \in R\ \exists d \in \mathbf{R}^+ \diamond q|q' \xrightarrow{d} q_1|q_1'$ for some $q_1|q_1' \in R_1$, and $q|q' \xrightarrow{d'} \widetilde{q}_1|\widetilde{q}_1' \in R \cup R$ for all $0 < d' \leq d$. The *region graph* of TN and TN' is the labelled directed graph $RG(TN, TN') = (V_{RG}, E_{RG}, l_{RG})$. The set of vertices V_{RG} is a stable partition of $ES(TN, TN')$ into regions, the set of edges E_{RG} is the leading relation on regions of V_{RG}, the labelling function $l_{RG} : E_{RG} \longrightarrow Act_\tau \cup \{\chi\}$ is defined as: $l((R, R_1)) = z \iff R \xrightarrow{z} R_1$. For $R \in V_{RG}$ and $z \in Act \cup \{\chi\}$, we define $Der(R, z) = \{R_1 \in V_{RG} \mid R \xrightarrow{z} R_1\}$. We also need a notion of the *weak leading relation on regions* of V_{RG}, given by: $R \stackrel{\epsilon}{\Rightarrow} R_1 \iff R \xrightarrow{\tau}{}^{*} R_1$, and $R \stackrel{z}{\Rightarrow} R_1 \iff R \stackrel{\epsilon}{\Rightarrow}\xrightarrow{z}\stackrel{\epsilon}{\Rightarrow} R_1$ ($z \in Act \cup \{\chi\}$).

We next define the concepts of a class (the τ-closure of regions) and the class graph [3] in the setting of time Petri nets. Let $RG(TN, TN') = (V_{RG}, E_{RG}, l_{RG})$ and $Q \subseteq V_{RG}$. A set $Q^\tau = \{R' \in V_{RG} \mid \exists R \in Q\ .\ R \stackrel{\epsilon}{\Rightarrow} R'\}$ is called a *class* of TN and TN'. Define $Q_0 = \{[q_{TN}|q_{TN'}]\}^\tau$, and $Der(Q,\ z) = \bigcup_{R \in Q} Der(R,\ z)$. For classes Q, Q_1 and $z \in Act \cup \{\chi\}$, the *leading relation on classes* is given by: $Q \xrightarrow{z} Q_1$, if $Q_1 = (Der(Q,\ z))^\tau$. Define $S(Q) = \{z \in Act \cup \{\chi\} \mid Q \xrightarrow{z}\}$. A class Q is called *reachable*, if either $Q = Q_0$ or there exists a reachable class Q_1 and $z \in Act \cup \{\chi\}$ such that $Q_1 \xrightarrow{z} Q$. The *class graph* of TN and TN' is the labelled directed graph $CG(TN, TN') = (V_{CG}, E_{CG}, l_{CG})$. The set of vertices V_{CG} is the set of reachable classes of TN and TN', the set of edges E_{CG} is the projection of the leading relation on classes to the set $V_{CG} \times V_{CG}$, the labelling function $l_{CG} : E_{CG} \longrightarrow (Act \cup \{\chi\})$ is defined as: $l((Q, Q_1)) = z \iff Q \xrightarrow{z} Q_1$.

For the correctness of our decision algorithms (see below), we need to introduce the following notion.

Definition 4.1

Let $\langle w, d \rangle \in L(TN) \cap L(TN')$ and $CG(TN, TN') = (V_{CG},\ E_{CG},\ l_{CG})$. Let $p = Q_0\ \ldots\ Q$ be a path in $CG(TN, TN')$. Then $q|q' \in ES(TN, TN')$ is *reachable by* $\langle w, d \rangle$ *consistent with* p iff $[q|q'] \in Q$ and either:

- $p = Q_0$ and $\langle w, d \rangle = \langle \epsilon, 0 \rangle$, or
- $p = p' \xrightarrow{z} Q$, there exists $q_1|q_1' \in ES(TN, TN')$ is reachable by $\langle w', d' \rangle$ consistent with p', and either:

– $z = a \in Act$, $q_1|q_1' \stackrel{a}{\Rightarrow}\stackrel{d''}{\to} q|q'$, and $\langle w, d\rangle = \langle w'a(d' - \Delta(w')), d' + d''\rangle$, for some $d'' \in \mathbf{R}_0^+$, or

– $z = \chi$, $q_1|q_1' \stackrel{d''}{\Rightarrow} q|q'$, and $\langle w, d\rangle = \langle w', d' + d''\rangle$, for some $d'' \in \mathbf{R}^+$.

Introduce some auxiliary notations. Let $q_1 = (m_1, \natural_1)|q_2 = (m_2, \natural_2) \in ES(TN, TN')$, and $i, j = 1, 2$ $(i \neq j)$. Then $q_i \xrightarrow{a}_{q_j}$ iff $(a \in Act \ \wedge\ q_i \xrightarrow{a})$, and $q_i \xrightarrow{\chi}_{q_j}$ iff $(\exists d \in \mathbf{R}^+ \circ q_i \xrightarrow{d} \ \wedge\ (m_1, \natural_1 + d)|(m_2, \natural_2 + d) \notin [q_1|q_2]$.
We are now ready to define some notions of symbolic bisimulation.

Definition 4.2
Let $CG(TN, TN') = (V_{CG}, E_{CG}, l_{CG})$, $\Pi \subseteq V_{CG}$ and $z \in (Act \cup \{\chi\})$ be given. Then

- $\beta \subseteq V_{CG}$ is called a *symbolic* Π-*bisimulation* between TN and TN', if $\beta \subseteq \Pi$, $Q_0 \in \beta$, and for all $Q \in \beta$ and $[q|q'] \in Q$ it holds:
  – if $q \xrightarrow{z}_{q'}$, then $\exists Q_1 \in \beta$ s.t. $[q|q''] \xrightarrow{z} [q_1|q_1']$ for some $[q|q''] \in Q$ and $[q_1|q_1'] \in Q_1$,
  – if $q' \xrightarrow{z}_{q}$, then $\exists Q_1 \in \beta$ s.t. $[q''|q'] \xrightarrow{z} [q_1|q_1']$ for some $[q''|q'] \in Q$ and $[q_1|q_1'] \in Q_1$,
- TN and TN' are *symbolic* Π-*bisimilar* (denoted $TN \approx_\Pi TN'$), if there exists a symbolic Π-bisimulation between TN and TN'.

A symbolic V_{CG}-bisimulation is a symbolic bisimulation (denoted $\approx$). We further define the relation $\Pi \subseteq V_{CG}$ as follows: a vertex Q of $CG(TN, TN')$ belongs to Π, iff for all $[q|q'] \in Q$ $([q|q'] \stackrel{\bar{z}}{\not\to})$ it holds:
– $\exists [q|q_1'] \in Q$ $([q|q_1'] \stackrel{\bar{z}}{\not\to}) \circ (S(q_1') \mid_{Act} \subseteq S(q) \mid_{Act})$, $(S(q) \mid_{\mathbf{R}^+} = \emptyset \Rightarrow S(q_1') \mid_{\mathbf{R}^+} = \emptyset)$,
– $\exists [q_1|q'] \in Q$ $([q_1|q'] \stackrel{\bar{z}}{\not\to}) \circ (S(q_1) \mid_{Act} \subseteq S(q') \mid_{Act})$, $(S(q') \mid_{\mathbf{R}^+} = \emptyset \Rightarrow S(q_1) \mid_{\mathbf{R}^+} = \emptyset)$.

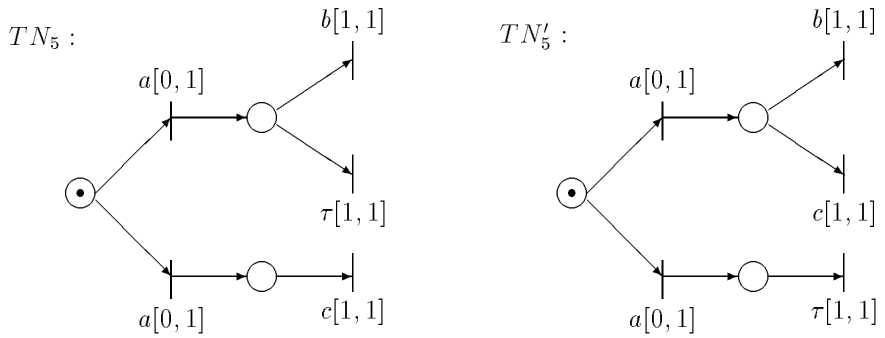


Fig. 3.

Illustrate the notions introduced by the following examples. First consider the non-bisimilar time Petri nets TN_2 and TN_2', shown in Fig. 2. There is no symbolic bisimulation between TN_2 and TN_2', because in $G(TN_2, TN_2')$ there exists a vertex Q with a region $[q|q']$ ($q|q'$ is reachable by $\langle a(1), 1\rangle$ consistent with the path from Q_0 to Q) such that $q \xrightarrow{a}_{q'}$, but in $G(TN_2, TN_2')$ there is no vertex Q_1 such that $[q|q''] \xrightarrow{a} [q_1|q_1']$ for some $[q|q''] \in Q$ and $[q_1|q_1'] \in Q_1$. Next consider the time Petri nets TN_3 and TN_3', shown in Fig. 2. We have $TN_3 \approx TN_3'$, but $TN_3 \not\approx_\Pi TN_3'$, because in any symbolic bisimulation between TN_3 and TN_3' there exists a vertex Q, which contains only two regions $[q|q'], [q|q'']$ ($q|q', q|q''$ are reachable by $\langle a(1), 1\rangle$ consistent with the path from Q_0 to Q) such that $S(q) = \{\{c\} \cup (0,1]\}$ and $S(q') = \{\{c\}\}$, i.e. $\neg(S(q')\mid_{\mathbf{R}^+} = \emptyset \Rightarrow S(q)\mid_{\mathbf{R}^+} = \emptyset)$. Finally, consider the symbolic Π-bisimilar nets TN_5 and TN'_5 shown in Fig. 3.

We further provide alternative characterizations of the concepts of symbolic bisimulation.

PROPOSITION 4.3

Let $CG(TN, TN') = (V_{CG}, E_{CG}, l_{CG})$ be given. Then **(i)** $TN \approx TN' \iff V_{CG}$ is a symbolic bisimulation, **(ii)** $TN \approx_\Pi TN' \iff V_{CG}$ is a symbolic Π-bisimulation.

5 Deciding the Equivalences

In this section, we consider the problem of deciding timed testing equivalences. The point of our decision algorithms is that timed testing relations are to be reduced to appropriate symbolic bisimulations. However, it has turned out that not every *test*-equivalent time Petri nets from the class $\mathcal{TN}_\tau$ are symbolic Π-bisimilar and vice versa. As was shown above, we have $TN_2 \simeq_{test} TN'_2$, but $TN_2 \not\approx TN'_2$ (see Fig. 2). On the other hand, consider the time Petri nets $TN_6 = [(a[0,1] \parallel b[0,1]) \parallel c[1,2]] \# [a[0,1]\ , (c[1,2] \parallel b[0,1])] \# [c[1,2] \parallel (a[0,1]\ , b[0,0])] \# [c[1,1] \parallel (a[0,1]\ , b[0,1])] \# [(b[0,1]\ , a[0,1]) \parallel c[1,2]] \# [(a[0,1] \# (a[0,1]\ , \tau[0,0])) \parallel (b[1,1] \parallel (\tau[1,2], \tau[0,0]) \parallel c[1,2])]$ and $TN'_6 = [a[0,1]\ , (c[1,2] \parallel b[0,1])] \# [c[1,2] \parallel (a[0,1]\ , b[0,0])] \# [c[1,1] \parallel (a[0,1]\ , b[0,1])] \# [(b[0,1]\ , a[0,1]) \parallel c[1,2]] \# [(a[0,1] \# (a[0,1]\ , \tau[0,0])) \parallel (b[1,1] \parallel (\tau[1,2]\ , \tau[0,0]) \parallel c[1,2])]$. We have $TN_6 \approx_\Pi TN'_6$ and $TN_6 \not\simeq_{may} TN'_6$, because the timed word $\langle a(0.5)b(0.2)c(0.5), 1.2\rangle$ belongs to $L(TN_6)$, but does not to $L(TN'_6)$. Here, to make our examples easier to understand, we gave for them algebraic expressions (see [5]) over actions with the time intervals of the corresponding transitions.

Therefore we now introduce a subclass of time Petri nets in which timed *may*- and *must*-equivalences are proved to coincide with symbolic and symbolic Π-bisimulation equivalences, respectively. A time Petri net TN is *deterministic*, if $\forall t, t' \in T \diamond \exists r = q_{TN} \xrightarrow{x_1} q_1 \ \dots\ q_{n-1} \xrightarrow{x_n} q_n \in R(TN)$ s.t. $q_n \xrightarrow{t}$ and $q_n \xrightarrow{t'} \Rightarrow (l(t) \neq l(t'))$, TN is *τ-discrete*, if $\forall t \in T \diamond l(t) = \tau \Rightarrow D(t) \subset \mathbf{N}$. Let $\mathcal{DTN}_\tau$ denote the class of time Petri nets having the determinacy and τ-discreteness properties. Notice, none of the example nets mentioned in the section possesses the properties.

The following theorem establishes the coincidence between timed testing relations and appropriate symbolic bisimulations.

THEOREM 5.1

Let $TN, TN' \in \mathcal{DTN}_\tau$. Then **(i)** $TN \simeq_{may} TN' \iff TN \approx TN'$, **(ii)** $TN \simeq_{must} TN' \iff TN \approx_\Pi TN'$ [3].

We are now ready to decide the problem of checking the equivalences.

THEOREM 5.2

Let $TN, TN' \in \mathcal{DTN}_\tau$. The checking problems for $TN \approx TN'$, $TN \approx_\Pi TN'$, $TN \simeq_{test} TN'$ are decidable.

PROOF. The algorithm for checking symbolic bisimulation between TN and TN' from $\mathcal{DTN}_\tau$ consists of the following steps. First, we construct the region graph of TN and

[3] It has turned out that the reduction of timed testing to symbolic bisimulation is crucial to decide testing even in the case of time Petri nets from $\mathcal{DTN}_\tau$, because the mutual synchronization of nets is still necessary.

TN', $RG(TN,TN')$, using the algorithm from [1]. In addition, each vertex $R=[q|q']$ of $RG(TN,TN')$ is equipped with a pair $\langle S_R, S'_R\rangle$, where $S_R=\{z\in(Act\cup\chi)\mid q\xrightarrow{z}_{q'}\}$ and $S'_R=\{z\in(Act\cup\chi)\mid q'\xrightarrow{z}_{q}\}$. Second, we construct the class graph of TN and TN', $CG(TN,TN')$, using the algorithm from [3]. In addition, each vertex Q of $CG(TN,TN')$ is equipped with $S_Q=\bigcup_{R\in Q}(S_R\cup S'_R)$. Let V_{CG} denote the set of all vertices of $CG(TN,TN')$. Using It is easy to show that for deciding whether V_{CG} is a symbolic bisimulation, it is sufficient to check whether $S_Q\subseteq S(Q)$ for each $Q\in V_{CG}$. The algorithm for checking symbolic Π-bisimulation between TN and TN' from $\mathcal{DTN}_\tau$ is based on that presented above. In addition, each vertex R of $RG(TN,TN')$ is equipped with acc_R ($acc_R \iff (R \stackrel{\mathcal{I}}{\not\Rightarrow} S_R = S'_R)$), and each vertex Q of $CG(TN,TN')$ is equipped with acc_Q ($acc_Q \iff \bigwedge_{R\in Q} acc_R$). It is easy to show that for deciding $V_{CG}\subseteq\Pi$, it is sufficient to check the truth of acc_Q for each $Q\in V_{CG}$, if V_{CG} is a symbolic bisimulation.
The decidability of checking $TN \simeq_{test} TN'$ follows from the existence of the above algorithms. ∎

Make some remarks on the complexity of the algorithm for checking a symbolic Π-bisimulation between time Petri nets TN and TN' from the class $\mathcal{DTN}_\tau$ (see the proofs of Theorem 5.2): according to [1], the procedure of constructing the region graph is exponential in the size of TN and TN'; according to [3], the procedure of constructing the class graph is polynomial in the size of the region graph; checking, whether V_{CG} is a symbolic Π-bisimulation, takes time complexity linear in the size of the class graph.

References

[1] R. Alur, C. Courcoubetis, D. Dill. Model checking in dense real time. *Information and Computation* **104** (1993) 2–34.

[2] K. Čerāns. Decidability of bisimulation equivalences for parallel timer processes. *Lecture Notes in Computer Science* **663** (1993) 302–315.

[3] R. Cleaveland, M. Hennessy. Testing equivalence as a bisimulation equivalence. *Lecture Notes in Computer Science* **407** (1989) 11–23.

[4] R. Cleaveland, A.E. Zwarico. A theory of testing for real-time. Proc. *LICS'91* 110–119.

[5] R. van Glabbeek, F. Vaandrager. Petri Net Models for Algebraic Theories of Concurrency. *Lecture Notes in Computer Science* **259** (1987) 224–242.

[6] R. De Nicola, M. Hennessy. Testing equivalence for processes. *Theoretical Computer Science* **34** (1984) 83–133.

[7] Merlin, P., Faber, D.J. Recoverability of communication protocols. *IEEE Trans. of Communication* COM-**24**(9) (1976).

[8] D. Park. Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science* **154** (1981) 561–572.

[9] B. Steffen, C. Weise. Deciding testing equivalence for real-time processes with dense time. *Lecture Notes in Computer Science* **711** (1993) 703–713.

[10] Vogler W. Faster Asynchronous Systems. *Lecture Notes in Computer Science* **962** (1995) 299–312.

A Gradient-Based Boosting Algorithm for Regression Problems

RICHARD S. ZEMEL, *Department of Computer Science, University of Toronto, Toronto, ON M5S 1A4, CANADA.*
E-mail: zemel@cs.toronto.edu

TONIANN PITASSI, *Department of Computer Science, University of Toronto, Toronto, ON M5S 1A4, CANADA.*
E-mail: toni@cs.toronto.edu

Abstract

Adaptive boosting, in which several weak learners trained sequentially are combined to boost the overall algorithm performance, is one of the most significant current developments in machine learning. Recently adaptive boosting methods for classification problems have been derived as gradient descent algorithms. This formulation justifies key elements and parameters in the methods, all chosen to optimize a single common objective function. The same formulation has yet to be developed for adaptive boosting of regression problems. We propose a novel objective function for regression problems that leads to a simple boosting algorithm. We prove that this method reduces training error, and compare its performance to other regression methods.

Keywords: adaptive boosting algorithms, regression problems, machine learning

1 Introduction

Boosting is a technique originally developed in computational learning theory (Schapire, 1990). Its aim is to "boost" the small advantage that a *hypothesis* produced by a *weak* learner can achieve over random guessing, by using the weak learning procedure several times on a sequence of carefully constructed distributions. Later boosting methods, notably AdaBoost (Freund & Schapire, 1997), created simple yet powerful algorithms which proved to be easy to implement and yielded excellent results in practice (e.g., Breiman, 1998; Schwenk & Bengio, 1998).

The significance of boosting extends beyond the learning community. It has important links to computational complexity (Impagliazzo, 1995; Klivans & Servedio, 1999), such as hardness amplification methods that have been used to prove inapproximability of certain boolean functions. There also turns out to be a close connection between boosting, game theory, and linear programming (Freund & Schapire, 1996; Grove & Schuurmans, 1998).

Two crucial elements of the adaptive boosting algorithms are the way in which a new distribution is constructed for the learning procedure to produce the next hypothesis in the sequence, and the way in which the (just-better-than-chance) hypotheses are combined to produce a highly accurate output. Both of these involve a set of parameters, whose values appeared to be determined in an ad hoc manner. Recently adaptive boosting algorithms have been derived as gradient descent algorithms (Schapire

& Singer, 1998; Friedman et al., 1999; Mason et al., 1999). These formulations justify the parameter values as all serving to optimize a single common objective function.
To date these optimization formulations of adaptive boosting have been developed for *classification* problems,but not for *regression* problems.We propose a new boosting algorithm for regression problems, in which a single common objective function determines the cost function used to produce the hypotheses, and the other parameters in the algorithm. In our method, standard algorithms may be used to produce hypotheses, and the algorithm is simple to implement and run. Finally, we provide a simple proof guaranteeing convergence on training error.
In the remainder of this paper, we first describe the basic adaptive boosting algorithm (for classification problems), and existing boosting methods for regression. We then present our novel method for boosting regression problems. Following the theorem and proof of error convergence, we describe empirical results comparing our method to others on some standard problems.

2 A Brief Summary of Boosting Methods

Adaptive boosting methods such as AdaBoost (Freund & Schapire, 1997) are simple modular algorithms that operate as follows. Let $g : \mathbf{X} \rightarrow Y$ be the function that we are trying to learn, where the label set Y is finite, typically binary-valued. The algorithm uses a learning procedure, which has access to n training examples, $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, drawn randomly from $\mathbf{X} \times Y$ according to distribution $\mathcal{D}$. After some amount of time, the learning procedure must output a hypothesis $f : \mathbf{X} \rightarrow Y$. The error of the hypothesis f is the expected value of a loss function on $f(\mathbf{x}), g(\mathbf{x})$ where $\mathbf{x}$ is chosen according to $\mathcal{D}$. A *strong* learning procedure is a procedure that given $\epsilon, \delta > 0$ and access to random examples, outputs with probability $1 - \delta$ a hypothesis with error at most ϵ, and where the running time is polynomial in $1/\epsilon$, $1/\delta$ and the number of examples. A *weak* learning procedure satisfies the same conditions, but where ϵ need only be better than $1/2$.
Schapire (1990) showed that given any weak learning procedure, denoted **WeakLearn**, it is possible to "boost" it, or efficiently transform it into a strong learning procedure. The AdaBoost algorithm achieves this by calling **WeakLearn** multiple times, in a sequence of T stages, each time presenting it with a different distribution over $\mathbf{X}$ and finally combining all of the hypotheses. At stage t, the algorithm maintains a set of weights w_t^i over the training examples and a distribution $\mathcal{D}_t$ is computed by normalizing these weights. The initial distribution $\mathcal{D}_1$ is typically uniform, so $w_1^i = \frac{1}{n}$ for all $i \leq n$. The algorithm loops through these two steps:

1. At stage t, the distribution $\mathcal{D}_t$ is given to **WeakLearn**, which generates a hypothesis f_t. The error rate ϵ_t of f_t w.r.t. $\mathcal{D}_t$ is:

$$\epsilon_t = \sum_{i : f_t(\mathbf{x}^i) \neq y^i} w_t^i / \sum_{i=1}^{n} w_t^i$$

2. The new training distribution is obtained from the new weights:

$$w_{t+1}^i = w_t^i * (\epsilon_t / (1 - \epsilon_t))^{1 - |f_t(\mathbf{x}^i) - y^i|}$$

Intuitively, the new training distribution is chosen to focus the next hypothesis on the parts of the input space that are mis-classified by the existing hypotheses. Thus, w^{t+1} should increase whenever f_t errs on example i, but the exact choice of w^{t+1} did not appear to be derived to optimize any particular function, but rather just given and then shown to work.

After T stages, a test example $\mathbf{x}$ will be classified by a combined hypothesis which is a weighted majority of the outputs of the T hypotheses: $\widehat{y} = \mathbf{sgn}(\sum_{t=1}^{T} c_t f_t(\mathbf{x}))$. Each *combination coefficient* $c_t = \mathcal{L}((\infty - \epsilon_{\sqcup})/\epsilon_{\sqcup})$ takes into account the accuracy of hypothesis f_t with respect to its distributions. Again here, while it is clear that c_t should increase as ϵ_t decreases, the rationale for these particular c_t values are somewhat of a mystery.

The original boosting algorithm described above applies to binary-valued classification problems. Adaptive boosting methods have been extended to apply to multiple-valued classification problems, and regression problems. Intuitively, regression problems (i.e., with real-valued targets) present special difficulties because hypotheses may not just be right or wrong, but can be a little wrong or very wrong. Previous regression boosting methods overcome this difficulty using a variety of techniques. One method converts the regression problem into a series of binary classification problems (Freund & Schapire, 1997); this approach permits the application of some of the analysis developed for the classification methods but is unwieldy in practice. Avnimelech & Intrator (1999) use a threshold to evaluate a response as correct or incorrect, and attempt to minimize the number of incorrect cases. This produces a strong dependence on the choice of threshold. Drucker (1997) avoids this conversion to classification-style problems by using a continuous loss function for training new hypotheses and reweighting the patterns. This method scales the errors of each hypothesis based on its maximal error, which can lead to big changes in combination coefficients due to a singular large error.

3 An Objective Function for Boosting Regression Problems

None of these boosting algorithms for regression take advantage of the recent work showing how adaptive boosting for classification can be derived as an optimization algorithm (Schapire & Singer, 1998; Friedman et al., 1999; Mason et al., 1999). These methods allow the parameters c_t and w_t to be derived from an overall objective function, which uses the training set to estimate the expected error of the combined hypotheses. The same objective is used in the **WeakLearn** procedure, as the new hypothesis is the step in function space in the direction of steepest descent of this objective.

We adopt this approach to derive a gradient-based boosting algorithm for regression problems. The objective function is:

$$J_T = \frac{1}{n} \sum_{i=1}^{n} \exp\left(\sum_{t=1}^{T} c_t [(f_t(\mathbf{x}^i) - y^i)^2 - \tau]\right) \tag{3.1}$$

In this formulation, each hypothesis is trained to minimize the squared error of a weighted training distribution. The constant τ is a threshold used to demarcate correct from incorrect responses. This threshold is the single parameter of this algorithm that must be chosen in a problem-dependent manner.

Adopting this objective makes the costs on each example multiplicative in the costs of the individual hypotheses. This key feature allows a boosting method to be formulated in an optimization framework; two attractive properties follow: (1) the objective function used to optimize a hypothesis is formed simply by re-weighting the training distribution; and (2) the new weight of a training example only depends on its old weight and the error of the latest hypothesis on that example.

In our algorithm, training hypothesis T to minimize J_T, the cost after T stages of boosting, amounts to minimizing the cost of that hypothesis on the weighted training distribution:

$$\begin{aligned} J_T &= \frac{1}{n}\sum_{i=1}^{n} \exp\Big(\sum_{t=1}^{T-1} c_t[(f_t(\mathbf{x}^i) - y^i)^2 - \tau]\Big) \exp(c_T[(f_T(\mathbf{x}^i) - y^i)^2 - \tau]) \\ &= \sum_{i=1}^{n} w_T^i \exp(c_T[(f_T(\mathbf{x}^i) - y^i)^2 - \tau]) \end{aligned}$$

The training distribution for the next hypothesis is then formed by normalizing these updated weights.

In the standard AdaBoost algorithm, the combination coefficient c_t can be analytically determined by solving $\frac{\partial J_t}{\partial c_t} = 0$ for c_t. Unfortunately, one cannot analytically determine the combination coefficient c_t in our algorithm, but a simple line search can be used to find value of c_t that minimizes the cost J_t. Finally, minimizing J with respect to y produces a simple linear combination rule for the estimate: $\widehat{y} = \sum_t c_t f_t(\mathbf{x}) / \sum_t c_t$. The algorithm can be summarized as follows:

New Boosting Algorithm

1. Input:
   - training set examples $(\mathbf{x}_1, y_1), ...(\mathbf{x}_n, y_n)$ with $y \in \Re$;
   - **WeakLearn**: a learning procedure that produces a hypothesis $f_t(\mathbf{x})$ whose accuracy on the training set is judged according to J
2. Choose initial distribution $p_1(\mathbf{x}^i) = p_1^i = w_1^i = \frac{1}{n}$
3. Iterate:
   - Call **WeakLearn** – minimize J_t with distribution $\mathbf{p}_t$
     (accept iff $\epsilon_t = \sum_i p_t^i \exp[(f_t(\mathbf{x}^i) - y^i)^2 - \tau] < 1$)
   - Set c_t to minimize J_t (using line search)
   - Update training distribution

$$\begin{aligned} w_{t+1}^i &= w_t^i * \exp(c_t[(f_t(\mathbf{x}^i) - y^i)^2 - \tau]) \\ p_{t+1}^i &= w_{t+1}^i / \sum_j w_{t+1}^j \end{aligned}$$

4. Estimate output y on input $\mathbf{x}$:

$$\widehat{y} = \sum_t c_t f_t(\mathbf{x}) / \sum_t c_t$$

4 Proof of Convergence

We currently can prove the standard boosting property for our algorithm in the case where all combination coefficients $c_t = 1$. Let $p_t^i = w_t^i / \sum_{j=1}^n w_t^j$, and ϵ_t denote the error that hypothesis t makes on its distribution,

$$\epsilon_t = \sum_{i=1}^n p_t^i \exp[(f_t(\mathbf{x}^i) - y^i)^2 - \tau]$$

Theorem: *Assume that for all $t \leq T$, hypothesis t makes error ϵ_t on its distribution. If the combined output $\widehat{y}$ is considered to be in error iff $(\widehat{y} - y)^2 > \tau$, then the output of the boosting algorithm (after T stages) will have error at most ϵ, where*

$$\epsilon = P[(\widehat{y^i} - y^i)^2 > \tau] \leq \prod_{t=1}^T \epsilon_t$$

.

Proof: We follow the approach used in the AdaBoost proof (Freund & Schapire, 1997). We show that the sum of the weights at stage T is bounded above by the product of the ϵ_t's, while at the same time, for each input i that is incorrect, its corresponding weight w_T^i at stage T is significant.

$$\sum_{i=1}^n w_{T+1}^i = \sum_i w_T^i \exp[(f_T(\mathbf{x}^i) - y^i)^2 - \tau] = \epsilon_T \sum_i w_T^i = \prod_{t=1}^T \epsilon_t$$

The second equality holds because $\epsilon_t = (\sum_i w_t^i \exp[(f_t(\mathbf{x}^i) - y^i)^2 - \tau]) / \sum_i w_t^i$. Now let $\widehat{y^i} = \sum_t f_t(\mathbf{x}^i)/T$. Then the weight of example i at time t is:

$$\begin{aligned} w_t^i =& \exp(\sum_t [(f_t(\mathbf{x}^i) - y^i)^2 - \tau]) = \exp(\sum_t [((f_t(\mathbf{x}^i) - \widehat{y^i}) + (\widehat{y^i} - y^i))^2 - \tau]) \\ =& \exp(T[\mathbf{Var}(f(\mathbf{x}^i)) + (\widehat{y^i} - y^i)^2 - \tau]) \geq \exp(T[(\widehat{y^i} - y^i)^2 - \tau]) \end{aligned}$$

The last inequality above holds because $\mathbf{Var}(f(\mathbf{x}^i)) = 1/T \sum_t (f_t(\mathbf{x}^i) - \widehat{y^i})^2$.
Now consider an example input i such that the final answer is an error. Then we have $(\widehat{y^i} - y^i)^2 > \tau \quad \Rightarrow \quad w_t^i \geq 1$ If ϵ is the total error rate of the combination output, then $\sum_i w_t^i \geq \epsilon$. Thus, as desired,

$$\epsilon \leq \sum_i w_{T+1}^i = \prod_{t=1}^T \epsilon_t$$

Note that as in the binary AdaBoost theorem, there are no assumptions made here about ϵ_t, the error rate of individual hypotheses. Also, if all $\epsilon_t < 1 = \Delta$, where $\Delta < 1$, then $\epsilon < \Delta^T$.

5 Empirical Tests of Algorithm

We report results comparing the performance of our new algorithm with two other algorithms. The first is the sequential residual-fitting algorithm (Hastie & Tibshirani, 1990), which produces hypotheses sequentially to reduce *residual* error. Here hypothesis T is trained not to produce the target output y^i on a given case i, but instead to fit the current residual, $\widetilde{y}$, where $\widetilde{y}^i = y^i - \sum_{t=1}^{T-1} c_t f_t(\mathbf{x})$. We used a least-squares loss function; Friedman (1999) explored the performance of this algorithm under various loss criteria. Note that this algorithm is similar to ours in that it develops a series of hypotheses all based on optimizing a common objective. It deviates from standard boosting, however, in that the distribution of examples is not used to control the generation of hypotheses, and each hypothesis is not trained to learn the same underlying function. Karakoulas and Shawe-Taylor (1999) have proposed a more standard boosting method based on residual-fitting.

The second algorithm used for comparison is a version of the mixture-of-experts algorithm (Jacobs et al., 1991). Here the hypotheses (or experts) are trained simultaneously. The standard mixture-of-experts uses a gating network to combine the hypotheses, and these combination coefficients also depend on the input. We set the weights from the input into the gating network to zero, and only adapted the biases to these units in order to make this model comparable to the others. This algorithm provides a good alternative to the greedy stage-wise methods, in that the experts are trained simultaneously to collectively fit the data.

We have selected two test problems on which to evaluate these algorithms. The first was proposed by Friedman (1991). F1 is a nonlinear prediction problem which has 10 independent input variables that are uniform in $[0, 1]$.

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - .5)^2 + 10x_4 + 5x_5 + n$$

where n is a random variable drawn from a mean-zero, unit-variance normal distribution. In this problem, only five input variables (x_1 to x_5) are needed to predict the correct output y, but the predictor must try to distinguish those input variables that have no prediction ability (x_6 to x_{10}) from those that do. We rescaled the target values y to be in $[0, 3]$. We used 400 training examples, and 100 validation and 100 test examples. The second test problem is the Boston Housing problem, a standard regression problem from the UCI machine learning database. Here there are 506 examples with the dependent variable being the price of housing in the Boston area, and twelve continuous input variables. We scaled the input variables to be in $[0, 1]$, and the outputs to be in $[0, 3]$. We used 400 of the examples for training, 50 for validation, and the remainder to test generalization.

In these experiments, we used neural networks as the hypotheses and back-propagation as the learning procedure to train them. Each network had a layer of 3 $tanh()$ units between the input units and a single linear output. For all three algorithms, we used early stopping with a validation set in order to reduce over-fitting in the hypotheses. Figure 1 shows average results over 20 runs of each algorithm on the two problems, with each run involving a new assignment of examples into the three sets. For the results shown in the figure, $\tau = 0.1$ in our algorithm. The results were consistent for different values of τ. All three algorithms steadily reduce the training error. The critical comparison considers their performance on the test examples. We measure test error as the normalized mean squared error (NMSE), the MSE divided by the variance across the data. As can be seen in the figure, the residual-fitting algorithm initially out-performs our boosting algorithm. Yet with more stages of boosting, the test error of the residual-fitting algorithm began to *increase*, which means that the combined

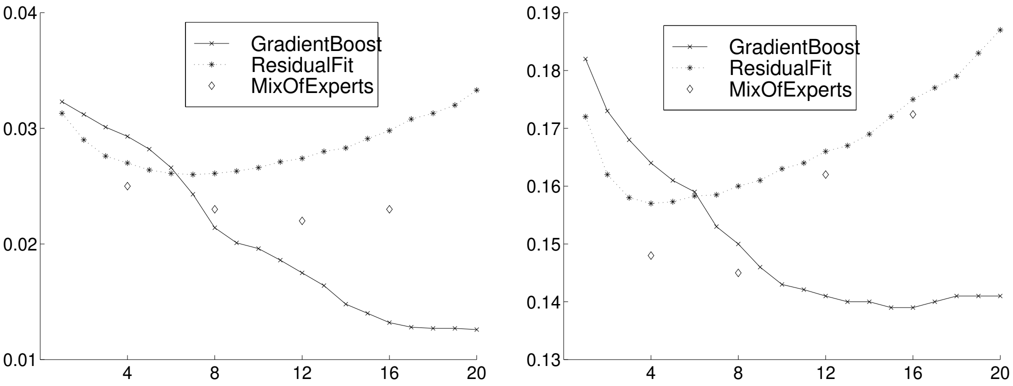


FIG. 1: Generalization results for our gradient-based boosting algorithm, compared to the residual-fitting and mixture-of-experts algorithms. Left: Test problem F1; Right: Boston housing data. Normalized mean-squared error is plotted against the number of stages of boosting (or number of experts for the mixture-of-experts).

hypotheses were over-fitting the training examples. This was not the case for our algorithm, as the test error steadily decreased until it reached a plateau (although some over-fitting was observed after many stages). The mixture-of-experts algorithm generally out-performed the sequential training algorithms for a small number of experts, but its performance was also not as good as the gradient-based boosting after many stages. The results followed this same pattern for different numbers of hidden units in the neural networks.

6 Conclusion

We have proposed a new boosting algorithm for regression problems. This algorithm shares properties with existing boosting methods for regression, but its collection of key features make it unique. A single common objective function is both used by the weak learning procedure to produce hypotheses, and determines the other parameters in the algorithm. A simple proof guarantees convergence on training error. The distribution of examples is used to control the generation of hypotheses, and each hypothesis is trained to learn the same underlying function. Standard algorithms, such as back-propagation, may be used to produce the hypotheses, training to minimize a weighted cost function based on mean-squared error. And finally, a threshold is used to determine if a response is correct or not.

Empirical investigations have suggested that this algorithm is less prone to over-fitting than another greedy stage-wise approach, the residual-fitting algorithm. Both algorithms, however, will over-fit after a sufficient number of stages. We are currently exploring a regularization technique known as shrinkage, where the combination coefficients are scaled by a parameter less than one. This method has been found to improve generalization in gradient boosting applications to classification (Friedman, 1999). Our empirical studies also suggest that the greedy stage-wise approach can out-perform a simultaneous training method, like the mixture of experts. Generally one would expect a simultaneous approach to be less prone to local minima problems than a greedy one, but an important advantage of the boosting methods is that the number of experts need not be specified in advance.

We are currently exploring an extension of our gradient boosting method, one which brings it a step closer to the full mixture-of-experts algorithm. The combination coefficient c_t associated with hypothesis t can be input-dependent – a learner can return not only $f_t(\mathbf{x}^i)$ but also $k_t(\mathbf{x}^i) \in [0,1]$, a measure of confidence in its prediction. Other sequential regression algorithms do not allow this form of specialization, although it is mentioned in (Freund & Schapire, 1997). This elaboration makes the weak learning task harder, but may extend the range of applicability of the algorithm: letting each learner focus on a subset of its weighted training distribution permits a divide-and-conquer approach to function approximation.

Acknowledgements

This work was funded by ONR N00014-98-1-0509 to RZ; NSF grants CCR-9457782 and CCR-9820831, and US-Israel BSF Grant 95-00238 to TP.

References

[1] Avnimelech, R. & Intrator, N. (1999). Boosting regression estimators. *Neural Computation, 11*, 499-520.

[2] Breiman, L. (1996). Bagging predictors. *Machine Learning, 26*, 123-140.

[3] Drucker, H. (1997). Improving regressors using boosting techniques. In *14th International Conference on Machine Learning*. Morgan Kaufmann.

[4] Freund, Y. & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, 148-156.

[5] Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Comp. and System Sci., 55*, 119-139.

[6] Friedman, J. H. (1991). Multivariate Adaptive Regression Splines. *Annals of Statistics, 19*, 1-82.

[7] Friedman, J. H. (1999). Greedy function approximation: a gradient boosting machine. Technical report, Dept. of Statistics, Stanford University.

[8] Friedman, J. H., Hastie, T., & Tibshirani, R. (1999). Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, To appear.

[9] Grove, A. J. & Schuurmans, D. (1998). Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intellige nce*.

[10] Hastie, T. & Tibshirani, R. (1990). *Generalized Additive Models*. Chapman and Hall.

[11] Impagliazzo, R. (1995). Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science*, 538-545.

[12] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*, 79-87.

[13] Karakoulas, G., & Shawe-Taylor, J. (1999). Towards a strategy for boosting regressors. In *Advances in Large Margin Classifiers*, Smola, Bartlett, Schölkopf & Schuurmans (Eds.).

[14] Klivans, A. R. & Servedio, R. A. (1999). Boosting and hard-core sets. In *40th Annual Symposium on Foundations of Computer Science*, 624-633.

[15] Mason, L., Baxter, J., Bartlett, P., & Frean, M. (1999). Boosting algorithms as gradient descent in function space. In *Advances in Neural Information Processing Systems 11*. Morgan Kaufmann.

[16] Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning, 5*, 197-227.

[17] Schapire, R. E. & Singer, Y. (1998). Improved boosting algorithms using confidence-rated precitions. In *Proceedings of the 11th COLT Conference*.

[18] Schwenk, H. & Bengio, Y. (1998). Training methods for adaptive boosting of neural networks for character recognition. In *NIPS 10*. Morgan Kaufmann, 647-653.

Towards the Mechanical Verification of Textbook Proofs

CLAUS ZINN, *Division of Informatics, University of Edinburgh, 2 Buccleuch Place, Edinburgh, EH8 9LW, Scotland, UK.*
E-mail: zinn@cogsci.ed.ac.uk

Abstract

Our goal is to implement a program for the machine verification of textbook proofs. We study the task from both the linguistics and automated reasoning perspective and give an in-depth analysis for a sample textbook proof. We propose a framework for natural language proof understanding that extends and integrates state-of-the-art technologies from Natural Language Processing (Discourse Representation Theory) and Automated Reasoning (Proof Planning) in a novel and promising way, having the potential to initiate progress in both of these disciplines.

Keywords: text understanding, proof verification, discourse representation theory, proof planning

1 Motivation

In [18], John McCarthy notes that *"Checking mathematical proofs is potentially one of the most interesting and useful applications of automatic computers"*. In the first half of the 1960s, one of his students, Paul Abrahams, implemented a Lisp program for the machine verification of mathematical proofs [1]. The program, named `Proofchecker`, *"was primarily directed towards the verification of textbook proofs, i.e., proofs resembling those that normally appear in mathematical textbooks and journals"*. Abrahams soon revised his goal. If *"a computer were to check a textbook proof verbatim, it would require far more intelligence than is possible with the current state of the programming art"*. Therefore, *"the user must create a rigorous, i.e., completely formalised, proof that he believes represents the intent of the author of the textbook proof, and use the computer to check this rigorous proof"*. Abrahams points further out that *"it is a trivial task to program a computer to check a rigorous proof; however, it it not a trivial task to create such a proof from a textbook proof"*. Abrahams was right. In his implementation and in all later projects, proofs had to be written in a *formal language* using a *restricted set of proof construction commands* in order to verify them. A human user is required to perform the formalisation task.

Basically, two main approaches towards the formalisation and verification of proofs were taken. In the first, the `Automath/Mizar` approach [20, 24], the user is required to give a *full* and *explicit* construction of a formal proof. The proof component then checks the proof for correctness (the compiler approach). A well-known example of a large formalisation is van Benthem-Jutting's translation of Landau's 'Grundlagen der Analysis' into `aut-qe`, one of the formalisms of the `Automath`-language family [26]. The `Mizar` system offers, besides the rich formal language and the proof checker component, a large library of formalised mathematics which allows one to formalise proofs without excessive preparatory work.

In the second approach, the user constructs the proof interactively with the proof system (the interpreter approach). The user has a set of proof construction commands at hand and asks the system to apply them. The system keeps track of the proof obligations and guarantees that the constructed proof is correct. A well-known example of a proof system of this kind is `Nuprl` [8].
For both approaches, proof planning technology could be used to allow the generation of high-level proofs [4]. During proof construction, instead of carrying out elementary steps at the inference level, *tactics* are applied. A tactic is a program that describes the application of a sequence of inference rules. Tactics are formally described by *methods* which specify the preconditions that must hold for a tactic to be carried out. The `Oyster/Clam` system [5] implements an AI plan formation algorithm which seeks to construct a proof plan by searching a tree of methods that combine to prove a given conjecture. These proof plans lead to formal proofs if each method can successfully execute its associated tactic.

Our goal is to mechanise the machine verification of textbook proofs. In the process, we must answer two questions: What formal representation can be obtained from textbook proofs? And what is needed to formally verify these representations? Building a program that checks informal proofs allows us to study natural language understanding in conjunction with automated mathematical reasoning.
From the linguistics point of view, verifying textbook proofs is a text understanding task. The text genre, however, has a number of characteristics that makes this task feasible. The art of writing good mathematical texts focuses on clearness and conciseness and not on an embellished style of expression [27]. The expert language used by mathematicians is stylised, that is, characterised by the use of standard phrases and keywords [25]. The use of terms and formulae introduce formal parts into mathematical discourse. In addition, textbook proofs are, in general, a highly structured form of discourse. A crucial prerequisite to understanding the course of the argumentation within a proof is to identify the discourse relations (e.g., logical consequence) between sentences of that discourse. Deriving the discourse relations of a given textbook proof means reconstructing the intentional structure (describing how sentences within a discourse segment contribute to a common discourse purpose) and the informational structure (describing how sentences within a segment are related to each other by some relation) of the proof [19].
From the automated reasoning point of view, verifying textbook proofs is a deductive task where one must identify the logical structure of the proof: identifying assumptions and conclusions, the scope and quantification of variables, and substructures which themselves form subproofs. Identifying the proof plan of a textbook proof is a prerequisite for verification. It encapsulates informal mathematical reasoning, and therefore, allows us to follow, as close as possible, the proof that was expressed *informally* in the natural language argument. Often, the form of the theorem presupposes possible proof plans and the concepts it contains often hint to definitions being used in the proof. A proof understander must offer a high-level proof analysis as well as a technical low-level access to details of the proof. Gaps and flaws must be recognised.

2 Example

Fig. 1 depicts LeVeque's existence proof of the fundamental theorem of arithmetic [16]. We analyse this proof from both the linguistic and mathematical point of view.

THEOREM 2-2. Every integer $a > 1$ can be represented as a product of one or more primes.

Proof: The theorem is true for $a = 2$. Assume it to be true for $2, 3, 4, \ldots, a-1$. If a is prime, we are through. Otherwise a has a divisor different from 1 and a, and we have $a = bc$, with $1 < b < a$, $1 < c < a$. The induction hypothesis then implies that

$$b = \prod_{i=1}^{s} p_i', \quad c = \prod_{i=1}^{t} p_i'',$$

with p_i', p_i'' primes and hence $a = p_1' p_2' \ldots p_s' p_1'' \ldots p_t''$.

FIG. 1. A textbook proof taken from [16]

2.1 Linguistic analysis

All kinds of linguistic phenomena which can occur in other text genres also occur in textbook proofs [29]. We discuss a selection of problems for the proof at hand.

A prerequisite for parsing textbook proofs is to be able to parse formulae that occur in these proofs. Parsing formulae in isolation is trivial. Problems arise if the textual context has to be taken into account, and when references from the text to formulae and their parts need to be resolved. This is because constants and variables have a domain and scope which extends across text and formulae. A simple interaction between text and formulae is 'every integer $a > 1$'. This construction makes 'every integer' and 'a' denote the same object.

Assigning names to entities plays an important role in mathematical writing. Having names available for variables enables us to show explicitly and concisely which part of an argument depends on other parts of the argument, and which part does not. This allows for disentanglement and decomposition of the argument. However, the improper use of names can also be a source of confusion. An example is LeVeque's proof. The theorem contains the named variable a. However, for the theorem only, it is not necessary to name this variable — it is used only once. Unfortunately, in LeVeque's proof, the name a is used again – suggesting that both occurrences of a refer to the same mathematical entity. Of course, this is not true. The a in the theorem is a universally quantified variable (its denotation is supposed to vary), the a in the proof is a free variable (it denotes a fixed, unknown object). In informal mathematics, frequently, the difference between bound and free variables is blurred.

Other examples of text–formula interactions include (i) 'a divisor' and the introduction of 'b' and 'c' in the proof; and (ii) the recognition that 'with p_i', p_i'' primes' is not about two individual primes but, taking the context introduced by the $\prod$ formulae into account, reads as two universally quantified statements.

The discourse in Fig. 1 contains three obvious referential expressions: 'the theorem', 'it' and 'the induction hypothesis', all of which are of propositional type. The propositional type is either induced by the verb phrase 'is true for' – only propositions can be true; or by 'implies' – only propositions can imply other propositions.

Mathematical discourse is full of conditional claims, and the discourse in Fig. 1 is no exception. The proof contains a number of clue words (e.g., 'if', 'otherwise', 'assume', 'implies', 'hence'), and cue phrases (e.g., 'is true for', 'we are through') that indicate its logical structure. However, to mechanically reconstruct the logical structure of an informal proof, the use of these cues is not sufficient. Consider LeVeque's proof. The third proof sentence, starting with 'if', could be read as follows: introduce the assumption 'a is prime' and conclude 'we are through' (signalling proof termination). However, this sentence has to be embedded into a larger context. In fact, 'if' introduces a proof per cases construct, where 'a is prime' defines the first case, and where the cue phrase 'we are through' terminates this case segment — discharging the assumption. The cue phrase 'otherwise' is an elliptical construction. It initiates the second case and implicitly introduces the assumption 'a is not prime'. The whole remainder of the proof depends on this assumption. The last clause of the discourse 'hence $a = p'_1 p'_2 \ldots p'_s p''_1 \ldots p''_t$' contains the discourse marker 'hence'. Unfortunately, the use of 'hence' does not allow us, linguistically, to identify all the premises (and the inference rules) necessary to conclude '$a = p'_1 p'_2 \ldots p'_s p''_1 \ldots p''_t$'.

2.2 Mathematical analysis

The theorem of Fig. 1 is of the form *For every integer $n \geq n_0$, $P(n)$ holds.* To prove theorems of this logical form, a proof per *Nœtherian induction* (also called *Generalised induction*) can be performed: (i) assume $P(i)$ for $n_0 \leq i < n$; (ii) show that $P(n)$ holds. Knowing this proof method, it is easy to understand LeVeque's proof. The necessity for the first sentence will be explained later. The second sentence states the induction hypothesis. The remainder of the proof constitutes the induction step which itself is structured as a proof per cases: either a is prime or a is not prime. The third proof sentence initiates and terminates the first case. In the second case, forward reasoning from the assumption *a is not prime* takes place. That a has a divisor different from 1 and a follows from definitional expansion of the concept of prime and a proper treatment of negation. Continuing forward reasoning, the statements $a = bc$, $1 < b < a$, and $1 < c < a$ are obtained by definitional expansion of the concept of divisor. The representation of a and b as a product of primes is obtained by applying the induction hypothesis twice (for b using $1 < b < a$, for c using $1 < c < a$). Rewriting $a = bc$ by the product of prime representation of b and c terminates the proof. — The proof makes use of the following definitions and lemmatas:

$$
\begin{aligned}
\forall n \in \mathbb{N} : prime(n) &\Longleftrightarrow n > 1 \wedge \forall d \in \mathbb{N} : d|n \rightarrow d = 1 \vee d = n \\
\forall a \in \mathbb{N} : \forall b \in \mathbb{N} : a|b &\Longleftrightarrow \exists c \in \mathbb{N} : b = ac \wedge a \leq b \wedge c \leq b \\
\forall n \in \mathbb{N} : prod\_primes(n) &\Longleftrightarrow prime(n) \vee \exists p_1, p_2 \in \mathbb{N} : n = p_1 p_2 \wedge p_1 < n \wedge p_2 < n \\
&\qquad \wedge prod\_primes(p_1) \wedge prod\_primes(p_2)
\end{aligned}
$$

$$
\begin{aligned}
\forall a \in \mathbb{N} : \forall b \in \mathbb{N} : a \leq b \wedge \neg(a = b) \rightarrow a < b \\
\forall a \in \mathbb{N} : \forall b \in \mathbb{N} : \forall c \in \mathbb{N} : a = bc \wedge \neg(b = 1) \rightarrow c < a \\
\forall a \in \mathbb{N} : \forall b \in \mathbb{N} : \forall c \in \mathbb{N} : a = bc \wedge \neg(a = b) \rightarrow c > 1
\end{aligned}
$$

In a more rigorous analysis, we used λ-CLAM, a higher-order logic version of the CLAM proof planner [5] to construct a corresponding proof plan. This deeper analysis reveiled a number of steps that have no equivalent in the textbook proof.

3 Architecture

A number of issues influenced the design of our architecture: (i) the semantics of an informal mathematical discourse is its corresponding formal proof; (ii) textbook proofs are a highly structured form of discourse; (iii) the recognition of discourse structure is essential from both the linguistic and mathematical perspective; (iv) proof plans allow us to capture the high-level reasoning of informal proofs and impose proof structure; (v) semantics construction must be supported by a pragmatic component, namely the proof planner which has mathematical and meta-mathematical knowledge.

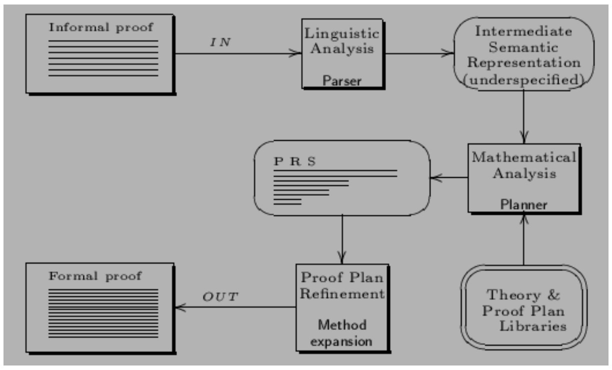


FIG. 2. Architecture of VIP

Fig. 2 depicts the architecture of our system for verifying informal proofs, VIP. VIP gets as input an informal proof. For each sentence of this proof, the **parser** performs a linguistic analysis which results in a set of intermediate semantic representations. These representations are *underspecified* since the grammar underdetermines the content of certain expressions. For example, they contain referential expressions that have to be resolved by subsequent processing since most of the expressions that occur in this intermediate semantic representation are anaphoric. The **proof planner** is the *pragmatics* component; its role is to replace underspecified conditions generated by the grammar with more complete information. The planner incorporates domain knowledge (a library of formalised mathematics) and knows about the uses of argument in mathematical discourse (a library of formalised mathematical reasoning). For each linguistic analysis produced by the parser, the planner determines its full semantic content dynamically by incorporating it in the proof context that has been established by having processed the former sentences. The proof context is represented as a proof representation structure (PRS). It is a hierarchical data structure that consists of discourse entities, discourse statements and discourse markers. Because the planner might be able to incorporate a linguistic analysis in several ways, it has to maintain a set of possible proof continuations. Some of them will be ruled out if they cannot be continued in subsequent processing. If the complete textbook proof has been processed in this manner, the resulting PRS is given to the **proof plan refiner** which expands all proof methods into inference-level steps, such that a formal proof will result. In general, a set of formal proofs will be returned. Each of these formal proofs constitutes one possible semantics for the given informal proof.

3.1 The Parser Module

We have written a definite clause grammar that covers the first proofs of [11], and the proof we discussed above. For semantics construction, our starting point was Kamp's *Discourse Representation Theory (DRT)* [13]. To deal with particular syntactic constructions that occur in mathematical proofs, we modified and added a number of DRT construction rules. For example, one characteristic of mathematical discourse is that mathematicians themselves frequently and explicitly introduce discourse referents. In syntactic constructions like 'let a be an integer > 1', 'a' functions as a discourse referent for the object being introduced by 'an integer'. We therefore defined a DRT construction rule, *naming*, that combines the semantics of 'an integer' (introducing an anonymous discourse referent) with the semantics of the name 'a' (introducing a named discourse referent that anaphorically refers to the object introduced by 'an integer'). This is a simple instance of the problem that variables have a domain and scope which extends across text and formulae.

Another example is to handle the subtle but necessary difference between bound and free variables in mathematical discourse. To compose the semantics of LeVeque's third sentence 'If a is prime, we are through'DRT would apply its construction rule for conditionals. This would result in a universally quantified logical form. However, in the proof context, a must be free. Using the syntactic structure of the sentence alone, it is not possible to determine the domain and scope of variables — one of many requirements to understand mathematical discourse.

Conditionals play a central role in mathematical discourse. In addition to 'if ... then ...', many other forms exist to express the logical relations between arguments. The large number of propositional anaphora makes it necessary to regard mathematical statements as first class citizens of semantics construction. Mathematical statements like 'hence $a = p'_1p'_2 \dots p'_sp''_1 \dots p''_t$' express a logical relation between the assertion object introduced by '$a = p'_1p'_2 \dots p'_sp''_1 \dots p''_t$' and assertion objects introduced by the former discourse. In this sense, 'therefore A' is anaphoric, i.e., it refers to all premises that are necessary to logically conclude A. Mathematically, not all previous statements of a discourse should be accessible. Proofs are a highly structured form of discourse — proofs consist of sub-proofs, and any theory of semantics must reflect this structure.

For all these reasons, we were forced to considerably modify and extend the DRT framework. First, instead of discourse representation structures, we now speak of *proof representation structures*. Second, the simple algorithm for discourse update in standard DRT (basically, a new sentence is incorporated in the discourse context by the union operator) becomes more complex. The proof planner adds a pragmatic component to the discourse update procedure.

3.2 Proof representation structures

Proof representation structures play a central role in our framework[1]. An example PRS for LeVeque's proof is given in Fig. 3 (please ignore the stars for the moment). PRSs contain discourse referents, discourse conditions and discourse markers. Discourse referents are introduced by a LET construct. Conditions are numbered and can be either assumptions or conclusions, the former are prefixed by ASSUME. Each conclusion can be accompanied by a justification list which is a list of numbers, each of which refers to the condition it prefixes. Instead of boxes, in PRS, accessibility is imposed by a numbering scheme. For example, take the condition labelled 2.2.4. It contains the terms b and a. Their quantification is determined by the closest accessible

[1] PRSs are similar in many respects to Lamport-style proof presentations [14].

LET constructs that introduce these names.
Note that the free variable a shadows the universally bound variable a. For justifying 2.2.4 the conditions 2.2.1-2.2.3 and 1. are accessible, the conditions 2.1.1 and 2.1.2 are not accessible. Discourse markers are used to provide non-number referents to either assumptions (e.g., *the induction hypothesis*) or sub-structures (e.g., *the second case*).
For ease of argumentation, the PRS of Fig. 3 is still semantically underspecified. Variables marked with '?' have scope, but no decision has yet been made about their quantification. Also, no conclusion is justified.

LET $a \in \mathbb{N}$ be universally quantified	$*^1$
THEOREM $a > 1 \rightarrow prod\_primes(a)$	$*^1$
Proof per Nœtherian induction	$*^2$
LET $a \in \mathbb{N}$ be arbitrary	$*^2$
LET $k \in \mathbb{N}$ be universally quantified	$*^2$
1. *induction_hypothesis*	$*^2$
ASSUME $k < a \rightarrow (k > 1 \rightarrow prod\_primes(k))$	$*^2$
PROVE $a > 1 \rightarrow prod\_primes(a)$	$*^2$
2. *Proof per cases*	
2.1. *first_case*	
2.1.1. ASSUME $prime(a)$	
2.1.2. $prod\_primes(a)$ BY ? (QED)	
2.2. *second_case*	
2.2.1. ASSUME $\neg prime(a)$	
LET X be ?	
2.2.2. $divisor(X, a)$ BY ?	
2.2.3. $X \neq 1$ BY ?	
2.2.4. $X \neq a$ BY ?	
LET b, c be ?	
2.2.5. $a = bc$ BY ?	
2.2.6. $1 < b < a$ BY ?	
2.2.7. $1 < c < a$ BY ?	
LET $p'_1, p'_2 \ldots, p''_1, p''_2 \ldots \in$ PRIMES be ?	
LET s, t, i be ?	
2.2.8. $b = \prod_{i=1}^{s} p'_i$ BY ?	
2.2.9. $c = \prod_{i=1}^{t} p''_i$ BY ?	
2.2.10. $a = p'_1 p'_2 \ldots p'_i p''_1 \ldots p''_t$ BY ? (QED)	

FIG. 3. A proof representation structure

3.3 The planner module

The proof planner module gets an underspecified semantic representation of an informal proof sentence and tries to incorporate it into the PRS. The proof planner consults two libraries: a library of mathematical knowledge and a library of meta-mathematical knowledge. The theory library contains domain knowledge, basically, a set of definitions and theorems. Our experience showed that it is useful to have multiple definitions for the same concept. The proof plan library contains a collection of proof plans (more precisely: proof plan schemas).[2] Quite often, the form of the theorem presupposes more than one applicable proof plan. In other cases, the theorem can often be transformed by expanding one of its concepts (definitional rewriting).
Fig. 4 depicts one of the applicable proof plans for theorems of the form $\forall n \in \mathbb{N} : P(n)$. It introduces two variables (a free and a bound one), an assumption, a discourse marker, and a remaining proof obligation. For illustrating the PRS construction algorithm, assume an empty PRS, and that the parser returns the following representation for the first sentence of Fig. 1:

LET $n \in \mathbb{N}$ be universally quantified
PROVE $P(n)$
Proof per Nœtherian induction
LET $n \in \mathbb{N}$ be arbitrary
LET $k \in \mathbb{N}$ be universally quantified
induction hypothesis
ASSUME $k < n \rightarrow P(k)$
PROVE $P(n)$

FIG. 4. Nœtherian induction

$[[a, 1|a > 1] \rightarrow [\_|prod\_primes(a)]]$. Since the PRS is empty, the planner assumes it's reading the semantic representation for a theorem (i.e., it assumes a being universally quantified). It adds the enriched semantic representation into the PRS. Now, we have a PRS that consists of the top two-lines of the PRS as depicted in Fig. 3. Instead of interpreting the underspecified representation of LeVeque's first proof sentence, the proof planner tries to figure out future proof continuations. Consulting the proof

[2]However, we cannot hope that any proof plan library is complete. There is a potentially infinite number of different induction schemes, and creative mathematicians often combine existing methods in a novel way. Learning new plans from the content of discourse is a difficult task — also for the human reader!

plan library, applicable proof plans are identified, amongst them the proof plan shown in Fig. 4. Instantiating this proof plan for the theorem at hand results in the lines marked '$*^2$' of the PRS in Fig. 3. These starred lines define a number of expectations, and therefore allow us to view remaining informal statements as anaphoric entities that refer to their place in the PRS.

Unfortunately, the underspecified semantic representation of the first proof sentence of Fig. 1 cannot be matched to the proof plan. For $a = 2$, we have that $\forall k < a : [k > 1 \rightarrow product\_primes(k)]$ is trivially true since there is no k such that $k < a$ and $k > 1$. Therefore, the case $a = 2$ is treated separately.

The second sentence can be matched to one of our expectations if we give to the elliptic construct '$2, 3, 4, \ldots, a - 1$' a universal reading: $\forall n \in \mathbb{N} : n < a$. Note that the a that occurs in this expression refers to the free PRS variable a, and not to the universally quantified PRS variable a that was introduced by processing the theorem.

The third sentence contains the discourse marker 'if' indicating an assumption. Since we cannot match this assumption with the PRS expectations, the proof planner consults the plan library. Unexpected assumptions are often of the form $A \vee \neg A$ (here, A is 'a is prime'), and a proof per cases is likely to take place. The proof planner adds this to the PRS as one possible proof continuation. This allows us to resolve 'if a is prime'. The constituent 'we are through' marks the end of this first case leaving the proof planner with the obligation to resolve the logical relation between $prod\_primes$ (2.1.2) and all statements of the proof context that are accessible from 2.1.2. Note that, due to a wrong proof continuation, we might not be able to identify this logical relation. The occurrence of *otherwise* in the fourth sentence seems to confirm a proof per cases. The remainder of the textbook proof has to fit into the second case yielding to **2.2.2** to **2.2.10**.

The PRS is in Fig. 3 is still semantically underspecified: most of the variables do not have quantifications yet and all conclusions are still unjustified. The proof planner must invest much inference to validate its structure and to compute the necessary information to fill in the semantic gaps.

For step **1.** of the PRS the remaining proof obligation, or *goal*, is of the form $A \rightarrow B$. To prove statements of this form, the *implication method* can be applied: one assumes A and then uses A to show B. This inference is not explicitly part of the textbook proof. It must be derived to facilitate subsequent discourse processing. The proof planner extend the PRS by the lines 'ASSUME $a > 1$' and 'PROVE $prod\_primes(a)$'.

Justifying **Step 2.1.2.** involves a couple of inference steps. *Definitional rewriting* on $prod\_primes$ reduces **2.1.2** to

$$prime(a) \vee \exists p_1 \in \mathbb{N} : \exists p_2 \in \mathbb{N} : a = p_1 p_2 \wedge p_1 < a \wedge p_2 < a \wedge prod\_primes(p_1) \wedge prod\_primes(p_2).$$

Having now a disjunction in the goal, a *proof per elimination* can be performed. There are two possible proof plan schemes (Fig. 5). Applying *proof per elimination-II*, we obtain a new hypothesis (in addition to the induction hypothesis, $a > 1$ and **2.1.1**):

PROVE $A \rightarrow B1 \vee B2$	PROVE $A \rightarrow B1 \vee B2$
Proof per elimination-I	*Proof per elimination-II*
ASSUME A	ASSUME A
ASSUME $\neg B1$	ASSUME $\neg B2$
PROVE $B2$	PROVE $B1$

FIG. 5. Proof per elimination

$$\neg(\exists p_1 \in \mathbb{N} : \exists p_2 \in \mathbb{N} : a = p_1 p_2 \wedge p_1 < a \wedge p_2 < a \wedge product\_primes(p_1) \wedge product\_primes(p_2))$$

and a new goal: $prime(n)$. Since the goal is contained in the hypotheses, we are through. — Due to space restrictions, we omit a detailed analysis of the second case.

3.4 Proof plan refinement.

The step from proof plans to formal proofs is a simple problem. It involves no search because to each large proof step, we can attach its expansion into inference-level steps. Take the proof method of Nœtherian induction (Fig. 4). It is formally defined as (3.1).

$$\forall n \in \mathbb{N} : [(\forall k \in \mathbb{N} : k < n \rightarrow P(k)) \rightarrow P(n)] \rightarrow \forall n \in \mathbb{N} : P(n). \tag{3.1}$$

To prove a formula of the form $\forall n \in \mathbb{N} : P(n)$, we prove

$$\forall n \in \mathbb{N} : [(\forall k \in \mathbb{N} : k < n \rightarrow P(k)) \rightarrow P(n)] \tag{3.2}$$

and apply Modus Ponens using (3.1) and (3.2). To prove (3.2), we can assume n being an arbitrary number and assume $\forall k \in \mathbb{N} : k < n \rightarrow P(k)$. These steps are on the inference level of a Gentzen-style calculus [9].

4 Related and Future Work

The automated verification of textbook proofs has long been ignored. The only citable reference is Simon's Phd thesis [22]. However, Simon fails to seriously address both linguistic and mathematical issues. It remains unclear how Simon handles linguistic phenomena – he does not propose an adequate theory for doing so. It remains also unclear how Simon is able to obtain adequate discourse structure. We regard this as an essential requirement for following the proof author's argumentation line and for subsequent formalisation.

The inverse of our task, translating formal proofs into natural language proofs, has first been described in [7]. More recent work is [12]. Facing the problem that proofs generated by conventional theorem provers are unstructured, tediously long and therefore unreadable, a readable, structured and short proof (omitting all the low-level details) has to be generated. Finding appropriate abstractions from low-level proofs is a hard problem.

Proof planning is an enabling technology for constructing and representing high-level proofs [4]. Proof plans allow us to capture informal mathematical reasoning, and enable us to fill in the details that mathematicians find trivial to write down. Our task to mechanically verify mathematical discourse forces us to closer examine existing mathematical proofs and methods of proof. We share Wang's opinion that *"the existing body of mathematics contains a great wealth of material and constitutes the major source of our understanding of mathematical reasoning. The reasonable course would be to distill from this great reservoir whatever is mechanizable"* [28]. This is a promising approach to make proof machines more usable.

Our work also relates to extensions of DRT, in particular Asher's SDRT [2]. SDRT discourse update mechanism allows to effectively interface semantics with pragmatics. The discourse update procedure consists of *relating* a sentence to its prior discourse (not just adding it). To compute such rhetorical relations (e.g., background, narration, elaboration), Asher and Lascarides make use of domain knowledge and their DICE inference engine [15, 3].

Given the enormous complexity of the entire problem, much implementation work is to be done to complete VIP. However, our definite clause grammar covers already a considerable amount of mathematical text on the sentence level. For this level, DRT has been extended to account for four different kinds of discourse referents:

constants, free variables, quantified variables, and functional terms. A number of DRT construction rules have been implemented that capture typical syntactic constructions in informal mathematical text.

For the discourse level, we extended discourse representation structures to proof representation structures. A novel algorithm for discourse update has been developed. Its core is the proof planner which takes the intermediate semantic representations returned by the parser module, and interprets them in the current proof context. In the future, the discourse update algorithm has to be fully implemented and tested on a substantial number of examples. We would also like to interface to the λ-CLAM proof planning system that we have already used to manually generate formal proofs from informal proofs. This would give us access to a large number of already formalised proof plans and λ-CLAM's proof planning engine.

The view of proofs as highly structured discourse is compelling. Linguistically, we would like to study how our way of handling discourse structure compares with existing discourse models like [10]. Another related line of research is to view the problem of discourse understanding as a plan recognition problem [17],[6], [21]. The proof planner we described certainly generates expectations and tries to match them later. But proof recognition goes hand in hand with constructive planning. The relation between proof construction and plan recognition has to be worked out in greater detail.

5 Conclusion

We proposed an architecture for the automatic verification of informal proofs (mathematically); constructing the semantics and pragmatics of informal mathematical discourse (linguistically). We argued that the recognition of structure plays a decisive role. It enables us to follow the argumentation line of the proof author, and to give us access to a high-level strategic proof understanding. Linguistically, discourse structure allows us to properly handle linguistic phenomena. We argued that proof plans not only provide for such structure; they also generate expectations about possible proof continuations.

The design of VIP extends and integrates state-of-the-art technologies from Natural Language Processing (Discourse Representation Theory) and Automated Reasoning (Proof Planning) in a novel and promising way, For representing mathematical discourse, we introduced proof representation structures as the central data structure. PRSs are a considerable extension to discourse representation structures accommodating our need for representing discourse structure, and handling substructures and mathematical sentences as first-class citizen. For constructing PRSs, we proposed a discourse update algorithm that is powered by pragmatics. A proof planner incorporates an underspecified semantic representation into the proof context by making use of mathematical and meta-mathematical knowledge.

The automatic checking of mathematical textbook proofs is a challenging application scenario. For a long time, human intervention will be necessary to direct theorem provers through the enormous search space of mathematical tautologies. Being able to process textbook proofs allows us to close the gap between the language and the reasoning of mathematicians, and the language and reasoning of proof systems. It is appealing to think of a *natural* language interaction between mathematician and machine. It will enable us to build systems that really assist mathematicians.

References

[1] P. W. Abrahams. *Machine verification of mathematical proofs.* PhD thesis, MIT, 1963.

[2] N. Asher. *Reference to abstract objects in discourse.* Kluwer Academic Publishers, 1993.

[3] N. Asher and A. Lascarides. The semantics and pragmatics of presupposition. *Journal of Semantics*, 15(3):239–300, 1998.

[4] A. Bundy. The use of explicit proof plans to guide inductive proofs. In R. Lusk and R. Overbeek, editors, *Ninth Conference on Automated Deduction*, pages 111 – 120. Springer, 1988.

[5] A. Bundy, F. van Harmelen, C. Horn, and A. Smaill. The Oyster-Clam system. In M. E. Stickel, ed., *Proceedings of the 10th International Conference on Automated Deduction.* Springer, 1990.

[6] S. Carberry. Modeling the User's Plans and Goals. *Comp. Linguistics*, 14(3):23–37, 1988.

[7] D. Chester. The Translation of Formal Proofs into English. *Artificial Intelligence*, 7:261–278, 1976.

[8] R. L. Constable. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, 1986.

[9] G. Gentzen. Untersuchungen über das logische Schliessen. *Mathematische Zeitschrift*, 39:176–210, 1935.

[10] B.J. Grosz and C.L. Sidner. Attention, Intention, and the Structure of Discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[11] G. Hardy and E. Wright. *An introduction to the theory of numbers.* Oxford at the Clarendon Press, 4th. edition, 1971.

[12] X. Huang and A. Fiedler. Proof verbalization as an application of NLG. In M. E. Pollack, editor, *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 965–970. Morgan Kaufmann, 1997.

[13] H. Kamp and U. Reyle. *From Discourse to Logic.* Kluwer Academic Publishers, 1993.

[14] L. Lamport. How to write proofs. In *Global Analysis in Modern Mathematics*, pages 311–321. Publish or Perish, Houston, Texas, U.S.A., February 1993.

[15] A. Lascarides and N. Asher. Discourse relations and defeasible knowledge. In *Proceedings of the 29th Annual Meeting of the Assoc. for Computational Linguistics*, pages 55–63. ACL, 1991.

[16] W. J. LeVeque. *Elementary theory of numbers.* Addison-Wesley, 1965.

[17] D.J. Litman and J.F. Allen. Discourse processing and commonsense plans. In P.R. Cohen, J. Morgan, and M.E. Pollack, eds., *Intentions in communication*, p. 365–388. MIT Press, 1990.

[18] J. McCarthy. Computer programs for checking mathematical proofs. In *Recursive Function Theory, Proceedings of Symposia in Pure Mathematics*, volume 5. AMS, 1962.

[19] J.D. Moore and M.E. Pollack. A Problem for RST: The Need for Multi-Level Discourse Analysis. *Computational Linguistics*, 18(4), 1992.

[20] R. P. Nederpelt, J.H. Geuvers, and R.C. de Vrijer, editors. *Selected papers on Automath*, volume 133 of *Studies in Logic and the foundations of Mathematics.* North-Holland, 1994.

[21] C.F. Schmidt, N.S. Sridharan, and J.L. Goodson. The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence. *Artificial Intelligence*, 11:45–83, 1978.

[22] D. L. Simon. *Checking Number Theory Proofs in Natural Language.* PhD thesis, Austin, 1990.

[23] D. Solow. *How to read and do proofs.* John Wiley & Sons, 1990.

[24] Andrzej Trybulec. The MIZAR-QC/6000 logic information language. *ALLC*, 6:136–140, 1978.

[25] J. Trzeciak. Writing mathematical papers in english. Gdańsk Teacher's Press, Institute of Mathematics, Polish Academy of Science, 1993.

[26] L.S. van Benthem Jutting. *Checking Landau's "Grundlagen" in the Automath system.* PhD thesis, TH Eindhoven, 1977.

[27] A.J.M. van Gasteren. *On the shape of mathematical arguments*, volume 445 of *Lecture Notes in Computer Science.* Springer, 1990.

[28] H. Wang. Towards Mechanical Mathematics. *IBM Journal of research and development*, 4(1), 1960.

[29] C. Zinn. Understanding Mathematical Discourse. In *Proceedings Amstelogue'99, Workshop on the Semantics and Pragmatics of Dialogue.* Amsterdam University, 7-9 May 1999.