Linear Logic and Applications

Valeria de Paiva, Josef van Genabith, Eike Ritter and Richard Crouch

September 30, 1999

1 Introduction

Linear Logic was introduced by J.-Y. Girard in 1987, and has attracted much attention from computer scientists as a logical way of coping with resources and resource control. The basic idea of Linear Logic is to consider formulae in a derivation as resources, which can be consumed or produced. To realize this basic idea we consider a logical system where the duplication and/or erasing of formulae must be explicitly marked using a unary operator, called an exponential or a modality. Rather than an alternative to Classical Logic, Linear Logic is a refinement of it: using Girard's well-known translation we can investigate the usage of resources in proofs of the traditional theorems. Linear Logic shares with Intuitionistic Logic a well-developed proof-theory and shares with Classical Logic the involutive behavior of negation, which makes its model theory particularly pleasant.

More importantly, because resource control is a central issue for Computer Science, Linear Logic has been applied to several areas within it, including functional programming, logic programming, general theories of concurrency, syntactic and semantic theories of natural language, artificial intelligence and planning. Several sessions in prestigious conferences like LiCS (Logic in Computer Science) as well as whole conferences (Cornell 1993, Cambridge 1995, Tokyo 1996, Luminy-Marseille 1998) have been devoted to Linear Logic, a measure of the penetration of these ideas in the community.

2 Report

The Dagstuhl seminar Linear Logic and Applications contained talks concerning a number of different application areas of linear logic, as well as talks on more foundational issues.

(a) There was a large representation of linguistic applications. A repeated theme was a need for proof search methods that could efficiently un-

cover all distinct proofs of a given formula from a set of premises; distinct proofs either corresponding to distinct parses of a sentence, or distinct semantic interpretations of a parse. Many of the linguistic applications employed some version of categorial grammar, which through the Lambek calculus bear close connections to linear logic. However, the applications were not limited to this one style of linguistic theory: there were presentations on how linear logic could account for the resource sensitive nature of minimalism, how it could be used for rewrite rules in machine translation, and how categorial semantics could be combined with other grammatical theories. There was additionally discussion of how proof nets could account for observed processing costs in different types of linguistic construction. There was also discussion of how Linear Logic could be used to encode pragmatic distinctions (in natural language meanings), e.g. between assertions of fact, obligations.

(b) Applications to verification and specification. There were two presentations on using linear logic for formal specification and verification. Both stated that linear logic appeared to provide a compact and intuitive representation for a variety of problems, as compared to many other logical specification languages. However, the need for a combined linear and temporal logic was clearly felt.

(c) There were also presentations on the application of linear logic to functional programming. There has been a longstanding hope that the resource/usage counting aspects of linear logic could be used to allow efficient memory management / garbage collection in functional languages. Presentations showed how this goal could be met in a number of useful special cases, but higher-order functions continue to raise problems for the more general goal.

(d) Semantics of linear logic. The seminar included talks on game semantics for Linear Logic as well as semantic spaces useful for linguistic representations.

(e) Proof theory, search, complexity and syntax. Talks on proof search and checking included a discussion of potentially incremental connection methods, and the presentation of a linear time algorithm for multiplicative linear logic. There was also a talk on the complexity of proofs, using their graph-theoretical properties and also presentations on the treatment of quantifiers as infinite tensors and pars, as well as the significance for various translations / embeddings between logics.

(f) The semantics of resources and bunched implications. The seminar also contained three presentations that, collectively, argued that linear logic should not be viewed as "the" logic of resources. Thus while

> Linear Logic may serve well as a process logic, and as providing a better understanding of proof and the nature of some familiar connectives, it was argued that it fell short in its account of resources. It was argued instead that a logic of bunched implication grew more naturally out of considering the nature of resources.

It is probably fair to say that the (negative) claims about the resource sensitivity of linear logic provoked the most controversy in the meeting. While there was perhaps a fairly general consensus that linear logic should not be seen as the only logic of resources, there was no similarly general agreement that it was not a resource logic at all, nor that it was thereby devoid of interest, e.g. in shedding light on proof theory / cut-elimination. Also, to many of the participants working at the applied end, much of the dispute had the air of an internal argument, where it was unclear whether or not the results would have far-reaching consequences for many of the more practical applications.

At a more informal level, the meeting succeeded admirably in bringing together people from a variety of different backgrounds, and with different expectations of linear logic, and provoking lively, friendly and productive discussion.

3 Programme of the Seminar

Monday, 23 August

Mark Hepple	*Linear Logic, Tree Descriptions, and Early Deduction*
Alessandra Carbone	*On the Complexity of Proofs*
David Pym	*The Logic of Bunched Implications*
Luiz Carlos Pereira	*Translations and Normalization: applications to Linear Logic*
Gianluigi Bellin	*A pragmatic interpretation of Substructural Logics*
Dick Crouch	*Linear Logic for Natural Language Semantics: skeletons, modifiers and proofs with identities*

Tuesday, 24 August

Uday Reddy	*Linear Logic, SCI and Resources*
Eike Ritter	*Linear Abstract Machines with the Single Pointer Property*
Peter O'Hearn	*On Resource Semantics of Bunched Implications*
Martin Hofmann	*In place update with linear types or how to compile functional programs into malloc()-free C*
Geert-Jan Kruijff	*Resource Logics, Natural Language Grammar and Processing*
David Sinclair	*Specifying and Verifying Protocols using Linear Logic*

Wednesday, 25 August

Andrea Schalk	*Abstract Games for Linear Logic*
Glyn Morrill	*Proof Nets as Syntactic Structure of Language*

Thursday, 26 August

Hermann Haeusler	*Infinitary Linear Logic and Multiplicatively Quantified Linear Logic*
Francois Lamarche	*Spaces for the Semantics of Linear Logic and Linguistic Representation*
Philippe de Groote	*Towards a Generalisation/Implementation of Type Logical Grammars*
Christian Retoré	*Linear Logic and Chomsky's Minimalist programme for Linguistics*
Sara Kalvala	*A Substructural Logic for Formal Verification*
Bertram Fronhoefer	*Proof Search in Multiplicative Linear Logic*

Friday, 27 August

Tsutomu Fujinami	*A Decidable Linear Logic for Speech Translation*
Akim Demaille	*Hybrid Linear Logics*
Luke Ong	*A Linear-Time Algorithm for Verifying MLL Proof Nets via Lamarche's Essential Nets*

4 Abstracts of the talks

Linear Logic, Tree Descriptions, and Earley Deduction

Mark Hepple,
University of Sheffield

This talk draws parallels between a number of logical and grammatical formalisms, with a view to adapting ideas and methods available for one approach to some of the others. The approach of categorial grammar is firstly introduced, as well as a specific categorial system, the (product-free) Lambek calculus. Then, the relationship between the Lambek calculus and the implicational fragment of linear logic is discussed, and it is shown how the latter can be used as a basis for implementing the former by use of the labelled deduction methodology of Gabbay. We then discuss the relationship between the Lambek calculus and grammatical representations which employ tree descriptions. This leads to a demonstration that Lambek deductions can be compiled to a variant of the formalism Multiset-valued Linear Indexed Grammar, in which categories are augmented with labels that capture the Lambek system's ordering constraints. This compilation allows for an efficient deduction method which is related to Earley's top-down, predictive, chart parsing algorithm. This approach is readily adapted to provide an efficient tabular deduction method for implicational linear logic.

Linear Logic for Natural Language Semantics: skeletons, modifiers and proofs with identities

Dick Crouch,
Xerox Palo Alto Research Center

We start by briefly reviewing the 'glue language' approach to the semantic interpretation of natural language [1]. This uses an (implicational) fragment of linear logic to assemble the meanings of words and phrases in a syntactically analysed sentence. This raises an important, but neglected, proof search problem of relevance to computational linguistics.

From a given set of premises (representing lexical items), and with a given conclusion in mind (representing an analysis of the full sentence), there is typically a multitude of distinct normal form derivations, each assigning a distinct proof/meaning term to the sentence. Many applications in computational linguistics require exhaustive proof search, where all alternative derivations are found, and as efficiently as possible.

In the case of glue language derivations, it turns out that all alternative derivations arise through different ways of introducing modifiers into a single skeleton derivation. In logical terms, modifiers correspond to identities of the form $\phi \multimap \phi$ (for atomic or complex formulas ϕ). Given this feature of glue language derivations, the following three questions are addressed

1. How can normal form modifier derivations concluding in identities, $\Gamma_1 \vdash \phi \multimap \phi$, be inserted into normal form skeleton derivations, $\Gamma_2 \vdash \psi$, to create all possible normal form derivations, $\Gamma_1, \Gamma_2 \vdash \psi$?

2. How can modifier derivations be separated out from skeleton derivations in the first place?

3. Why might this 'skeleton-modifier' approach to derivations be computationally efficient, especially when combined with standard tabular / Earley deduction techniques

References

[1] Dalrymple, M. (ed). 1999. *Semantics and Syntax in Lexical Functional Grammar: the resource logic approach*, MIT Press.

[2] Lamping, J. & V. Gupta. 1998. *Efficient Linear Logic Meaning Assembly*, Proc 36th ACL.

Resource Logics, Natural Language Grammar and Processing

Geert-Jan Kruijff
Charles University, Prague

This talk consists of three parts. The first part reviews the use of substructural logics in formal approaches to natural language grammar and semantics. In particular, labelled deductive systems of the form $\langle$ label, formula $\rangle$ are considered as a means to model substructural logics. Employing different algebras over the labels that guide the inference over the formulas they are attached to, a whole landscape of substructural logics arise that spans from the non-associateve Lambek calculus NL (weakest) to the Lambek-van Benthem calculus LP (strongest). t The latter has a correspondence to multiplicative linear logic (MLL).

In the second part of the talk a tabular deduction method is presented. Its aim is to provide an efficient means of processing with grammars that are modelled as LDS-style substructural logics. The method is based on a method proposed by Hepple (see also talks by Hepple, Crouch). The method presented here extends Hepple's method by including the means for handling directional implications, structural rules and many modals. Finally in the third part we raise several questions regarding various formal characteristics we would like to establish for complex systems in which substructural logics of various strengths coexist and interact.

Linear logic and Chomsky's minimalist programme for linguistics

Christian Retoré
INRIA-Rennes

This talk is meant to explain the growing interest in the connection between resource-logical theories of grammar and the minimalist grammars of the transformational tradition in syntax. Although there are substantial differences that can be debated, the prospects also look good for identifying a valuable common ground. In particular, the rich descriptive tradition of transformational theory may become more accessible to resource-logical frameworks, and the frameworks may stimulate a more sophisticated understanding of the mechanisms of minimalist grammar, in particular from a computational viewpoint (parsing and learning algorithms). Linear logic is an appealing framework for a logical treatment of minimalist grammars; indeed linear logic is a neat and well studied logic from a proof theoretical perspective which is able to handle both resource logic for syntax (like the

Lambek calculus) and logic for semantics (like intuitionistic proofs depicting formulae of usual predicate calculus used in Montague semantics).

I will first survey the generative perspective on language. — a possible reference is [4] by J.-Y. Pollock which presents various stages of the Chomskyan theories. I will insist on Chomsky's minimalist program for linguistics [5], which gets closer to Lambek's categorial grammars:

- grammars are assumed to be fully lexicalized
- lexical entries are made of features (categorial, or functional) which are "linguistic resources".
- all the operations of the grammar are driven by feature consumption,
- the two generative operations MERGE (the combination of phrases) and MOVE are triggered by the consumption of categorial features.

Nevertheless, the difference lies in the presence of phrasal movement for producing the phonological form and the logical form of a sentence.

Then I will present Stabler's formal (tree) grammars [1] which formalize these ideas, and finally show how they can be encoded in plain Lambek calculus, plus a simple automaton, as done in [2]. The simple automata is used after the Lambek proof is constructed in order to compute word order and logical form. The key point is to use product elimination (that is "par" introduction) for encoding movement: during the computing of the labels (phonological and logical forms) from an "ordinary" Lambek proof, this rule is used to copy the label of the main premise to the two discharged hypotheses. The strength of the features which are discharged explain cross language variation, as in Chomsky's minimalist program.

Acknowledgement: Edward Stabler (UCLA, Los Angeles) gave the first formalization of minimalist grammars, the one we are using [1]. The encoding of Stabler's minimalist grammars is from a recent paper with Alain Lecomte (UPMF, Grenoble) [2]. The general relation between these two fields is taken up again from a recent paper with Edward Stabler on this connection [3].

References

[1] Edward Stabler. Derivational minimalism. In C. Retoré , editor, Logical Aspects of Computational Linguistics, LACL'96, volume 1328 of LNCS/LNAI. Springer-Verlag, 1997. pages 68–95.

[2] Alain Lecomte and Christian Retore'. Towards a minimal logic for minimalist grammars: a transformational use of Lambek calculus. In G.-J. Kruijff and Richard T. Oehrle, editors, Formal Grammar '99. FoLLI, 1999. pages 83–92.

[3] Christian Retoré and Edward Stabler, Resource Logics and Minimalist Grammars, Introduction to the workshop, European Summer School in Logic Language and Information, Utrecht, 1999. FoLLI.

[4] Jean-Yves Pollock. Langage et cognition: le programme minimaliste de la grammaire générative. Presses Universitaires de France, Paris, 1997.

[5] Noam Chomsky. The minimalist program. MIT Press, Cambridge, MA, 1995.

Towards a Generalisation/Implementation of Type Logical Grammars

Philippe de Groote
INRIA - LORIA

We review the notions of Lambek Categorial Grammars [1], of Montague Grammars [2], and of Type Logical Grammars [3]. Then we propose a general framework, inspired by these different formalisms.

In this new framework, an abstract lexicon is simply a higher-order linear signature, and a concrete view of such a lexicon is a realization of the signature by means of linear λ-terms. This scheme allows several views of a same lexicon to be defined: typically, a syntactic view and a semantic view. We argue that the resulting formalism is well-suited for an implementation.

References

[1] J. Lambek. The mathematics of sentence structure. *Amer. Math. Monthly*, 65:154–170, 1958.

[2] R. Montague. The proper treatment of quantification in ordinary english. In J. Hintikka, J. Moravcsik, and P. Suppes, editors, *Approaches to natural language: proceedings of the 1970 Stanford workshop on Grammar and Semantics*, Dordrecht, 1973. Reidel.

[3] G.V. Morrill. *Type Logical Grammar*. Kluwer Academic Publishers, Dordrecht and Hingham, 1994.

Proof Nets as Syntactic Structures of Language

Glyn Morrill,
Universitat Politècnica de Catalunya

Type Logical Grammar (Morrill 1994; Moortgat 1996; Carpenter 1998) is a theory of categorial grammar which defines the relation of syntactic formation logically, by an interpretation of categorial types, rather than by stipulation of a formal system. The logic is generally some variety of non-commutative, probably intuitionistic, linear logic, paradigmatically Lambek's Syntactic Calculus. Since derivations are proofs, computational concerns raise the question: what is the essential structure of proofs? To which a powerful answer is: proof nets, leading to the slogan 'proof nets as syntactic structures of language'.

We have introduced methods based on proof nets addressing four aspects of categorial processing: tabularization of Lambek proof nets (Morrill 1996), type logical generation (Merenciano and Morrill 1997), preevaluation of lexico-syntactic interaction (Morrill 1999) and incrementality and best-first resolution of local and global ambiguity (Morrill 1998). The first and second are not presented here. The third observes that if syntactic derivation and lexical semantics are represented uniformly as proof nets, the substitution of lexical semantics into derivational semantics, usually postevaluated by on-line lambda conversion, can be preevaluated by off-line lexical Cut-elimination 'getting rid of lambda conversion once and for all'. The fourth observes that a wide range of psycholinguistic processing phenomena are explained on the plausible assumption that incremental processing resolves axiom link dependencies as soon as possible.

References

[1] Carpenter, 1998. *Type-Logical Semantics*, MIT Press, Cambridge, Mass.

[2] Moortgat, 1996. *Categorial type logics*, in van Benthem and ter Meulen (eds.) *Handbook of Logic and Language*, Elsevier, Amsterdam, 93-177.

[3] Morrill, 1994. *Type Logical Grammar: Categorial Logic of Signs*, Kluwer, Dordrecht.

[4] Morrill, 1996. *Memoisation of categorial proof nets: parallelism in categorial processing*, Report LSI-96-24-R, UPC. To appear in *Computational Logic for Natural Language Processing*, Springer, Berlin.

[5] Merenciano and Morrill, 1997. *Generation as deduction on labelled proof nets*, in Retoré (ed.) *Log. Asp. Comp. Ling. '96*, Springer, Berlin, 310-328.

[6] Morrill, 1998. *Incremental Processing and Acceptability*, Report LSI-98-46-R, UPC.

[7] Morrill, 1999. *Geometry of Lexico-Syntactic Interaction*, 9th Conf. Europ. Assoc. Comp. Ling., Bergen, 61-70.

A Decidable Linear Logic for Speech Translation

Tsutomu Fujinami,
School of Knowledge Science, JAIST

The structure of objects employed in the study of Natural Language Semantics has been increasingly complicated to represent the items of information conveyed by utterances. The complexity becomes a source of trouble when we employ those theories in building linguistic applications such as speech translation systems. To understand better how programs operating on semantic representations work, we adopt a logical approach and present a monadic and multiplicative linear logic. In designing the fragment, we refine on the multiplicative conjunction to employ both the commutative and non-commutative connectives. The commutative connective is used to glue up a set of formulae representing a semantic object conjointly. The non-commutative connective is used to glue up a list of formulae representing the argument structure of an object, where the order matters. We also introduce to our fragment a Lambek slash, the directional implication, to concatenate the formula representing the predicative part of the object and the list of formulae representing the argument part. The monadic formulae encode each element of the argument part by representing its sort as the predicate and the element as the place-holder. The fragment enjoys the nice property of being decidable. To encode contextual information involved in utterances, however, we extend the fragment with the exponential operator. The context is regarded as a resource available as many times as required, but not infinitely many. We encode the items of context with the exponential operator, but ensure that the operator should appear only in the antecedent. The extension keeps the fragment decidable because proof search will not fall into an endless search caused by the coupling of unlimited supply and consumption. We show that the fragment is rich enough to encode and transform semantic objects employed in the contemporary linguistic theories. The result guarantees that the theories on natural language semantics can be implemented reasonably and safely on computers.

A pragmatic interpretation of Substructural logics

Gianluigi Bellin (Univ. Verona) and
Carlo Dalla Pozza (Univ. Lecce)

In the philosophical approach proposed in Dalla Pozza [1995, 1997] Tarski's semantics for classical propositional logic is compatible with Heyting's interpretation of intuitionistic connectives, if the former applies to the logic of propositions and the latter is regarded as an informal model for the logic

of judgement and of obligation. Namely, a language $\mathcal{L}^P$ is proposed where Heyting's interpretation holds of formulas with assertive pragmatic force, deontic pragmatic force is interpreted as a modal operator and Tarski's semantics applies to radical formulas, without a sign of pragmatic force. We present here a sequent calculus that formalizes reasoning with mixed assertive and deontic contexts: these are essential to formalize fundamental schemes in legal reasoning such as conditional obligations ("if A is forbidden and it is assertible that A, then sanction B is obligatory"). The guiding principle of our formalization is Hume's law ("we cannot derive an 'ought' from an 'is' and vice-versa"). Taking this principle seriously forces the use of a mixed relevant and ordinary intuitionistic consequence relation. Finally, we conclude that linear logic is the logic of pragmatic schemes where the sign of pragmatic force is unspecified, and such that the scheme remains valid under the widest range of substitutions of assertive or deontic signs of force for the unspecified ones.

References

[1995] C. Dalla Pozza and C. Garola. A pragmatic interpretation of intuitionistic logic. *Erkenntniss* 43, 1995.

[1997] C. Dalla Pozza. Una logica prammatica per la concezione 'espressiva' delle norme, in A. Martino *Logica, Informatica, Diritto, in onore di Carlos Alchourron*, Pisa 1987.

Specifying and Verifying Protocols using Linear Logic

David Sinclair,
City University Dublin

Communications protocols are significant elements of today's electronic communications infrastructure. The ability to formally specify these protocols, and the ability to verify their properties is essential. Many formalisms have been used. Some are more suited to specification, others are more suited to verification. This talk advances Linear Logic as a formalism that is suited to both the specification and verification of protocols. The TCP/IP protocol is taken as an example. A Linear Logic specification of the interface to the IP and TCP layers is presented, as well as a specification of the data transfer portion of the protocol. Properties of the IP are proved. The composition of the specifications of the IP layer interface and the data transfer protocol is shown to have the same properties as the TCP layer interface, namely that data is transferred as a stream of octets. The approach leads to relative simple specifications that enable verification. However this work has indicated that the addition of temporal reasoning to Linear Logic would be a great advantage.

A Substructural Logic for Formal Verification

Sara Kalvala,
University of Warwick

While several toy examples, such as in the "blocks world" or concerning cigarettes and chocolates, have been used to demonstrate the applicability of Linear Logic to capture the concept of resources in the real world, Linear and other substructural logics have not yet been successfully used to model temporal properties, which are necessary for verification of computing systems (software and hardware). Traditionally, such systems are specified using variants of temporal logics, which capture many of the notions of behaviour over a possibly infinite computation; however, what they are not ideal for is to describe and reason about *each* step of the computation.

Linear logic can be seen to model single computation steps directly, as it is often claimed that Linear Logic captures the notion of state, and a single step in a computation is a change of state. If a computation step *consumes* certain properties of state and generates new properties, it can be represented by a linear formula

$$P_1(s) \multimap P_2(s)$$

where s is the representation of state and P_1 and P_2 are predicates characterizing each state. Programs consist of *actions*, which can either be single commands represented as above or structuring constructs. We can represent a choice between two actions (a generalized version of if-then-else) by the linear par ($\parr$).

Loops play an important role in computations. The simplest model of a loop is an action that can be repeated (potentially infinitely) many times until some condition is met. Using linear constructs, this can be written simply with an exponential around an action; of course the condition needs to be coded in.

Let us take as an example an algorithm which calculates the GCD (Greatest Common Denominator) of two numbers by the difference method. This repeatedly replaces the larger of two values by the difference between it and the smaller value. The repetition continues until one of the values becomes zero, at which point the other value is the greatest common denominator of the two initial numbers. Here is a linear logic formula representing it:

$$x = a \otimes y = b,$$
$$!\forall v1, v2.((x = v1) \otimes (y = v2) \otimes (v1 < v2) \otimes (v1 > 0)) \multimap ((x = v1) \otimes (y = v2 - v1))$$
$$\parr((x = v1) \otimes (y = v2) \otimes (v1 \geq v2) \otimes (v2 > 0)) \multimap ((x = v1 - v2) \otimes (y = v2))$$
$$\vdash (x = 0 \otimes y = GCD(a, b))|(x = GCD(a, b) \otimes y = 0)$$

Let us see what this means: the initial state for the computation is specified by assigning the values a and b to x and y, respectively. The main body of the program is a loop, in which there is a choice between two actions, depending on whether the value stored in x is less than the value stored in y or not. The loop is only enabled if both values are greater than 0. Finally, the desired final state of the computation is reached when either x or y become 0, in which case the other variable contains the calculated GCD.

The above is a simplistic use of linear logic operators for describing algorithms. There are still many issues to be addressed - such as how to represent concurrency, or data scoping, or non-determinism, or which variant of linear/substructural logic is ideal for this kind of use, from the semantical point of view. It is hoped that this presentation will fuel further discussion and result in collaboration between logic developers and the applications community keen on exploring new avenues to facilitate formal verification.

Linear Abstract Machines with the Single Pointer Property

Eike Ritter,
School of Computer Science, University of Birmingham

Resource control in functional programming was an early application of linear logic. The idea was that as linear variables are used exactly once, objects of a linear type should have exactly one pointer to them. Hence update-in-place for linear objects should be possible. This turned out to be problematic, as was first observed by Gunter et al. I present in this talk a different approach. I describe a calculus which separates language constructs for resource control from language constructs for manipulating data. In this way we obtain a calculus which describes update-in-place. Linearity plays only a limited role in this calculus: if the all operations on a given type are destructive, it follows that variables of this type are linear. If a type admits also non-destructive operations (eg arrays) linearity is not a necessary condition for update-in-place. In general, with higher-order functions, linearity is also not a sufficient condition for update-in-place.

This is joint work with Valeria de Paiva.

In-place update with linear types or how to compile functional programs into malloc()-free C

Martin Hoffman,
University of Edinburgh

We show how linear typing can be used to write `C` functions which modify

their heap allocated arguments in a functional way without explicit "pointer surgery."

We present this both as a "design pattern" for writing `C`-code in a functional style and as a compilation process from linearly typed first-order functional programs into `malloc()`-free `C` code.

The main technical result is the correctness of this compilation.

We discuss various extensions, in particular an extension with FIFO queues admitting constant time catenation and enqueuing, and an extension of the type system to fully-fledged intuitionistic linear logic.

Spaces for the semantics of linear logic and linguistic representations

Francois Lamarche,
INRIA–Loria

This talk is the first announcement of a research program.

There are many areas of computer science that need a "theory of spaces whose points are little spaces themselves". For example in formal language theory (both as a discipline by itself and as an aid to linguistics), the individual words/sentences generated by a formal language are sets-with-structure. In this case the structure is very simple, being only a finite total ordering whose elements are marked by atomic symbols. But this structure, in addition to being variable from word to word (the vectors vary in length), is naturally endowed with a strong spatial character (words are definitely one-dimensional spaces). At the same time the set of all such words/sentences generated by a formal language has a strong intuitive, while hard to formalize, notion of "neighborhood" attached to it. Two sentences may be related by a simple substitution of words, or an active-passive transformation, which shows they are more closely related than two random samples. For another example replace the space of all sentences by the space of all their parsing trees. Here, the main difference is that the spatial character of the "little spaces" is not simply linear anymore, but tree-like.

So we have identified two levels of "spaceness", "big" and "small", the former serving as domain of variation, in the sense of Lawvere, for the latter. It turns out that once it is recognized, this situation appears in many areas of computer science and applied mathematics: concurrency theory (a process is a big space and its little spaces are its states), statistical learning theory, knowledge representation, rewriting theory, population biology...

We will present a general theory of such spaces. It is informed by two paradigms, that have to be adapted to fit together. One is the Grothendieck-

Lavwere theory of toposes, with its connection both to geometry and to model theory. The second one is linear logic: The operations that generate and split little spaces will be seen as generalized multiplicative connectors of linear logic, while the structure that unites all the little spaces into a big one proceeds from the additive fragment of linear logic. The natural "cement" between these two paradigms will be seen to be a class of theories in linear universal algebra, which can be seen as a "general theory of little spaces", and at the same time, as a theory of contexts for a multi-modal multiplicative fragment. A common slogan in proof theory is that "a logic is generated by its theory of contexts". The extension of a theory of contexts to a full logic can be be seen roughly as the addition of the negative connectors (par and with) to the purely positive fragment (tensor and plus). This adds the important property of *functionality* to the theory of spaces, which can be already observed in the simplest possible example, that of MALL.

Abstract Games for Linear Logic

Andrea Schalk,
Cambridge University

We draw attention to a number of constructions which lie behind many concrete models for linear logic; we develop an abstract context for these and describe their general theory. Using these constructions we give a model of classical linear logic based on an abstract notion of game. We derive this not from a category with built-in computational content but from the simple category of sets and relations. To demonstrate the computational content of the resulting model we make comparisons at each stage of the construction with a standard very simple notion of game. Our model provides motivation for a less familiar category of games (played on directed graphs) which is closely reflected by our notion of abstract game. We briefly indicate a number of variations on this theme and sketch how the abstract concept of game may be refined further.

Proof Search in Multiplicative Linear Logic

Bertram Fronhöfer,
Institut für Informatik, Technische Universität München

In [1] we gave a characterization of theoremhood in terms of the Connection Method for the multiplicative fragments of Affine (Contraction-Free) and Linear Logic. Such matrix characterisations constitute a basis for the adaption of the classical proof search procedures to the just mentioned logics.

Based on this work computationally advantageous variants of the previously developed matrix characterisations are established. Instead of the traditional complementarity of (horizontal) paths they hinge on the total connectedness of a sufficient number of virtual columns (vertical parths) in a matrix (non-purity). This yields a new view of the proof search process: Instead of having to examine all possible extension steps for a not yet complementary partial path, we just have to 'complete' not yet totally connected virtual columns. This means less possibilities and thus pruning away parts of the search space.

Thus we derive a pruning technique for proof search in acyclic (and linear) matrix graphs, which can be used for multiplicative Affine Logic. Extending proof search to multiplicative Linear Logic, we show that the additionally required minimality of spanning sets is automatically assured by depth-first search, and that the necessary connectedness of all literals can be tested incrementally.

References

[1] B. Fronhöfer: "The Action-as-Implication Paradigm: Formal Systems and Application", (revised version of Habilitationsschrift, TU München 1994), CSpress, München, Germany, Computer Science Monographs 1, 1996.

A Linear-time Algorithm for Verifying MLL Proof Nets via Lamarche's Essential Nets

A. S. Murawski and C.-H. L. Ong,
Oxford University Computing Laboratory

We consider the following decision problems:

PROOFNET : Given a multiplicative linear logic (MLL) proof structure, is it a proof net?

ESSNET : Given an essential net of an intuitionistic MLL formula, is it correct?

In this paper we give a linear-time algorithm for ESSNET. By showing that PROOFNET is linearly reducible to ESSNET, we obtain a linear-time algorithm for PROOFNET. Hence the question of whether a given linearly balanced MLL formula is provable can be settled by a linear-time algorithm.

On the Complexity of Proofs

Alessandra Carbone
University of Paris XII

We present a survey of results on graphs associated to formal proofs. From graph-theoretical properties we derive consequences that explain the expansion of proofs under cut-elimination, both for the propositional calculus and for the predicate calculus. Linear, polynomial, exponential and super-exponential complexities arise from the topology of the graphs.

Infinitary Linear Logic and Multiplicatively Quantified Linear Logic

Edward Hermann Haeusler and Luiz Carlos P.D. Pereira
Pontificial Catholic University of Rio de Janeiro

This work has the purpose of describing the multiplicative quantifiers that arise from infinitary versions of the tensor $\otimes$ and the par $\parr$ connectives.

Blass [1992] has made the observation that an infinitary version of the par connective would need a generalized form of sequent and its proof theory would be quite hard. In this talk we present:

- An infinitary sequent calculus for classical linear logic (with infinitary par and tensor) with semantics based on an extension of Phase-Spaces to cope with infinitary operations. The calculus is shown to be complete and sound with respect to this semantics.
- A Sequent Calculus for the Multiplicative Quantifiers that arise from the infinitary calculus. A notion of Linear Structure based on infinitary Phase Spaces is used to provide a sound and complete semantics for this calculus. The cut-elimination theorem is proved for this calculus.
- A multiple-conclusion Natural Deduction for infinitary classical linear logic. In order to obtain the normalization theorem we need to work with a restricted version of the proposed infinitary classical linear logic. This restrictin concerns the existence of finite upper bound for the degrees of formulas and for the height of the derivations.

As one conclusion from this work we can mention that the problems pointed out by Blass really concern *infinitary* classical linear logic rather than *multiplicatively quantified* classical linear logic.

Hybrid Linear Logics

Akim Demaille,
EPITA, France

Embedding problems within a logic is an attractive way of formalizing: the ground is known to be sound, there are usually many nice results etc. Unfortunately, in general, while the coding is sound (positive results for the initial problem are translated as theorems), it is far from being reciprocal (logic gives false positives).

A typical example of this phenomenon is the translation of lcc into linear logic by Paul Ruet : while from intuitionistic linear logic success are observable (Saraswat and Lincoln), failures are not. The reason is simply that

$$c \otimes (c \multimap 1) \vdash c \multimap (c \otimes 1)$$

is a theorem, while we would like it not to be. This is simply solved by introducing a second multiplicative family (i.e., tensor, par, implication(s) and the associated meta connective), so that

$$c \otimes (c \multimap\!\bullet 1) \nvdash c \multimap\!\bullet (c \otimes 1)$$

no longer holds.

The same problem arises in other works, while other domains need more structural policies (typically linguistics). Each time one realizes that one way out is to move to a system which has a bigger number of connectives, which are linked together via *domain specific* rules. There is therefore a need for a generic study of *hybrid* systems which allows to *design* a logical system suitable for a given problem.

Linear logic proves to be an excellent ground for such constructions. Guided by works by M. Abrusci, U. Reddy, Ph. De Groote and P. Ruet, the key point in order to superimpose several times linear logic (commutative or cyclic) in a single system is to design structured contexts, or *blocks*.

Then one may add any number of commutative and cyclic multiplicative families and still obtain a system with a natural phase semantics, soundness, completeness and cut elimination results in both classical and intuitionistic schemes.

These families may communicate via several *linkage rules*, such as *entropy* (which states that $A \odot B \vdash A \otimes B$), or *relax* $((A_1 \odot A_2) \otimes (B_1 \odot B_2) \vdash (A_1 \otimes B_1) \odot (A_2 \otimes B_2)$. Both orientations are valid, and suits different readings.

Let $\otimes$ be commutative and $\odot$ be non commutative. Then, with a process reading, it is natural to admit

$$A \otimes B \vdash A \odot B$$

since it reflects the fact that "one way to perform A and B is to do A and then B.

But with an *information* reading (or token, or resource), the converse is expected:

$$A \odot B \vdash A \otimes B$$

"if I have A and then B, then in particular I have A and B".

Intuitionistic systems prove to be much more robust to the addition of linkage rules. The only "risk" is to have equivalent families in case of circular dependencies. Classical systems do not support finer linkage rules than *entropy*. Any finer rule, such as *relax*, makes the system collapse to linear logic (all the tensors are the usual $\otimes$).

Modalities are also accommodated nicely in this framework.

Translations and Normalization: applications to linear logic

Luiz Carlos Pereira
Department of Philosophy, PUC-Rio/UFRJ

The aim of this paper is to give a brief presentation of some connections that can be established between translations and normalization proceedures. Special attention will be given to translations from classical linear logic into intuitionistic linear logic.

The double-negation family of translations (Gentzen, Gödel, Kolmogorov) can be easily justified, from the proof-theoretical point of view, by the normalization strategy used by Prawitz in [1]. We can also prove that Seldin's normalization procedure (see [2],[3]) gives rise to a new translation from classical logic into intuitionistic logic. When we consider classical linear logic, we can prove that:

- the $(\bot, -\circ)$-fragment satisfies Prawitz's strategy; and
- the $(\bot, \otimes)$-fragment satisfies Seldin's strategy.

These results allow us to define two natural translations from classical (propositional) linear logic into intutionistic (propositional) linear logic. In the final part of the paper we suggest how multiple conclusion systems for linear logic could be used in the definition of translations that are not based on the unity of the multiplicative disjunction.

References

[1] Prawitz, D. 1965. *Natural Deduction*, Almqvist & Wiksel, Stockholm.

[2] Seldin, J. 1989. *Normalization and Excluded Middle I* Studia Logica 48, pp.193-217.

[3] Seldin, J. 1986.*On the Proof Theory of the Intermediate Logic MH*, JSL 51, 1986, pp.626-647.

The Logic of Bunched Implications

David Pym,
Queen Mary and Westfield College, University of London

We present the logic of bunched implications, **BI**, in which a multiplicative (or linear) and an additive (or intuitionistic) implication live sided-by-side. Propositional **BI** can be seen to arise from an analysis of the proof-theoretic relationship between conjunction and implication, and may be viewed as a merging of intuitionistic logic and multiplicative, intuitionistic linear logic. The predicate version of **BI** includes, in addition to the usual additive quantifiers, multiplicative (or intensional) quantifiers, $\forall_{\mathbf{new}}$ and $\exists_{\mathbf{new}}$, which arise from observing restrictions on structural rules at the level of terms as well as propositions. Moreover, these restrictions naturally allow the distinction between additive and multiplicative predication for each propositional connective. We provide natural deduction and sequent calculi for **BI**. **BI** can also be seen to arise from an axiomatic theory of resources, which gives rise to a Kripke-style forcing semantics.

More generally, we give a BHK (categorical) semantics of proofs (together with a lambda-calculus). We mention computational interpretations of **BI**, based on locality and sharing, at both the propositional and predicate levels. We explain **BI**'s relationship with linear logic and other relevant logics.

On Resource Semantics of Bunched Implications

Peter O'Hearn
Queen Mary and Westfield College University of London

I describe a "spatial" reading of the logic **BI** of bunched implications, which derives from Reynolds's possible world semantics of imperative programming. In this view, **BI**'s implications and conjunctions get the following readings:

- $A \mathrel{-\!\!\circ} B$: if I have enough (current) resources to make A true, then then I can establish B as well.
- $A \mathrel{-\!\!*} B$: if you give me enough resources to make A true, then combining with the resourcs I have I can show that B is true.

- $A \& B$: A and B are true, of the current resources.
- $A * B$: the current resources can be decomposed int two parts, one of which makes A true and the other of which makes B true.

I consider a "money and chocolates" example to illustrate these readings, and also a model based on Petri nets. Example judgements are used to drive home the intuitive reading, and how it differs from the number-of-uses reading of linear logic.

- $A -\!\!* B \vdash A \to B$ is not derivable
- $A \vdash (A \to A \to B) \to B$ is derivable

The number-of-uses interpretation would answer the opposite for these judgements; for example, in the second the assumption A is used twice. We conclude that **BI** does not support, in an evident way, a uses interpretation. But the spatial reading provides a view of connectives which provides a way of understanding why these judgements are the way they are.

Linear Logic, SCI and Resources

Uday S. Reddy
The University of Illinois at Urbana-Champaign

We trace the evolution of two systems: Syntactic Control of Interference (SCI), due to Reynolds, and Intuitionistic Linear Logic (ILL), due to Girard. SCI has a a clear resource interpretation, where functions take arguments with which they share no resources. ILL, on the other hand, has a clear process interpretation where tensor means independent processes, and linear function space means processes that involve communication between the argument and the result. The BCI fragment of ILL does admit an emergent interpretation in terms of resources, but this interpretation does not seem to extend to "!". The author's Linear Logic Model of State (LLMS) is an extension of ILL with additional connectives to represent processes evolving in time, and servers to model imperative programs with SCI type regimen. Such processes inhabit structures called "object spaces," which form a symmetric monoidal closed category. This category can be Yoneda-embedded into a functor category (Kripke model). This gives rise to a model of Bunched Implications (BI) with two closed structures: one, symmetric monoidal, to model SCI functions, and the other, cartesian, to model general imperative programming functions. All said and done, this evolution represents an excellent study in the interplay of resource and process interpretations of substructural logics.