EPSRC Grant GR/L54639/01

Logic Programming, Imperative Progamming and Categorical Semantics

Final Report

P.W. O'Hearn D.J. Pym E.P. Robinson

October 25, 2000

1 Introduction

The goal of this research was to probe the semantic structures needed for imperative and logic programming, taking careful account of abstraction and control.

In imperative programing, a crucial problem area has always been *sharing*. Because of aliasing, and behind-the-scenes dependencies caused by state change, specification and reasoning techniques for realistic languages are excessively complex. The reasons for this complexity run deep, and are not merely down to idiosyncharicies in language designs. But models of imperative computation can have elegant properties, as was demonstrated in work on possible world semantics and, particularly, interference (e.g., [5, 4]).

In logic programming, a crucial problem has always been to control the dependencies between branches of a proof encountered in proof-search. This can also be seen as a sharing problem, and is essential to understand in order to give a comprehensive treatment of logic programming languages with substructural features or modules. Also, an account of sharing on the logic programming level could give rise to a principled approach to management of dependencies between different parts of a specification, a serious problem in specification formalisms.

Sharing is just one of the areas of commonality between logic programming and imperative programming. Others are control and abstraction. We first conceived this project with the view that these intuitive connections were suggesting deeper semantic links – between logic and logic programming, imperative programming, and categorical semantics – than is commonly supposed. We believed that cohesive theoretical results, and novel applications, could be obtained by pursuing these connections.

We would suggest that the initial, broad conception of our project has been well borne out by the results we have obtained, as we describe below. Although not all of our results span all of the areas involved, a certain cohesiveness can be seen, for example, in the work on control, with its basis in categorical semantics, and the related use of continuations in work on the proof theory of classical logic. And it can be seen even more strongly in our work on bunched logic, which might be considered as a meeting of the logical perspective of Pym and the imperative viewpoint of O'Hearn, with categorical semantics as mediator; the amount of gelling of the varied constituents exceeded any of our initial expectations.

2 Research Achievements

2.1 Bunched Logic

Our work in bunched logic [OP99, Pym99, O'H99] represents the strongest outcome of this project. The logic **BI** of bunched implications was introduced by O'Hearn and Pym in a short article in the Bulletin of Symbolic Logic [OP99]. A systematic presentation, including its semantics and model theory, proof theory, type theory and computational interpretation, is presented in a forthcoming monograph by Pym [Pym00b] (which will appear either in the Studies in Logic and Computation series, published by RSP/Wiley, or in Kluwer's Applied Logic series).

BI is a logic that combines intuitionistic logic (with the usual connectives $\wedge$, $\rightarrow$, $\vee$, $\perp$) and a basic substructural logic (with implication $-\!\!*$ and conjunction $*$). Although other logics (particularly linear logic) also combine the two fragments, the way that **BI** mixes them together is novel, as illustrated by the core theoretical perspectives on it.

- *Algebraic and Categorical Models.* A categorical model of **BI** is a *doubly-closed category*, which is a (bi)cartesian closed category together with an additional symmetric monoidal closed structure. A collapsed version of this structure is a **BI**-algebra, a Heyting algebra with an additional residuated commutative monoid structure. These models make the similarity and difference with linear logic evident. **BI** uses one category (or algebra) with two closed structures, where linear logic uses two separate categories or algebras (one of which is often a Kleisli category).

- *Bunched Proof Theory.* **BI**'s proof theory is formulated using *bunches*, which are nested, tree-like contexts built using two forms of combination. One form, ";", internalizes the cartesian product (additive conjunction) in a doubly closed category, while the other, ",", internalizes the monoidal product (substructural conjunction). The introduction rules for the two implications mimic the two adjunctions in a categorical model.

$$-\!\!*\, I \quad \frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi -\!\!* \psi} \qquad \to I \quad \frac{\Gamma; \phi \vdash \psi}{\Gamma \vdash \phi \to \psi},$$

- *Resource Semantics and Topological Models.* This is a truth semantics, which gives a declarative way of reading **BI**'s formulae. The basic idea is that the truth of a formula ϕ is judged "locally", in the sense that it talks about a circumscribed collection of resources. This is formulated in a possible worlds style, where the world m in a forcing judgement $m \models \phi$ is chosen from an ordered commutative monoid of worlds. In example models the resources correspond to areas of computer memory, markings in a Petri net, or processes. The most sophisticated version of the semantics uses topological ideas (Grothendieck topologies) to treat the intuitionistic connectives, with continuity conditions relating to the monoid structure on worlds.

- *The Sharing Interpretation.* This is a way of reading our various semantics, in terms of sharing between parts of a formula or parts of a proof/λ-term. For instance, when viewed as a function type, $A -\!\!* B$ is the type of functions that don't share resources with their arguments.

Although the basic system is intuitionistic, we have also investigated classical variations; for example, an algebraic model of Boolean **BI** is a **BI**-algebra in which the Heyting component is in fact Boolean.

These varied perspectives on **BI** reinforce one another, and lend support to our contention that it is an elegant and naturally occurring logic. Our technical investigations been driven by them.

- *Basic Proof Theory.* The system has a natural deduction and sequent calculus formulations, with normalization and cut elimination properties [Pym00b].

- *Type-theoretic Interpretation.* A λ-calculus of proof terms, syntactic results such as subject reduction and normalization, and soundness and completeness results for interpretation in doubly-closed categories. [O'H99, Pym00b].

- *Syntactic Control.* The type system is used to merge syntactic control of interference and Idealized Algol [O'H99, O'H00]. This is the best example of the sharing interpretation of proofs/λ-terms, and was in fact one of the early inspirations for **BI**. The resulting system resolves problems with recursion and jumps in the original syntactic control [9], which had befuddled experts for years.

- *Resource Semantics.* Soundness and completeness for a version of the semantics based on Grothendieck topologies. Demonstration of a range of contrete models that illustrate the informal reading, including models based on CCS, Petri nets, and pointers [OPY, Pym00b].

We have also explored **BI**'s place in the spectrum of substructural logics [8], contributing, we believe to the general understanding of these systems. **BI** is distinct both from the R-like relevant logics, in its semantics and treatment of implication, and from linear logic-like systems, in its treatment of the relationship between intuitionistic and linear implications and distributivity.

These results establish propositional **BI** as a theoretically cohesive system, with a significant early application. In addition, in work at QMW supported by another EPSRC grant ("Verified Bytecode"), and in work further afield, the pointer model of **BI** has been utilized as part of an approach to longstanding problems in Hoare logic for pointer programs

[3, 10, 11]. Research in this area, and on applications suggested by the other concrete models, is progressing rapidly.

To sum up our work on bunched logic, following its initial discovery based on proof-theoretic and category-theoretic ideas, the resource interpretations we have been developing have begun to take on more significance; the interpretation in terms of sharing, in particular, fits both imperative and logic programming extremely well. This direction shows good promise as far as further applications go, both those building one existing ones (on pointers and logic programming) and in new directions (such as concurrency and databases).

2.2 Predication and Type Dependency

Many substructural logics have been proposed where structural rules are limited, but where also the treatment of variables in term languages admits all structural rules. When substructural logics are used as type systems they give rise to substructural term languages, and it would seem natural to want to have an account of predication that takes such terms into account. But, although it seems natural, there are many conceptual and technical difficulties that must be overcome to obtain a workable approach; this is why *multiplicative quantification* or predication has remained a challenging open problem.

The predicate verion of **BI** investigated by Pym [Pym99, Pym00b] has two universal and existential quantifiers, just as propositional **BI** has two conjunctions and implications. There are additive quantifiers, whose semantics is given just as in intuitionistic logic. There are also multiplicative quantifiers, $\forall_{\mathbf{new}}$ and $\exists_{\mathbf{new}}$. In terms of resource semantics, $\forall_{\mathbf{new}} x.\, \phi$ says that ϕ is true for all individuals that access separate resources from the current resource. Good example models are obtained from imperative programming, where $\forall_{\mathbf{new}} x$ quantifies over values that refer to new or fresh storage. Indeed, the nomenclature for the quantifiers was partially inspired to their relationship to new storage variables in Idealized Algol.

The syntax and proof theory of multiplicative quantification uses trees or *bunches of variables*, just as propositional **BI** uses bunches on the propositional level. This gives a link into another, earlier, piece of work, on the substructural logical framework RLF. RLF uses an ordered form of bunched context to formulate its rules.

In the course of the project we obtained the following results.

- The $\lambda\Lambda$-calculus, the type theory underlying RLF, has been formulated. This resolves a difficult open problem concerning the combination of linearity and type dependency. Basic technical results such as normalization and confluence have been established [IP98, 2].
- We showed how the multiplicative function type could be used to represent substructural object logics, with a "uniform encoding" that establishes tight connections between proofs in the object logic and the metalogic. This was a problem area in the original LF.
- Logical frameworks for operational semantics [2, IP98]: an adequate representation of languages such as ML with reference types including, for the first time in logical frameworks, a treatment with close control over garbage cells.
- A class of concrete, possible worlds models of the dependent linear function type [IP99, IP].
- A categorical semantics, model theory, and proof theory of multiplicative quantification in predicate **BI** [Pym99, Pym00b].
- Proof-search (logic programming) [OP99, Pym99]: the proof theory of predicate **BI** provides a basis for a logic programming language within which sharing and non-sharing of data by procedures can be modelled directly. This work is continuing, in particular with a study of applications to modules, really exploiting multiplicative predication and quantifiers, in the the work of Pym's PhD student, Pablo Armelin.

These results are but a start. From the theoretical point of view multiplicative quantification and type dependency are delicate and raise many problems. A more substantial and general theory is called for, and is under development (taking cue from these initial results). As for applications, this direction appears to be particularly relevant in situations where substructural features are on view in a type system or object logic, and for this reason substructural frameworks are beginning to attract attention from researchers on proof-carrying code.

2.3 Classical Logic, Proof-search and Categorical Semantics

Pym, in joint work with Eike Ritter, has pursued a substantial project in the semantics and proof theory of classical logic. Motivated by a desire to understand the operational and denotational semantics of models of computation based on proof-search, *e.g.*, logic programming.

Our work here falls into three main chunks, each of which relies on a type-theoretic analysis of the key classical connective, i.e. disjunction:

- Basic proof theory [PW00a]: Here we develop a proof-theoretic analysis of the embedding of the intuitionistic sequent calculus within the classical sequent calculus. We exploit type-theoretic methods to show how to recover intuitionistic soundness from classical searches. We apply our analysis to uniform proofs, a candidate theoretical foundation for logic programming;

- Application to resolution [PW00b]: We apply our methods to obtain a rational reconstruction of intuitionistic resolution from classical resolution. Again, our techniques are primarily type-theoretic. This work is related to Pym's work with Harland on resource-distribution in linear logic, in which linearly sound proofs are recovered from classical searches via systems of Boolean constraint equations [1]. Both of these works are contributions to our general apraoch to a theory of search [GP00, Pym00a];

- Categorical model theory [PR]: We provide a semantic analysis of classical disjunction, revealing how the familar collapse of a categorical semantics of classical proofs is sensitive to the formulation of disjunction.

Of particular significance is our development of a semantics of classical disjunctive proofs using a category of *continuations*.

2.4 Control and continuations

Our most substantial work on control was led primarily by our RA, Hayo Thielecke, who arrived in July 1997, having completed a thesis on the categorical structure of continuations. While highly original, this thesis also gave an extremely abstract picture of control. We, and he, were interested to see if its ideas could be leveraged to make more concrete insights about control.

This began with a paper on the expressiveness of the most powerful form of control, the call/cc construct. Thielecke showed that the ability to use a continuation multiple times, by backtracking, led to great expressive power, in terms of distinguishing ability. This work came about as a result of an exploration of technical properties of the categorical models, specifically properties of the *centre* of a *premonoidal category*. But the results could be phrased just in terms of programs and operational semantics.

Thielecke's study of the relationship between the centre and notions of "effect free" morphism has played a leading role in other work. One paper [PT99], by him and John Power, gives a categorical account of higher-order structure in call by value, building on the Power-Robinson notion of premonoidal category. Other work, by Carsten Fuhrmann at Edinburgh, investigates some of the properties identified by Thielecke for effects other than continuations.

Thielecke's next step was to begin an analysis of the behavioural, jumping properties of exceptions. exceptions. This was done first by comparing their expressiveness to that of call/cc. In a paper with Jon Riecke from Bell Labs, Thielecke showed that pure exceptions and call/cc (i.e., without state) are *incomparable* in expressive power: each can break program equivalences that the other can not. This drives a very clear wedge between these two prominent forms of control, one that is fundamental in that it involves program equivalence only, and does not rely on implementation strategies (stacks v heaps), important though these are.

In a further paper the impact of state on the expressive power of continuations and exceptions is analyzed [Thi00], and a final paper makes a furher comparison by exploring their typing poperties and making connections to logic [Thi].

All told, Thielecke's work is remarkable for two reasons. First, it links very abstract categorical notions to concrete programming language questions, questions that can be posed without mentioning the abstract techniques used to help resolve them. Second, he has provided the first fundamental study of behavioural properties of exceptions. In ways, the exception work is modest theoretically, but that is part of the point given the dearth of work giving formal anayses of exceptions. This begins to correct an imbalance in the theory of control. While call/cc is extremely powerful only one language, Scheme, has it as part of its standard. But while there have been dozens of papers on continuations, there has only been a handful of papers analyzing the much more widely used exceptions.

2.5 Abstract structures, logical relations and realizability

Our work on **BI** helps address notions of shared and private state. However we have also made progress on reasoning about abstraction properties. The major work here was done by Robinson in collaboration with John Power [PR00a, PR00b], but building on earlier pieces of work by Robinson alone and O'Hearn. The achievement here was to make precise a standard notion of (observational) equivalence for abstract structures, and to prove that a certain generalised form of logical relation was sound and complete for proving equivalence. In order to allow for higher types one has to get beyond the simplest set-based semantics, and this makes the simple form of logical relations inappropriate. What Robinson did was take a form of relation introduced by Jung and Tiuryn to study definability, and called by them "Kripke logical relations of varying arity", show how these could be regarded as unary relations, and then adapt a binary variant to prove completeness. In [PR00a] the original presentation and proof is cleaned up considerably, and in [PR00b] the result is extended from cartesian closed to monoidal closed categories, thus extending the class of languages to which it applies.

Two pieces of further work present themselves. The first is to attempt to extend the result to closed premonoidal categories, increasing the language class to include many more forms of control. The second is to relate this work to alternative approaches which use a relaxation of the logical relations condition. Some material on this had to be cut from the accepted version of [PR00b] for reasons of space. However Power and Robinson have established a link in which the relaxed version is a kind of global section of theirs.

The second thread was again in conjunction with Power [PR99]. This work is a generalisation of Moggi's notion of computational monads to allow data to be kept in the context. The point of this can be seen by considering the notion of state. In Moggi's original treatment the State monad is

$$X \mapsto [S \to [S \times X]]$$

so that a term of type B with a variable from A is represented by a morphism

$$A \longrightarrow [S \to [S \times X]]$$

A more natural and equivalent representation is

$$S \times A \longrightarrow S \times B$$

which would allow a more optimal treatment of the state, and this is what is allowed by Power and Robinson's generalisation. Specifically, this builds on previous work by Power and Rosolini, establishing one possible generalisation. Robinson observed that this could be made symmetric between input and output, allowing both a more natural mathematical structure, and potentially a wider range of examples. The work provides a mathematical treatment up to the point where it would be relatively simple to write a Haskell library implementing dyads, converting computational monads to dyads, and allowing composition of dyads. This is essentially the functionality of the standard library for computational monads.

A particularly tantalyzing account of abstraction is provided by realizability semantics. This provides a way of tying an extensional structure to an intensional one, so that one gets structures with extensional behaviour, but in which every operation is denotable by a term. This intuition is made much more precise and general than previously in some work in progress [MRR]. This work is still unfinished, but is concerned with the categorical construction of realizability models, and a a draft paper containing precise results should be ready for submission within the next month.

3 Research Impact

Most of the work presented here has had some impact inside the Semantics and Theoretical Computer Science communities. However, some of it is starting to have an impact outside of these communities.

Our work on **BI** is still very new, and challenges some long-held assumptions. Some leading figures, such as Girard, have been very supportive, while others have yet to see its advantages. **BI** has been used in recent advances on program logic for pointers, a longstanding open problem [3]. The basic idea is to use **BI**'s resource semantics to make local statements about areas of memory. We have submitted a grant proposal based on these ideas, but they have also been taken up elsewhere in the community, for example by John Reynolds at Carnegie-Mellon [10], and by Uday Reddy's group in Birmingham [11].

Thielecke results have been recognised well outside the conventional semantics community, particularly by the compiler design research groups at Carnegie-Mellon and Cornell. This is particularly because of his use of concrete runnable examples to illustrate his semantic distinctions.

Our work on classical logic has solved longstanding problems in the theory of proof-search, as well as in the semantics of classical proofs, and has generated problems contributing to two new MATHFIT grant proposals.

4 Training and Personal Outcomes

We employed two RA's on the grant. During the period he worked for us Hayo Thielecke cemented his position as the UK's leading expert on continuations, and extended his reputation internationally as one of the leaders in an admittedly rather specialised, but still significant, field. His intention on joining us had been to find a permanent position in a strong research group, and while with us he was offered and accepted a permanent lectureship at Birmingham, where he is now.

The second RA, Cristiano Calcagno is a research student whose supervision is being shared by Moggi at Genova and O'Hearn at Queen Mary. Calcagno has started to publish, and has a single-author paper accepted at the prestigious Principles of Programming Languages conference (POPL01). This is a significant achievement for a student. Calcagno's work is part of a stream which has led to a new research grant application by O'Hearn and Pym, bringing in Bornat.

We also used the grant to give some travel support to an EPSRC student Paul Levy, see publications [Lev99]. Levy is now in Boston, having been headhunted as an RA by Harry Mairson. Samin Ishtiaq and Pablo Armelin, both EPSRC-funded students, have been closely involved with the project. Ishtiaq has nowgraduated and has taken up a post in formal methods with the Cambridge-based chip-design company, ARM. Armelin's thesis is now nearing completion.

The investigators have also advanced their careers thanks in part to this grant. Both O'Hearn and Pym have been promoted to full professor, with the research conducted on the grant putting the final seal on their cases.

Finally, departmentally, this grant has been significant. It has been a major source of funding — leading to several current grant applications by O'Hearn and Pym, by Hyland, Pym and Robinson, and by Pym and Ritter — during the development of the Logic and Foundations of Programming research group.

5 Explanation of Expenditure

Staff: The major part of our expenditure was on our designated RA, Hayo Thielecke. Hayo left a little early, in order to fit in a lengthy visit to Tennent in Canada before taking up his post at Birmingham (this visit was funded by Tennent). This left us with a small amount spare, which enabled us to fund Calcagno for a period. Calcagno's funding should be viewed more as training than advanced research, and although his time on the project did not result in any direct publications, the training he has received has enabled him to produce work of the kind mentioned above.

Travel: The next largest item is travel. We have travelled extensively during the period, and presented a good deal of work at conferences, as can be seen from the attached project bibliography. Presentation of most of the conference papers there was supported out of this grant.

Equipment: There were two major areas of equipment expenditure. The first was the purchase of a number of personal computers at the start of the grant. This helped significantly to kit out the burgeoning research group. One of them is being used to prepare this report. The other area was contributions to the department's normal rolling program of equipment replacement. Notable expenditure here included an early contribution to a workgroup printer, and a late contribution when the department replaced its file server.

Consumables: A large item is a fixed charge levied by the department covering stationery, printing and photocopying costs, postage, phone bills and the like. The remainder is the usual small pieces of equipment and software.

6 Further Research

We have mentioned earlier the continuing nature of much of this work. In particular, we should single out again the continuing work on pointer logic and **BI**, for which we have submitted a new proposal to EPSRC. The present work has laid the foundations for an approach to proof-search, for which support is again being sought.

7 Web site and documentation

This report, together with links to participants pages and papers at
`http://www.dcs.qmw.ac.uk/~edmundr/L56439/`

The bibliography for this report is split into two sections. The list of supported publications contains those works which have been at least partially supported by this grant.

Supported Publications

[GP00] D. Galmiche and D. Pym. Proof-search in type-theoretic languages: an introduction. *Theoretical Computer Science*, 232:5–53, 2000.

[IP] S.S. Ishtiaq and D.J. Pym. Kripke resource models of a dependently-typed, bunched λ-calculus. Submitted: manuscript available at http://www.dcs.qmw.ac.uk/∼pym.

[IP98] S.S. Ishtiaq and D.J. Pym. A relevant analysis of natural deduction. *Journal of Logic and Computation*, 8(6):809–838, 1998.

[IP99] S.S. Ishtiaq and D.J. Pym. Kripke resource models of a dependently-typed, bunched λ-calculus (extended abstract). In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, volume 1683 of *LNCS*, pages 235–249. Springer, 1999.

[IP00] S. Ishtiaq and D. Pym. Corrections and remarks. Research Report RR-00-04. ISSN 1470-5559., Department of Computer Science, Queen Mary and Westfield College, University of London, London, 2000.

[Lev99] P.B Levy. Call by push-value: a subsuming paradigm. In J-Y. Girard, editor, *Typed λ-calculus and Applications*, volume 1581 of *Lecture Notes in Computer Science*. Springer, 1999.

[MRR] F. De Marchi, E.P. Robinson, and G. Rosolini. An abstract look at realizability. in preparation.

[O'H98] P.W. O'Hearn. Polymorphism, objects and abstract types. *SIGACT News*, 29(4):39–40, December 1998.

[O'H99] P.W. O'Hearn. Resource interpretations, bunched implications and the $\alpha\lambda$-calculus. In J-Y. Girard, editor, *Typed λ-calculus and Applications*, volume 1581 of *Lecture Notes in Computer Science*. Springer, 1999.

[O'H00] P.W. O'Hearn. On bunched typing. Submitted to Journal of Functional Programming, 2000.

[OP99] P.W. O'Hearn and D.J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, June 1999.

[OPY] P. O'Hearn, D. Pym, and H. Yang. Possible worlds and resources: the semantics of BI. Submitted to Theoretical Computer Science.

[PR] D. Pym and E. Ritter. On the semantics of classical disjunction. To appear: *Journal of Pure and Applied Algebra*.

[PR99] John Power and Edmund Robinson. Modularity and dyads. *Electronic Notes in Theoretical Computer Science*, 20:14pp, 1999. http://www.elsevier.nl/.

[PR00a] A.J. Power and E.P. Robinson. Logical relations and data abstraction. In Peter Clote and Helmut Schwichtenberg, editors, *Proceedings of Computer Science Logic 2000*, Lecture Notes in Computer Science. Springer Verlag, 2000.

[PR00b] A.J. Power and E.P. Robinson. Logical relations, data abstraction and structured fibrations. In Maurizio Gabrielli and Frank Pfenning, editors, *Proceedings of the Second International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP'00)*, pages 15–23. ACM Press, 2000.

[PT99] John Power and Hayo Thielecke. Closed Freyd- and kappa-categories. In Jǐrí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Proceedings ICALP '99*, number 1644 in LNCS, pages 625–634. Springer Verlag, 1999.

[PW00a] E. Ritter D.J. Pym and L.A. Wallen. On the intuitionistic force of classical search. *Theoretical Computer Science*, 232:299–333, 2000.

[PW00b] E. Ritter D.J. Pym and L.A. Wallen. Proof-terms for classical and intuitionistic resolution. *J. Logic Computat.*, 10(2):173–207, 2000.

[Pyma] D.J. Pym. The semantics and proof theory of the logic of bunched implications, I: Propositional **BI**. 1998. Available on the web at http://www.dcs.qmw.ac.uk/ ˜ pym, superceded by [Pym00b].

[Pymb] D.J. Pym. The semantics and proof theory of the logic of bunched implications, II: Predicate **BI**. 1998. Superceded by [Pym00b].

[Pym98] D.J. Pym. Logic programming with bunched implications (extended abstract). *Electronic Notes in Theoretical Computer Science*, 17, 1998. 24 pages.

[Pym99] D.J. Pym. On bunched predicate logic. In *Proc. LICS'99*, pages 183–192. IEEE Computer Society Press, 1999, 1999.

[Pym00a] D. Pym. Notes towards a semantics for proof-search. Manuscript: available at http://www.dcs.qmw.ac.uk/ pym, 2000.

[Pym00b] D.J Pym. *The semantics and proof theory of the logic of bunched implications*. 2000. forthcoming monograph.

[RT99] Jon G. Riecke and Hayo Thielecke. Typed exceptions and continuations cannot macro-express each other. In Jǐrí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Procedings ICALP '99*, volume 1644 of *LNCS*, pages 635–644. Springer Verlag, 1999.

[Thi] Hayo Thielecke. Comparing control constructs by typing double-barrelled CPS transforms. Submitted to the 2001 Continuations Workshop.

[Thi98] Hayo Thielecke. An introduction to Landin's "A generalization of jumps and labels". *Higher-Order and Symbolic Computation*, 11(2):117–124, 1998.

[Thi99a] Hayo Thielecke. Continuations, functions and jumps. *SIGACT News*, 30(2):33–42, June 1999.

[Thi99b] Hayo Thielecke. Using a continuation twice and its implications for the expressive power of `call/cc`. *Higher-Order and Symbolic Computation*, 12(1):47–74, 1999.

[Thi00] Hayo Thielecke. On exceptions versus continuations in the presence of state. In Gert Smolka, editor, *Programming Languages and Systems, 9th European Symposium on Programming, ESOP 2000,*, number 1782 in LNCS, pages 397–411. Springer Verlag, 2000.

Other References

[1] J. Harland and D. Pym. Resource-distribution via boolean constraints. In *Proc. CADE-14*, number 1249 in LNAI, pages 222–236. Springer, 1997.

[2] S.S. Ishtiaq. A relevant analysis of natural deduction. PhD thesis, Queen Mary and Westfield College, University of London, 1999.

[3] S.S. Ishtiaq and P. O'Hearn. **BI** as an assertion language for mutable data structures. To appear: Proc. POPL '01, 2001.

[4] P. W. O'Hearn, A. J. Power, M. Takeyama, and R. D. Tennent. Syntactic control of interference revisited. *Theoretical Computer Science*, 228(1-2):211–252, October 1999. Preliminary version appeared in the proceedings of MFPS'95.

[5] P. W. O'Hearn and R. D. Tennent. Parametricity and local variables. *J. ACM*, 42(3):658–709, May 1995.

[6] John Power and Hayo Thielecke. Environments, continuation semantics and indexed categories. In Martín Abadi and Takayasu Ito, editors, *Proceedings TACS'97*, number 1281 in LNCS, pages 391–414. Springer Verlag, 1997.

[7] D.J. Pym. A note on representation and semantics in logical frameworks. In *Proc. CADE-13 Workshop, Proof-search in type-theoretic languages*, 1996.

[8] G. Restall. *Substructural Logics*. Routledge, 2000.

[9] J.C. Reynolds. Syntactic control of interference. In *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages*, pages 39–46, Tucson, Arizona, January 1978.

[10] J.C. Reynolds. Lectures on reasoning about shared mutable data structure. IFIP WG, Tandil, Argentina, September 2000.

[11] H. Yang. An example of local reasoning in **BI** pointer logic: the Schorr-Waite graph marking algorithm. Manuscript, Univ of Birmingham, October 2001.