North American Jumelage "Type Systems"
Grant Number N00014-93-1-0102
Final Report

Andre Scedrov

October 20, 1993

Best
Available
Copy

1 Productivity measures

Principal Investigator's Name:	Andre Scedrov
PI Institution:	University of Pennsylvania
PI Phone Number:	(215) 898-5983, -8178
PI E-mail Address:	andre@cis.upenn.edu
Grant or Contract Title:	North American Jumelage "Type Systems"
Grant or Contract Number:	N00014-93-1-0102
Reporting Period:	15 Oct 92 - 14 Oct 93

Refereed papers submitted but not yet published:	0
Refereed papers published:	0
Unrefereed reports and articles:	0
Books or parts thereof submitted but not yet published:	1
Books or parts thereof published:	0
Patents filed but not yet granted:	0
Patents granted:	0
Invited presentations:	0
Contributed presentations:	0
Honors received (fellowships, technical society appointments, conference committee role, editorship, etc.):	0
Prizes or awards received (Nobel, Japan, Turing, etc.):	0
Promotions obtained:	0
Graduate students supported >= 1/4 of full time:	0

Post-docs supported >= 1/4 of full time: 0

Minorities supported (include Blacks, Hispanics, American Indians and other native Americans such as Aleuts, Pacific Islanders, etc., Asians, and Indians): 0

2 Detailed summary of technical progress

Principal Investigator's Name:	**Andre Scedrov**
PI Institution:	**University of Pennsylvania**
PI Phone Number:	**(215) 898-5983, -8178**
PI E-mail Address:	**andre@cis.upenn.edu**
Grant or Contract Title:	**North American Jumelage "Type Systems"**
Grant or Contract Number:	**N00014-93-1-0102**
Reporting Period:	**15 Oct 92 - 14 Oct 93**

North American Jumelage is a working group on "Type Systems", which meets once a year, usually in the fall. The meetings are conducted in an informal setting, in a focused "working group" atmosphere. The schedule involves a combination of a few invited one hour talks, with ample time for informal discussions and presentations of work-in-progress, similarly to the IFIP-style working groups. The focus of the working group is primarily on the theoretical aspects of type systems stemming from typed lambda calculi, but we also cover wider topics ranging from mathematical semantics to the software design issues concerning typed programming languages. Previous meetings were held at Stanford in 1990, hosted by John Mitchell, and at AT&T Bell Labs in 1991, hosted by Dave MacQueen. The North American working group was originally inspired by European Jumelage in Typed Lambda Calculus, led by Jean-Yves Girard and including researchers from a number of E.C. countries.

The general coordinator of the working group is Andre Scedrov (Penn). Albert Meyer (MIT), Rick Statman (CMU), John Mitchell (Stanford), and Anil Nerode (Cornell) are "regional university representatives". Philip Scott (Ottawa) is the representative for the participation of Canadian universities. The group also has "industrial affiliates": Luca Cardelli (DEC SRC) and David MacQueen (AT&T Bell Labs).

The 1992 meeting, which was partially supported by the ONR grant N00014-93-1-0102, was hosted by Anil Nerode at the Mathematical Sciences Institute, Cornell University, Ithaca, New York, on October 15-17, 1992. The meeting included 50 participants. The program, list of participants, and the abstracts of most the talks are included in this report. There were no published proceedings. During the 1992 North American Jumelage meeting it was decided that because of the intensity of research in the area of Linear Logic, a one-time special follow-up workshop on that topic would be held at the same site in June 1993.

Consequently, a Linear Logic Workshop was held June 14-18, 1993 at the Mathematical Sciences Institute, Cornell University, Ithaca, New York. The workshop, also partially supported by the ONR grant N00014-93-1-0102, was attended by about 70 participants from the U.S., Canada, Europe, and Japan. The program committee was chaired by Andre Scedrov (Penn) and included S. Abramsky (Imperial, London), J.-Y. Girard (CNRS Marseille), D. Miller (Penn), and J. Mitchell (Stanford). The program and the list of participants are included below. Proceedings of the workshop, edited by J.-Y. Girard, will be published as a hardcover book.

NORTH AMERICAN JUMELAGE '92

1992 North American Jumelage meeting was hosted by Anil Nerode at the Mathematical Sciences Institute, Cornell University, Ithaca, New York, on Thursday-Saturday, October 15-17.

MEETING SITE: Mathematical Sciences Institute (MSI)
Room 214, 2nd Floor
409 College Avenue, Ithaca, New York 14850

PROGRAM

Wednesday, October 14, 1992

Informal get together 7-9 p.m. at MSI. Light refreshments served.

Thursday, October 15, 1992

9:00 - 9:10	Welcome
9:10 -10:10	Invited Lecture B. Bloom (Cornell)
10:10 -10:30	Break
10:30 - 1:00	Session on Proof Theory Chair: J. Gallier (Penn)
1:00 - 2:30	Lunch
2:30 - 3:40	Session on Formalizing Algebra in Type Theory I Chair: R. Constable (Cornell)
3:40 - 4:10	Break
4:10 - 5:20	Session on Formalizing Algebra in Type Theory II Chair: R. Constable (Cornell)
7:00 - 9:00	Reception at MSI. Hot and cold appetizers served.

Friday, October 16, 1992

9:00 -10:00	Invited Lecture J. Mitchell (Stanford)
10:00 -10:30	Break
10:30 - 1:00	Session on Programming Language Design Chairs: C. Gunter (Penn) and R. Harper (Carnegie Mellon)
1:00 - 2:30	Lunch
2:30 - 3:30	Invited Lecture D. Leivant (Indiana)
3:30 - 4:00	Break

4:00 - 6:00   Student and postdoc presentations

Saturday, October 17, 1992

9:30 -10:30	Invited Lecture J.-L. Krivine (Paris 7)
10:30 -11:00	Break
11:00 -12:00	Invited Lecture M. Felleisen (Rice)
12:00 - 1:30	Lunch
1:30 - 2:50	Session on Full Abstraction I Chair: A.R. Meyer (MIT)
2:50 - 3:20	Break
3:20 - 5:00	Session on Full Abstraction II Chair: A.R. Meyer (MIT)
5:00	MEETING ENDS

PARTICIPANTS

Guy Blelloch	Guy.Blelloch@BLELLOCH.PC.CS.CMU.EDU
Bard Bloom	bard@cs.cornell.edu
Val Breazu-Tannen	val@saul.cis.upenn.edu
Stephen Brookes	Stephen.Brookes@BROOKES.PC.CS.CMU.EDU
Paul Broome	broome@brl.mil
Kim Bruce	kim@cs.williams.edu
Jawahar Chirimar	chirimar@saul.cis.upenn.edu
Edmund Clarke	emc@cs.cmu.edu
Robert Constable	rc@cs.cornell.edu
Stavros Cosmadakis	stavros@watson.ibm.com
Pierre Cregut	cregut@research.att.com
Matthias Felleisen	matthias@cs.rice.edu
Amy Felty	felty@research.att.com
Stacy Finkelstein	stacy@saul.cis.upenn.edu
Peter Freyd	pjf@cis.upenn.edu
Jean Gallier	jean@saul.cis.upenn.edu
Philippa Gardner	pag@saul.cis.upenn.edu
Carl Gunter	gunter@saul.cis.upenn.edu
Elsa Gunter	elsa@research.att.com
Robert Harper	Robert_Harper@GOTTLOB.TIP.CS.CMU.EDU
Brian Howard	bhoward@saul.cis.upenn.edu
Doug Howe	howe@cs.cornell.edu
Paul Jackson	jackson@cs.cornell.edu
Radhakrishnan Jagadeesan	rj2@doc.imperial.ac.uk
Lalita Jategaonkar	lalita@theory.lcs.mit.edu
Dexter Kozen	kozen@cs.cornell.edu
Jean-Louis Krivine	krivine@logique.jussieu.fr
Daniel Leivant	leivant@moose.cs.indiana.edu
Arthur Lent	aflent@theory.lcs.mit.edu
Patrick Lincoln	lincoln@theory.stanford.edu
James Lipton	lipton@saul.cis.upenn.edu
Dave MacQueen	dbm@research.att.com
Albert R. Meyer	meyer@theory.lcs.mit.edu
John Mitchell	jcm@cs.stanford.edu
Philip Mulry	phil@cs.colgate.edu

Anil Nerode	anil@math.cornell.edu
Peter O'Hearn	ohearn@top.cis.syr.edu
Mitsuhiro Okada	okada@cs.concordia.ca, okada@lri.lri.FR
Frank Oles	OLES%YKTVMV.bitnet@CUNYVM.CUNY.EDU
Prakash Panangaden	prakash@trichur.cs.mcgill.ca
Richard Platek	platek@math.cornell.edu
Michael Rathjen	rathjen@function.mps.ohio-state.edu
Jon Riecke	riecke@research.att.com
Andre Scedrov	andre@cis.upenn.edu
Philip J. Scott	scpsg@acadvm1.uottawa.ca
Roberto Segala	segala@theory.lcs.mit.edu
Ramesh Subrahmanyam	ramesh@saul.cis.upenn.edu
Robert Tennent	rdt@dcs.ed.ac.uk
Ramesh Viswanathan	vramesh@theory.stanford.edu
Stanley Wainer	wainer+@andrew.cmu.edu

~~~~~~

ABSTRACTS OF TALKS

Towards a Metatheory of Structural Operational Semantics

Bard Bloom

The methods of classical denotational semantics provide a number of valuable tools to designers of sequential Algol-like programming languages. In particular, Algol-like languages can be described by recursive domain equations and semantic clauses. It is rather difficult to solve such equations from first principles (Dana Scott got a Turing award for solving the first one). Fortunately for language designers, the metatheory of denotational semantics shows that every set of recursive domain equations has a solution. Indeed, the metatheory is so powerful that programming language designers can use the theory naively and be guaranteed that no foundational problems will arise.

However, the theory that has been so successful for Algol-like languages is less appropriate for concurrency. The approach to language definition that seems most effective so far is structural operational semantics (SOS). To date, most uses of SOS for language definitions have been ad-hoc. We sketch the outlines of a metatheory of SOSses as used to define concurrent languages, and in particular the discipline of process algebras.

The first concern is whether or not a set of SOS rules define an operational semantics at all. As there are negative rules -- viz., rules which state that one process can act if another cannot -- it is not clear that there are any sound transition relations. Nonetheless, for the class of GSOS rules [BIM88], we show that there always is a unique and satisfactory operational semantics. Thus language designers can use GSOS rules naively and be guaranteed that their language makes basic sense.

Indeed, a good deal more holds. Two of the three main schools of process algebras are based on the notion of bisimulation; we show that any GSOS language respects bisimulation (technically, bisimulation semantics are compositional), and the very successful proof methods using bisimulation apply to them.

Furthermore, it is often possible to give a more accurate semantics. That is, bisimulation is an extremely fine semantics; it makes many distinctions between processes. Coarser semantics, when they are adequate, allow more powerful reasoning principles: e.g., there are compiler optimizations allowed by, say, failures semantics which
~~~~~~

violate bisimulation semantics. These optimizations can be applied only if failures semantics are adequate. We present a collection of theorems showing when most of the common coarser semantics are adequate. That is, a language designer may simply look at the rules for the language and tell that, say, the ready trace model is adequate and no coarser model is likely to be.

Finally, we give methods building logics for verifying programs. From a GSOS specification, we show how to derive a complete equational axiom system with one infinitary axiom (viz. an induction principle). The axioms our algorithm produces are comparable to those devised by researchers, and in a few cases actually superior.

On the Proof Theory of Kruskal's Theorem

Michael RATHJEN

Kruskal's theorem (for short, KT) asserts that the finite trees are well--quasi ordered under embeddability. This theorem is the main tool for showing that certain sets of rewrite rules are terminating.

The usual proof of KT utilizes an impredicative Π^1_1 comprehension. Friedman showed that KT is not provable in predicative systems in that he devised an order homomorphism from the set of finite trees onto a system of ordinal notations for the ordinal Γ_0. Friedman's construction can be carried out for stronger notation systems. In this talk I will present the strongest ordinal notation system for which this can be done. This leads to a calibration of the proof--theoretic strength of KT, thereby giving, in some sense, the most constructive proof of KT. \\
This is joint work with A. Weiermann.

Abstract : "Ordinal Complexity of Recursive Definitions"
Stan Wainer (Visiting CMU from Leeds UK).

The methods of Proof Theory and Subrecursive Hierarchies are used to measure and compare the complexities of various kinds of recursive definition (and their modes of evaluation), according to the sizes of their termination orderings. This is an old-established theme in Mathematical Logic (in fact a theorem of Tait 1961 reappears here in a generalized form), but newer results have emerged only recently.
The objective is to compute the ordinal trade-off \alpha to \beta to \gamma such that arbitrarily nested (call by value) recursive definitions over wellorderings \alpha can be
(1) transformed into while-programs (tail recursions) over wellorderings \beta,
and (2) evaluated by rewriting over termination orderings \gamma.
Each such trade-off corresponds clearly to a form of Cut-Elimination and we have
(1) \beta = exp(\alpha) corresponding to Gentzen Cut-Reduction, and
(2) \gamma = countable collapse of \alpha +, corresponding to 'complete cut-elimination' a la Girard.

A Typed Pattern Calculus

Val Breazu-Tannen, University of Pennsylvania

ABSTRACT

Programming with pattern-matching function definitions is a very attractive feature that accounts for much of the popularity of functional languages such as Hope, ML, Miranda, and Haskell. It is a pity therefore that our current understanding of such programs is largely operational, and that no more of their structure than that explained by first-order rewrite systems has been analyzed. This situation would be changed if we could understand pattern constructs as well as we now understand Algol-like and functional programming constructs. A crucial role in understanding these latter constructs has been played by the lambda calculus and its various type disciplines. We present a corresponding ''calculus'' that models programs with pattern-matching.

To see how this pattern calculus comes about, recall the propositions-as-types/programs-as-proofs analogy, an extremely fruitful idea that originated with Curry and Howard. They have shown that there exists an ''isomorphism'' between the terms of typed lambda calculus and the natural deduction proofs of intuitionistic logic. The constructor terms of functional programming correspond to those proofs built using the introduction rule of natural deduction. Now, patterns may look like constructor terms, but operationally they are dual to them. There is one formulation of logical proof systems in which this duality is made clear, and this is Gentzen's sequent proof system. Our calculus arises as a computational interpretation of these proofs. the sequent system has right rules, which are the same as the introduction rules of natural deduction, left rules, which we use to build patterns, and the cut rule, which is interpreted as a {\em let} construct and where computations originate . The left contraction and left weakening rules correspond to the layered and wildcard patterns in ML or Haskell. In this calculus however, as oposed to practical languages, we can build patterns of arbitrary depth.

While passing some basic sanity tests such as decidability of typechecking, uniqueness of types, subject reduction, and termination of recursion-free programs, this formalism has a lot of aspects to be discovered, such as interpretations in ccc's, general reduction systems, and extensions dealing with a new class of ''deep primitive recursive'' algorithms that the usual typed lambda calculi do not directly express.

Joint work with Delia Kesner and Laurence Puel, INRIA and Paris XI.

Strong normalization for the theory of constructions: a Kripke-like interpretation

Jean Gallier

Abstract: A new proof of strong normalization for the theory of constructions (under β-conversion) is presented. Previous proofs are either incorrect (including Coquand's proof of normalization given in his thesis) or use infinite contexts, except for the proof given by Geuvers and Nederhof (1991). In this last proof, strong normalization in the theory of constructions is reduced to strong normalization in Girard's system F_{ω}, via a fairly long and complex argument. The proof sketched here (in joint work with Coquand) is more direct, does not use infinite contexts, and uses a kind of {\it Kripke interpretation\/} which suggests a possible relationship to the Mitchell-Moggi Kripke models of the simply-typed lambda calculus.

Carl Gunter:

We describe the abstract syntax and the operational semantics of a higher-order functional programming langauge. The language, which we call {\it RAVL} for {\it R}ecords {\it A}nd {\it V}ariants {\it L}anguage, has a polymorphic type system that supports flexible programming with records and variants. We prove that the type system for RAVL insures the absence of certain runtime type errors (such as selecting a field from a record where that field is missing). Our analysis includes a case study, using RAVL, of the nature of such proofs for languages with an operational semantics given using proof rules in the form sometimes known as 'natural' semantics.

==

David MacQueen:

Title: Higher-order functors in Standard ML

Abstract:

The Standard ML module system is application of type theory to the problem of structuring large programs and providing more flexible and powerful abstraction mechanisms for programming. Until now, the module system has been "first-order", in the sense that one could abstract over simple modules (called "structures" in Standard ML) to form parametric modules called "functors", but one could not abstract over functors to form higher-order functors.

In practice this higher-order abstraction is a natural and useful extension of the current Standard ML module system. We have developed a semantics for higher-order functors as an extension of the natural semantics formulation used in the Definition of Standard ML, and we have implemented higher-order functors in the Standard ML of New Jersey compiler. Both the semantics and implementation involve fundamentally new ideas and mechanisms to deal with the problem of propagation of sharing or identity information. Key issues are the contravariant behavior of functor signature matching and the dual elaboration of functor applications involving formal functor parameters, once at the point of functor definition and again at the point of functor application. A sketch of the definition of functor application is presented.

This is joint work with Pierre Cregut and Mads Tofte.

==

Blelloch:

Nesl: A Nested Data-Parallel Language

Guy E. Blelloch
Carnegie Mellon University

In this talk I will describe NESL, a strongly-typed, data-parallel language. NESL is intended to be used as a portable interface for programming a variety of parallel and vector supercomputers, and was designed to be particularly useful for problems with irregular and dynamic data-structures. NESL currently runs on the CM-2 and the Cray Y-MP. It generates fully parallel code and, for many algorithms, the current implementation achieves performance close to optimized machine-specific code. The language is based on a small set of

extensions to a first-order functional language. This talk will describe the data-parallel extensions and show several examples of code. It will also discuss how the parallel complexity in the Parallel Random Access Machine model can be derived from the code.

Classical logic and storage operators

Jean-Louis Krivine

An extension of second order lambda-calculus is considered, in which the underlying logic is no longer intuitionistic logic as in system F, but classical logic.
The pure lambda-calculus is then extended with a new constant C. The rule of head reduction of C is a particular case of a rule given by M. Felleisen for control operators.
It is then proved, by using the notion of "storage operator", that computation of data types is correctly handled in this frame.

SPCF: Its Model, Calculus, and Computational Power

Matthias Felleisen

This is joint work with Ramarao Kanneganti and Robert Cartwright. SPCF, a sequential extension of Plotkin's PCF, is an idealized sequential programming language that permits programmers and programs to observe the evaluation order of procedures. In this paper, we construct a fully abstract model of SPCF using a new mathematical framework suitable for defining fully abstract models of sequential functional languages. Then, we develop an extended typed \lcal\ to specify the operational semantics of SPCF and show that the calculus is complete for the constant-free sub-language. Finally, we prove that SPCF is {\it computationally complete}, that is, it can express all computable (recursively enumerable) elements in its fully abstract model.

The paper that started this research direction is a POPL'92 paper "Observable Sequentiality and Full Abstraction" by Robert Cartwright and Matthias Felleisen. One of the major challenges in denotational semantics is the construction of fully abstract models for {\it sequential} programming languages. For the past fifteen years, research on this problem has focused on developing models for PCF, an idealized functional programming language based on the typed lambda calculus. Unlike most practical languages, PCF has no facilities for {\it observing\/} and {\it exploiting\/} the evaluation order of arguments in procedures. Since we believe that such facilities are crucial for understanding the nature of sequential computation, this paper focuses on a sequential extension of PCF (called SPCF) that includes two classes of control operators: error generators and escape handlers. These new control operators enable us to construct a fully abstract model for SPCF that interprets higher types as sets of {\it error-sensitive} functions instead of {\it continuous} functions. The error-sensitive functions form a

Scott domain that is isomorphic to a domain of decision trees. We believe that the same construction will yield fully abstract models for functional languages with different control operators for observing the order of evaluation.

Rice University programming language papers are available from titan.cs.rice.edu via anonymous ftp in public/languages. The file README lists what is available.

Fully Abstract Semantics for Parallel Programs

Stephen BROOKES
Carnegie Mellon University
School of Computer Science

ABSTRACT

This talk focuses on the behavior of programs in a standard shared variable imperative parallel programming language. The classical semantics, due to Hennessy and Plotkin, uses a recursively defined domain of ''resumptions'' and fails to validate certain natural program equivalences. Moreover, the resumptions semantics cannot give a proper account of the behavior of program under fairness or finite-delay assumptions. I introduce a new semantics with several attractive features: it has an intuitively clean and simple structure, is fully abstract with respect to partial correctness behavior, it can be adapted to cope with deadlock and with total correctness, and it models fair execution adequately. The semantics can also be varied to allow for different levels of atomicity. Each of the semantics is fully abstract with respect to the relevant notion of program behavior: two phrases have the same meaning if and only if they are interchangeable in all program contexts without affecting the behavior of the overall program. As a consequence, these semantic models support compositional (or modular) reasoning about partial and total correctness and about deadlock-freedom of parallel programs, with or without fairness assumptions.

On completeness for typed lambda calculus with bottom

Stavros COSMADAKIS
IBM T.J. Watson Research Center

Full abstraction results can be viewed as a tool to develop reasoning principles for observational equivalence of program phrases; instead of reasoning about observational equivalence, one reasons about equality in a semantic model. Thus, since beta-eta is complete for the full continuous model, it is complete for proving observational equivalence of pure terms in PCF with parallel conditional.

I will present some ongoing research towards developing a complete proof system for typed lambda terms with a constant denoting bottom. I will also mention some related questions about sequential PCF.

Testing Equivalence for Petri Nets and CCS with Action Refinement and Self-Synchronization

Lalita JATEGAONKAR
MIT Laboratory for Computer Science

We introduce a unary "self-synchronization" operation on concurrent processes analogous to the binary operations of parallel-composition-with-synchronization found in CCS, TCSP and Process Algebra. The idea is that the self-synchronization on actions a, b and c of process P is a new process Q which acts like P, except that whenever P has a pair of concurrent transitions with labels a and b, then Q has an additional transition, labelled c, leading to the same state reachable by firing the a and b transitions.

Self-synchronization can enable sequential observers to detect a degree of concurrency: the self-synchronization on a-b-c of (a|b) has "c" as a visible trace, while the self-synchronization of (ab + ba) still fails on c. Standard trace and failure semantics equate a|b and ab+ba, and so are not compositional for self-synchronization. We show that a simple modification replacing actions by "steps", namely multisets of concurrent actions, yields semantics which are compositional for self-synchronization and all the usual CCS/TCSP operators. The resulting "step-trace" and "step-failure" semantics are in fact fully abstract for Testing Equivalence with respect to self-synchronization.

The same idea of replacing actions by steps carries over to a more fully concurrent pomset-failure semantics we developed previously. The new version of pomset-STEP-failures is fully abstract for Testing Equivalence with respect to the operations of action-refinement and self-synchronization on a safe Petri Net model of processes.

As an application of self-synchronization, we show how action-refinement in which communication occurs between refining processes can be expressed using self-synchronization and ordinary, noncommunicating refinement. Hennessy has suggested that such action-refinement-with-communication may be more useful than the noncommunicating version. Our results show that pomset-step-failure semantics is fully abstract for a simpler and more general action-refinement-with-communication operation than that considered by Hennessy.

This is joint work with Albert Meyer.

Relational Parametricity and Local Variables

P.W. O'HEARN and R.D. Tennent

J. C. Reynolds has argued that Strachey's intuitive concept of ''parametric'' (i.e., uniform) polymorphism is closely linked to {\em representation independence\/}, and used logical relations to formalize this principle in languages with type variables and user-defined types. Here, we use relational parametricity to address long-standing problems with the semantics of local-variable declarations, by showing that interactions between local and non-local entities obey certain relational criteria. The talk will begin with an overview of problematic aspects of local-variable semantics, then proceed to an explanation of how parametricity is relevant, and conclude by mentioning still unresolved problems.

The Logic of Block Structure

Arthur Lent
Massachusetts Institute of Technology
Laboratory for Computer Science

In the early 1980s Reynolds defined Specification Logic, a partial correctness logic for an ALGOL-like language (characterized by having block-structured local variables and higher order procedures). As a classical theory, Specification Logic turned out to be inconsistent. Nevertheless, Tennent, using a form of possible-world semantics originally tailored to ALGOL-like languages by Reynolds and Oles, developed a semantic interpretation of Specification Logic which demonstrated its consistency as an intuitionistic theory.

Consistency is of course a minimal condition on a logic: there remains the question of soundness. Related to soundness, and of independent significance, is the question whether these possible-world semantics are adequate--in a technical sense--for the standard operational semantics of an ALGOL-like language. For example, it was unknown whether a divergent term could have the same meaning as a convergent term in these models.

This talk will present a set of sufficient conditions for adequacy of possible-world models of ALGOL-like languages. The fact that a fragment of a model of Specification Logic is adequate has certain ramifications for the truth of formulas of Specification Logic. We will explore these ramifications and give an operational interpretation to a fragment of Specification Logic.

Full Abstraction as a Guide in Designing Language Features

Jon G. RIECKE
AT&T Bell Laboratories

The theoretical notion of full abstraction has (yet unfulfilled) potential as a tool in code verification, but it has also unexpected uses in the design of programming languages. Here we use full abstraction as a guide in designing an extension of call-by-value PCF+callcc with control delimiters. We first describe cps conversion for call-by-value PCF; show how the conversion does not preserve observational congruence, i.e., is not fully abstract; and show how it may be changed into a fully abstract translation using definable retractions. The retractions in the cps world lead to a notion of typed control delimiters, denoted by #, in the untranslated world. The control delimiters have one interesting property:

> Theorem: If M and N are closed PCF terms and M and N are observationally congruent in call-by-value PCF WITHOUT callcc, then #M and #N are observationally congruent in call-by-value PCF+callcc.

We give a small example showing why this theorem DOES NOT hold without #. In words, the theorem shows that # declares portions of a program to be "continuation-free", and forces those portions to pass continuations in tightly-controlled ways. We conclude with a discussion of the philosophical implications of having # in call-by-value PCF+callcc.

~~~~~~
~~~~~~

****

L I N E A R   L O G I C   W O R K S H O P

****

Mathematical Sciences Institute

Cornell University

Ithaca, New York

June 14 - 18, 1993

Partially supported by the U.S. Office of Naval Research and by the U.S. Army Research Office.

Program Committee: S. Abramsky, J.-Y. Girard, D. Miller, J. Mitchell, and A. Scedrov (Chair)

PROGRAM

All talks will be held in the Myron Taylor Hall Conference Room in the Cornell Law School. Registration fee is $30 ($15 for students).

Monday, June 14

8:15-9:00 Light Breakfast

9:00-9:05 Welcome

9:05-10:05 Opening Address.
J. Lambek, McGill University:
Bilinear logic in algebra and linguistics

10:05-10:35 Break

10:35-11:35 Y. Lafont, CNRS Discrete Mathematics Laboratory, Marseille:
Proof nets and interaction nets

11:40-12:10 T. Ehrhard, University Paris 7:
Hypercoherences: a denotational model of linear logic

12:10-2:00 Lunch Break

2:00-3:00 V. Pratt, Stanford University:
Chu spaces as classless objects: A mathematical alternative to logic

3:05-3:35 R. Blute, McGill University:
Modelling linear logic with vector spaces

3:35-4:00 Break

4:00-4:30 M. Barr, McGill University:
Non-symmetric *-autonomous categories

4:35-5:05 M. Abrusci, University of Rome:
Developments of noncommutative linear logic:
exchange connectives, phase semantics,
semantics of proofs, proof nets

5:15-7:00 Informal gathering at MSI, 409 College Avenue, 2nd Floor.
Drinks and light appetizers will be served.

Tuesday, June 15

8:15-9:00 Light Breakfast

9:00-10:00 A. Blass, University of Michigan:
Game semantics

10:00-10:30 Break

10:30-11:30 V. Danos, University Paris 7:
Geometry of interaction: An introduction

11:35-12:05 L. Regnier, CNRS Discrete Mathematics Laboratory, Marseille:
A local and asynchronous reduction of lambda-terms
stemming from the geometry of interaction

12:05-2:00 Lunch Break

2:00-2:30 R. Jagadeesan, Imperial College:
Game Semantics for Exponentials

2:35-3:05 F. Lamarche, Imperial College:
A linear logic for computer science

3:05-3:30 Break

3:30-4:00 G. Bellin, Oxford University:
Proof-nets without boxes and graphs with orientations

4:05-4:35 H. Schellinx, Univ. of Amsterdam and Univ. Paris 7:
Classical natural deduction and linear logic

6:00-9:00 Dinner reception at the Johnson Art Museum on the
Cornell Campus

Wednesday, June 16

8:15-9:00 Light Breakfast

9:00-10:00 P. Lincoln, SRI International:
Decision problems in linear logic

10:00-10:30 Break

10:30-11:00 M. Kanovich, Russian Humanitarian State University:
The expressive power of initial fragments of linear logic

11:05-11:35 H. Jervell, Oslo University:
Simulating computations by linear proofs

11:40-12:10 P.J. Scott, University of Ottawa:
Bounded linear logic

12:10-2:00 Lunch Break

2:00-2:30 I. Mackie, Imperial College:
Linear logic and implementations of the lambda calculus

2:35-3:05 S. Martini, University of Pisa:
A promotion rule for ILL based on two level sequents

3:10-3:40 S. Gay, Imperial College:
Confluent CCS and interaction nets

3:40-4:00 Break

4:00-6:00 Demonstration Session

Thursday, June 17

8:15-9:00 Light Breakfast

9:00-10:00 J.-M. Andreoli / R. Pareschi, ECRC Munich:
Coordination computing with linear logic

10:00-10:30 Break

10:30-11:00 V. Saraswat, Xerox PARC:
Higher-order linear concurrent constraint programming

11:05-11:35 N. Kobayashi, University of Tokyo:
Logical, testing, and observation equivalence for processes in a linear logic programming

11:40-12:10 D. Miller, University of Pennsylvania:
Some process formalisms as multiple conclusion logic programming

12:10-2:00 Lunch Break

2:00-2:30 M. Abadi, DEC SRC:
Linear logic without boxes I

2:35-3:05 G. Gonthier, INRIA Rocquencourt:
Linear logic without boxes II

5:00-8:00 Picnic dinner at Beebe Lake on the Cornell Campus

Friday, June 18

8:15-9:00 Light Breakfast

9:00-10:00 M. Moortgat, Utrecht University:

The fine-structure of linguistic resources

10:00-10:30	Break
10:30-11:00	J. Hudelmaier / P. Schroeder-Heister, University of Tuebingen: Classical Lambek logic
11:05-11:35	J. Vauzeilles, University Paris 13: Planification and taxonomic networks : a first attempt at a formalisation in linear logic
11:40-12:10	C. Gunter, University of Pennsylvania: Reference counting as a computational interpretation of linear logic
12:10-2:00	Lunch Break
2:00-3:00	A. Joyal, University of Quebec at Montreal: Games, strategies, and completion of categories
3:05-3:35	S. Abramsky, Imperial College: Interaction categories
3:35-4:00	Break
4:00-5:00	Closing Address. J.-Y. Girard, CNRS Discrete Mathematics Laboratory, Marseille: On the geometry of interaction of additives

****

Linear Logic Workshop

Mathematical Sciences Institute

Cornell University

June 14-18, 1993

List of Participants

Martin Abadi
Address: DEC SRC
130 Lytton Ave.
Palo Alto, CA 94301
U.S.A.
Email: ma@src.dec.com

Samson Abramsky
Address: Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ
UK
Email: sa@doc.imperial.ac.uk

V. Michele Abrusci
Address: Logica Matematica
Dip. di Studi Filosofici ed Epistemologici
Universita di Roma "La Sapienza"
Via Nomentana 118
00161 Roma
Italy
Email: ABRUSCI@sci.uniroma1.it

Gerard Allwein
Address: Center of Innovative Computing Applications
Office of University Computing
Poplars Building 819
Indiana University
700 E 7th Street
Bloomington, IN 47405
U.S.A.
Email: gtall@ogre.cica.indiana.edu

Jean-Marc Andreoli
Address: ECRC
Arabellastrasse 17
D-8000 Munich 81
Germany
Email: Jean-Marc.Andreoli@ecrc.de

Sergei Artemov
  Address: Steklov Mathematical Institute
  Vavilova str. 42
  Moscow 117966
  Russia
  Email: sergei@artemov.mian.su

Michael Barr
  Address: Department of Mathematics
  Burnside Hall
  McGill University
  805 Sherbrooke Street West
  Montreal, QC
  Canada H3A 2K6
  Email: barr@triples.math.mcgill.ca

Gianluigi Bellin
  Address: Wolfson College
  Oxford, OX2 6UD
  UK
  Email: bellin@ox.ac.uk

George Beshers
  Address: Hibbit, Karlsson & Sorensen, Inc.
  1080 Main St.
  Pawtucket, RI 02860
  U.S.A.
  Email: beshers@wotan.hks.com

Andreas Blass
  Address: Department of Mathematics
  University of Michigan
  Ann Arbor, Michigan 48109
  U.S.A.
  Email: ablass@umich.edu

Richard Blute
  Address: Department of Mathematics
  University of Ottawa
  585 King Edward
  Ottawa, Ontario
  Canada K1N 6N5
  Email: blute@triples.math.mcgill.ca

Alessandra Carbone
  Address: Department of Mathematics
  CUNY Graduate Center
  New York, NY 10036
  U.S.A.
  Email: ale@cunyvms1.gc.cuny.edu

Jawahar Chirimar

Address: Department of Computer and Information Science
University of Pennsylvania
200 South 33rd Street
Philadelphia, PA 19104-6389
U.S.A.
Email: chirimar@saul.cis.upenn.edu

Vincent Danos
Address: U.F.R. de Mathematiques
Universite Paris VII
Tour 45-55, 5eme Etage
2, Place Jussieu
75251 Paris Cedex 05
France
Email: danos@logique.jussieu.fr

Valeria de Paiva
Address: Computing Laboratory
University of Cambridge
New Museum Site
Pembroke Street
Cambridge CB2 3QG
UK
Email: valeria.paiva@cl.cam.ac.uk

Katherine Eastaughffe, Oxford Univ.
Address: Computing Laboratory
Oxford University
11 Keble Road
Oxford OX1 3QD
UK
Email: kae@prg.oxford.ac.uk

Thomas Ehrhard
Address: LITP
Couloir 55-56, 1er etage
Universite Paris VII
2 Place JUSSIEU
75251 PARIS CEDEX 05
France
Email: ehrhard@dmi.ens.fr

Berndt Farwer
Address: Universitaet Hamburg
FB Informatik
Vogt-Koelln-Str. 30
D-2000 Hamburg 54
Germany
Email: farwer@informatik.uni-hamburg.de

Stacy Finkelstein
Address: Department of Mathematics
University of Pennsylvania
209 South 33rd Street

Philadelphia, PA 19104-6395
U.S.A.
Email: stacy@saul.cis.upenn.edu

Marcelo Fiore
Address: LFCS
Department of Computer Science
The King's Building
University of Edinburgh
Mayfield Road
Edinburgh EH9 3JZ
UK
Email: mf@dcs.ed.ac.uk

Peter Freyd
Address: Department of Mathematics
University of Pennsylvania
209 South 33rd Street
Philadelphia, PA 19104-6395
U.S.A.
Email: pjf@saul.cis.upenn.edu

Simon Gay
Address: Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ
UK
Email: sjg3@doc.ic.ac.uk

Vijay Gehlot
Address: Dept of Computer and Info Science
103 Smith Hall
University of Delaware
Newark, DE 19716-2586
U.S.A.
Email: gehlot@udel.edu

Konstantinos Georgatos
Address: Department of Mathematics
CUNY Graduate Center
New York, NY 10036
U.S.A.
Email: geo@cunyvms1.gc.cuny.edu

Sylvia Ghilezan
Address: Department of Mathematics
Burnside Hall
McGill University
805 Sherbrooke Street West
Montreal, QC
Canada H3A 2K6
Email: ghilezan@math.mcgill.ca

Jean-Yves Girard
Address: Laboratoire de Mathematiques Discretes UPR 9016
163 route de Luminy case 930
13288 Marseille cedex 09
France
Email: girard@lmd.univ-mrs.fr
Jean-Yves.Girard@inria.fr

Sergei S. Goncharov
Address: Institute of Mathematics
Universitetski prosp. 4
Novosibirsk 90
630090 Russia
Email: gonchar@cnit.nsk.su

Georges Gonthier
Address: INRIA Rocquencourt
Domaine de Voluceau, B.P. 105
78153 Le Chesnay Cedex
France
Email: gonthier@margaux.inria.fr

Carl Gunter
Address: Department of Computer and Information Science
University of Pennsylvania
200 South 33rd Street
Philadelphia, PA 19104-6389
U.S.A.
Email: gunter@saul.cis.upenn.edu

Joshua Hodas
Address: Department of Computer and Information Science
University of Pennsylvania
200 South 33rd Street
Philadelphia, PA 19104-6389
U.S.A.
Email: hodas@saul.cis.upenn.edu

Radhakrishnan Jagadeesan
Address: Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ
UK
Email: rj2@doc.imperial.ac.uk

Herman Jervell
Address: Lingvistikk og filosofi
Boks 1102 - Universitetet
Oslo
Norway

Email: herman.jervell@ilf.uio.no

Jean-Baptiste Joinet
Address: Equipe de Logique
Tour 45-55
UFR de Mathematiques
2, place Jussieu - case 7012
75251 Paris Cedex 05
France
Email: joinet@logique.jussieu.fr

Andre Joyal
Address: Departement de Mathematiques et d'Informatique
Universite du Quebec a Montreal
Case Postale 8888 Succ "A"
Montreal, QC
Canada H3C 3P8
Email: joyal@math.uqam.ca

Max Kanovich
Address: Russian State Humanitarian University
Miusskaya pl. 6
Moscow 125267
Russia
Email: sergei@artemov.mian.su

Bruce Kapron
Address: Department of Computer Science
EOW 346
University of Victoria
Victoria, BC
Canada V8W 3P6
Email: bmkapron@csr.UVic.CA

Naoki Kobayashi
Address: Department of Information Science
Faculty of Science
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113
Japan
Email: koba@is.s.u-tokyo.ac.jp

Yves Lafont
Address: Laboratoire de Mathematiques Discretes UPR 9016
163 route de Luminy case 930
13288 Marseille cedex 09
France
Email: lafont@lmd.univ-mrs.fr
lafont@dmi.ens.fr

Francois Lamarche
Address: Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ
UK
Email: gfl@doc.ic.ac.uk

Jim Lambek
Address: Department of Mathematics
Burnside Hall
McGill University
805 Sherbrooke Street West
Montreal, QC
Canada H3A 2K6
Email: lambek@triples.math.mcgill.ca

Patrick Lincoln
Address: Computer Science Laboratory
SRI International
333 Ravenswood Ave.
Menlo Park, California 94025-3493
U.S.A.
Email: lincoln@csl.sri.com

James Lipton
Address: Department of Mathematics
Wesleyan University
Middletown, CT 06459-0128
U.S.A.
Email: lipton@crab.cs.wesleyan.edu

Ian Mackie
Address: Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ
UK
Email: im@doc.imperial.ac.uk

Simone Martini
Address: Dipartimento di Informatica
Universita di Pisa
Corso Italia, 40
56125 Pisa PI
Italy
Email: martini@di.unipi.it

Nax Mendler
Address: Department of Mathematics
University of Ottawa
585 King Edward
Ottawa, Ontario
Canada K1N 6N5

Email: nmendler@csi.uottawa.ca

Dale Miller
Address: Department of Computer and Information Science
University of Pennsylvania
200 South 33rd Street
Philadelphia, PA 19104-6389
U.S.A.
Email: dale@saul.cis.upenn.edu

John Mitchell
Address: Department of Computer Science
Stanford University
Stanford, California 94305
U.S.A.
Email: jcm@cs.stanford.edu

Michael Moortgat
Address: Research Institute for Language and Speech (OTS)
Utrecht University
Trans 10
3512 JK Utrecht
The Netherlands
Email: Michael.Moortgat@let.ruu.nl

Lawrence Moss
Address: Department of Mathematics
Rawles Hall 323
Indiana University
Bloomington, IN 47405
U.S.A.
Email: lsm@cs.indiana.edu

Philip Mulry
Address: Department of Computer Science
Colgate University
Hamilton, N.Y. 13346
U.S.A.
Email: phil@cs.colgate.edu

Koji Nakatogawa
Address: Faculty of Letters
Hokkaido University
North 10, West 7, Kita-ku
Sapporo
Japan, 060
Email: koji@csli.stanford.edu
koji@huph.hokudai.ac.jp
nakatog@tansei.cc.u-tokyo.ac.jp

Anil Nerode

Address: Department of Mathematics
White Hall
Cornell University
Ithaca, New York 14853
U.S.A.
Email: anil@math.cornell.edu

Mitsu Okada
Address: Department of Computer Science
Concordia University
1455 de Maisonneuve West
Montreal, QC
Canada H3G 1M8
Email: okada@cs.concordia.ca

Remo Pareschi
Address: ECRC
Arabellastrasse 17
D-8000 Munich 81
Germany
Email: Remo.Pareschi@ecrc.de

Marco Pedicini
Address: Universita di Roma
Italy
Email: marco@nextiac.iac.rm.cnr.it

Vaughan Pratt
Address: Department of Computer Science
Stanford University
Stanford, California 94305
U.S.A.
Email: pratt@cs.stanford.edu

Uday Reddy
Address: Department of Computer Science
University of Illinois at Urbana-Champaign
1304 West Springfield Avenue
Urbana, Illinois 61801
U.S.A.
Email: reddy@cs.uiuc.edu

Laurent Regnier
Address: Laboratoire de Mathematiques Discretes UPR 9016
163 route de Luminy case 930
13288 Marseille cedex 09
France
Email: regnier@lmd.univ-mrs.fr
regnier@logique.jussieu.fr

Kimmo Rosenthal
Adress: Department of Mathematics

Union College
Schenectady, New York 12308
U.S.A.
Email: rosenthk@gar.union.edu

Vijay Saraswat
Address: Xerox PARC
3333 Coyote Hill Road
Palo Alto, Ca 94304
U.S.A.
Email: saraswat@parc.xerox.com

Andre Scedrov
Address: Department of Mathematics
University of Pennsylvania
209 South 33rd Street
Philadelphia, PA 19104-6395
U.S.A.
Email: andre@saul.cis.upenn.edu

Harold Schellinx
Address: Department of Mathematics and Computer Science
University of Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
The Netherlands
Email: harold@fwi.uva.nl

Peter Schroeder-Heister
Address: Universitaet Tuebingen
Wilhelm-Schickard-Institut fuer Informatik
Sand 13
7400 Tuebingen
Germany
Email: schroeder-heister@mailserv.zdv.uni-tuebingen.de

Philip J. Scott
Address: Department of Mathematics
University of Ottawa
585 King Edward
Ottawa, Ontario
Canada K1N 6N5
Email: scpsg@acadvm1.uottawa.ca

Robert A.G. Seely
Address: Department of Mathematics
Burnside Hall
McGill University
805 Sherbrooke Street West
Montreal, QC
Canada H3A 2K6
Email: rags@triples.math.mcgill.ca

M.A. Taitslin
Address: Tver State University
33 Zhelyabova Str.
Tver 170013
Russia
Email: mat@tvegu.tver.su

Jacqueline Vauzeilles
Address: LIPN-Institut Galilee
Universite Paris-Nord
Avenue Jean-Baptiste Clement
93430 Villetaneuse
France
Email: jv@lipn.univ-paris13.fr