Full Abstraction and Full Completeness

Samson Abramsky

`http://web.comlab.ox.ac.uk/oucl/work/samson.abramsky/`

Oxford University Computing Laboratory

September 8th 2006

Full Abstraction

The Topic

Gordon's many contributions . . .

The Topic

Gordon's many contributions . . .

- We shall focus on Full Abstraction.

The Topic

Gordon's many contributions . . .

- We shall focus on Full Abstraction.
- We shall mainly focus on one paper!

The Topic

Gordon's many contributions . . .

- We shall focus on Full Abstraction.
- We shall mainly focus on one paper!
  LCF Considered As A Programming Language
  Theoretical Computer Science 5 (1977) 223–255.

The Topic

Gordon's many contributions . . .

- We shall focus on Full Abstraction.
- We shall mainly focus on one paper!
  LCF Considered As A Programming Language
  Theoretical Computer Science 5 (1977) 223–255.
  Known colloquially as: "The PCF paper".

The Topic

Gordon’s many contributions . . .

- We shall focus on Full Abstraction.
- We shall mainly focus on one paper!
  LCF Considered As A Programming Language
  Theoretical Computer Science 5 (1977) 223–255.
  Known colloquially as: “The PCF paper”.
- Gordon’s fourth most cited paper.

The Topic

Gordon's many contributions . . .

- We shall focus on Full Abstraction.
- We shall mainly focus on one paper!
  LCF Considered As A Programming Language
  Theoretical Computer Science 5 (1977) 223–255.
  Known colloquially as: "The PCF paper".
- Gordon's fourth most cited paper.
  (Can you guess what the top three are?)

The Topic

Gordon's many contributions . . .

- We shall focus on Full Abstraction.
- We shall mainly focus on one paper!
  LCF Considered As A Programming Language
  Theoretical Computer Science 5 (1977) 223–255.
  Known colloquially as: "The PCF paper".
- Gordon's fourth most cited paper.
  (Can you guess what the top three are?)
- A real classic.

The Topic

Gordon's many contributions . . .

- We shall focus on Full Abstraction.
- We shall mainly focus on one paper!
  LCF Considered As A Programming Language
  Theoretical Computer Science 5 (1977) 223–255.
  Known colloquially as: "The PCF paper".
- Gordon's fourth most cited paper.
  (Can you guess what the top three are?)
- A real classic.
  Advice to students: **Read the classics!**

The Context

The Context

Intellectual history is a bottomless pit!

The Context

Intellectual history is a bottomless pit!

Or: a fractal . . .

The Context

Intellectual history is a bottomless pit!

Or: a fractal . . .

What I **thought** the context was:

The Context

Intellectual history is a bottomless pit!

Or: a fractal . . .

What I **thought** the context was:

- Dana Scott, *A type-theoretical alternative to CUCH, ISWIM, OWHY*.

The Context

Intellectual history is a bottomless pit!

Or: a fractal . . .

What I **thought** the context was:

- Dana Scott, *A type-theoretical alternative to CUCH, ISWIM, OWHY*.

  Manuscript 1969. Plublished in TCS 121:1–2, 411–440, **1993**!

The Context

Intellectual history is a bottomless pit!

Or: a fractal . . .

What I **thought** the context was:

- Dana Scott, *A type-theoretical alternative to CUCH, ISWIM, OWHY*.
  Manuscript 1969. Plublished in TCS 121:1–2, 411–440, **1993**!

- Robin Milner, *Fully Abstract Models for Typed Lambda Calculi*. TCS 4:1–22, 1977.

The Context

Intellectual history is a bottomless pit!

Or: a fractal . . .

What I **thought** the context was:

- Dana Scott, *A type-theoretical alternative to CUCH, ISWIM, OWHY*.
  Manuscript 1969. Plublished in TCS 121:1–2, 411–440, **1993**!
- Robin Milner, *Fully Abstract Models for Typed Lambda Calculi*. TCS 4:1–22, 1977.
- Tait's work on Computability predicates for proving Strong Normalization.

More Context

More Context

What Gordon cites in his paper:

More Context

What Gordon cites in his paper:

- Papers by Milner and Newey on LCF.

More Context

What Gordon cites in his paper:

- Papers by Milner and Newey on LCF.
- Chris Wadsworth's work on semantics of untyped λ-calculus.

More Context

What Gordon cites in his paper:

- Papers by Milner and Newey on LCF.
- Chris Wadsworth's work on semantics of untyped λ-calculus.
- Troelstra and Howard on proof theory.

The Contribution

The Contribution

Of key importance is the **programme**, the template for future research.

The Contribution

Of key importance is the **programme**, the template for future research.

The key issue:
Correspondences and tensions between Operational and Denotational Semantics.

The Contribution

Of key importance is the **programme**, the template for future research.

The key issue:
Correspondences and tensions between Operational and Denotational Semantics.

In particular:

The Contribution

Of key importance is the **programme**, the template for future research.

The key issue:
Correspondences and tensions between Operational and Denotational Semantics.

In particular:

- The lower bound: **Computational Adequacy**.

The Contribution

Of key importance is the **programme**, the template for future research.

The key issue:
Correspondences and tensions between Operational and Denotational Semantics.

In particular:

- The lower bound: **Computational Adequacy**.
- The upper bound: **Full Abstraction**.

The Contribution

Of key importance is the **programme**, the template for future research.

The key issue:
Correspondences and tensions between Operational and Denotational Semantics.

In particular:

- The lower bound: **Computational Adequacy**.
- The upper bound: **Full Abstraction**.

(N.B. Neither notion goes by these names in the PCF paper!)

Influence of Wadsworth

Of course, instead of inventing the programming language, we could have interpreted these results directly in terms of LCF. However, we feel they exemplify a programme for the investigation of programming languages which, in fact, we learned from Wadsworth. In particular, we regard Theorems 3.1 [i.e. Computational Adequacy] and 4.3 [i.e. Full Abstraction] as analogous for PCF to corresponding theorems of his for the λ-calculus. [Theorems about **head normal forms** wrt the D_∞ model.]

Influence of Wadsworth

> *Of course, instead of inventing the programming language, we could have interpreted these results directly in terms of LCF. However, we feel they exemplify a programme for the investigation of programming languages which, in fact, we learned from Wadsworth. In particular, we regard Theorems 3.1 [i.e. Computational Adequacy] and 4.3 [i.e. Full Abstraction] as analogous for PCF to corresponding theorems of his for the λ-calculus.* [Theorems about **head normal forms** wrt the D_∞ model.]

N.B. Related results by Hyland.

Contrasts

Contrasts

- **Contrast with Dana Scott's LCF paper.**

Contrasts

- **Contrast with Dana Scott's LCF paper.** The focus on a **programming language** rather than a program logic.

Contrasts

- **Contrast with Dana Scott's LCF paper.** The focus on a **programming language** rather than a program logic. The central rôle of an **explictly formulated operational semantics**.

Contrasts

- **Contrast with Dana Scott's LCF paper.** The focus on a **programming language** rather than a program logic.
  The central rôle of an **explictly formulated operational semantics**.

- **Contrast with Robin Milner's paper on Full Abstraction**.

Contrasts

- **Contrast with Dana Scott's LCF paper.** The focus on a **programming language** rather than a program logic.
  The central rôle of an **explictly formulated operational semantics**.
- **Contrast with Robin Milner's paper on Full Abstraction**. That paper, beautifully and conclusively, closed a path. The PCF paper opened up a new territory to explore.

Contrasts

- **Contrast with Dana Scott's LCF paper.** The focus on a **programming language** rather than a program logic.
  The central rôle of an **explictly formulated operational semantics**.

- **Contrast with Robin Milner's paper on Full Abstraction**. That paper, beautifully and conclusively, closed a path. The PCF paper opened up a new territory to explore.

N.B. The puzzling nature of the Full Abstraction problem. Robin constructed a fully abstract model, and showed that it was uniquely characterized up to isomorphism. This **started** the quest . . . !

Computational Adequacy

For all **programs** M:

$$M\Downarrow \iff [\![M]\!] \neq \bot.$$

Computational Adequacy

For all **programs** M:

$$M\Downarrow \iff [\![M]\!] \neq \bot.$$

There had been similar results before in the literature (e.g. in Milne and Strachey, *Theory of Programming Languages*), but this was concise, elegant, systematic: above all, it introduced a new technique taken from Proof Theory into CS, which has proved to be of great flexibility and adaptability — a truly wonderful tool:

Computational Adequacy

For all **programs** M:

$$M\Downarrow \iff [\![M]\!] \neq \bot.$$

There had been similar results before in the literature (e.g. in Milne and Strachey, *Theory of Programming Languages*), but this was concise, elegant, systematic: above all, it introduced a new technique taken from Proof Theory into CS, which has proved to be of great flexibility and adaptability — a truly wonderful tool:

The Tait method of Computability Predicates.

Computational Adequacy

For all **programs** M:

$$M{\Downarrow} \iff [\![M]\!] \neq \bot.$$

There had been similar results before in the literature (e.g. in Milne and Strachey, *Theory of Programming Languages*), but this was concise, elegant, systematic: above all, it introduced a new technique taken from Proof Theory into CS, which has proved to be of great flexibility and adaptability — a truly wonderful tool:

The Tait method of Computability Predicates.

By the way:

Computational Adequacy

For all **programs** M:

$$M{\Downarrow} \iff [\![M]\!] \neq \bot.$$

There had been similar results before in the literature (e.g. in Milne and Strachey, *Theory of Programming Languages*), but this was concise, elegant, systematic: above all, it introduced a new technique taken from Proof Theory into CS, which has proved to be of great flexibility and adaptability — a truly wonderful tool:

The Tait method of Computability Predicates.

By the way:

Bill doesn't care!

Computational Adequacy

For all **programs** M:

$$M\Downarrow \iff [\![M]\!] \neq \bot.$$

There had been similar results before in the literature (e.g. in Milne and Strachey, *Theory of Programming Languages*), but this was concise, elegant, systematic: above all, it introduced a new technique taken from Proof Theory into CS, which has proved to be of great flexibility and adaptability — a truly wonderful tool:

The Tait method of Computability Predicates.

By the way:

Bill doesn't care!

A parting thought:

Computational Adequacy

For all **programs** M:

$$M\Downarrow \iff [\![M]\!] \neq \bot.$$

There had been similar results before in the literature (e.g. in Milne and Strachey, *Theory of Programming Languages*), but this was concise, elegant, systematic: above all, it introduced a new technique taken from Proof Theory into CS, which has proved to be of great flexibility and adaptability — a truly wonderful tool:

The Tait method of Computability Predicates.

By the way:

Bill doesn't care!

A parting thought:

Soundness is less glamorous than completeness, but is the bread and butter of everyday semantical life.

Full Abstraction and Definability

$$[\![M]\!] \sqsubseteq [\![N]\!] \iff M \lessapprox_{\mathsf{obs}} N$$

where

$$M \lessapprox_{\mathsf{obs}} N \iff \forall C[\cdot].\, C[M]\Downarrow \;\Rightarrow\; C[N]\Downarrow.$$

Full Abstraction and Definability

$$[\![M]\!] \sqsubseteq [\![N]\!] \iff M \lessapprox_{\mathsf{obs}} N$$

where

$$M \lessapprox_{\mathsf{obs}} N \iff \forall C[\cdot].\, C[M]\Downarrow \;\Rightarrow\; C[N]\Downarrow.$$

- A key feature of the discussion of full abstraction is the prominence given to **definability**, and the **fine structure** of the models. This has been very fruitful!

Full Abstraction and Definability

$$[\![M]\!] \sqsubseteq [\![N]\!] \iff M \lessapprox_{\mathsf{obs}} N$$

where

$$M \lessapprox_{\mathsf{obs}} N \iff \forall C[\cdot].\, C[M]\Downarrow \Rightarrow C[N]\Downarrow.$$

- A key feature of the discussion of full abstraction is the prominence given to **definability**, and the **fine structure** of the models. This has been very fruitful!
  Later developments: Concrete Domains, Sequential Algorithms, Game Semantics etc.

Full Abstraction and Definability

$$[\![M]\!] \sqsubseteq [\![N]\!] \iff M \lesssim_{\mathsf{obs}} N$$

where

$$M \lesssim_{\mathsf{obs}} N \iff \forall C[\cdot].\, C[M]\Downarrow \;\Rightarrow\; C[N]\Downarrow.$$

- A key feature of the discussion of full abstraction is the prominence given to **definability**, and the **fine structure** of the models. This has been very fruitful!
  Later developments: Concrete Domains, Sequential Algorithms, Game Semantics etc.

- Non-definability results. Use of the Context Lemma.

Full Abstraction and Definability

$$[\![M]\!] \sqsubseteq [\![N]\!] \iff M \lesssim_{\mathsf{obs}} N$$

where

$$M \lesssim_{\mathsf{obs}} N \iff \forall C[\cdot].\, C[M]\Downarrow \;\Rightarrow\; C[N]\Downarrow.$$

- A key feature of the discussion of full abstraction is the prominence given to **definability**, and the **fine structure** of the models. This has been very fruitful!
  Later developments: Concrete Domains, Sequential Algorithms, Game Semantics etc.
- Non-definability results. Use of the Context Lemma.
- The strategy of expanding the language to match the model.

Definability and Computability

Definability and Computability

The theme here is to find a perfect match between what can be defined in the language, and the computable elements of the model.

Definability and Computability

The theme here is to find a perfect match between what can be defined in the language, and the computable elements of the model.

> *Of course this is all really recursion theory, and continues some work of Scott . . . But it also indicates how one might use denotational semantics to find facilities which may have been unintentionally left out by the language designer.*

Definability and Computability

The theme here is to find a perfect match between what can be defined in the language, and the computable elements of the model.

> *Of course this is all really recursion theory, and continues some work of Scott . . . But it also indicates how one might use denotational semantics to find facilities which may have been unintentionally left out by the language designer.*

- A real tour de force is the proof that the **existential quantifier** is not definable in PCF, even from parallel or.

Definability and Computability

The theme here is to find a perfect match between what can be defined in the language, and the computable elements of the model.

> *Of course this is all really recursion theory, and continues some work of Scott . . . But it also indicates how one might use denotational semantics to find facilities which may have been unintentionally left out by the language designer.*

- A real tour de force is the proof that the **existential quantifier** is not definable in PCF, even from parallel or.
- Also the positive result, that adding the existential gives a language complete wrt computable elements of the model.

Definability and Computability

The theme here is to find a perfect match between what can be defined in the language, and the computable elements of the model.

> *Of course this is all really recursion theory, and continues some work of Scott . . . But it also indicates how one might use denotational semantics to find facilities which may have been unintentionally left out by the language designer.*

- A real tour de force is the proof that the **existential quantifier** is not definable in PCF, even from parallel or.
- Also the positive result, that adding the existential gives a language complete wrt computable elements of the model.
- Later developments: Logical Full Abstraction with John Longley. (And analogous results for Game Semantics etc.)

And there is more!

E.g. results on classes of interpretations and weak initiality.

Some Later History

Some Later History

Gordon's own subsequent work does not feature many papers concerned directly with Full Abstraction.

Some Later History

Gordon's own subsequent work does not feature many papers concerned directly with Full Abstraction.

Two notable exceptions:

- *Full Abstraction for a Simple Parallel Programming Language* with Matthew Hennessy.

Some Later History

Gordon's own subsequent work does not feature many papers concerned directly with Full Abstraction.

Two notable exceptions:

- *Full Abstraction for a Simple Parallel Programming Language* with Matthew Hennessy.
- The paper on *Logical Full Abstraction* with John Longley.

The French School

The French School

- Berry's **stable functions**.

The French School

- Berry's **stable functions**.
- Berry and Curien's **sequential algorithms on concrete data structures**.

The French School

- Berry's **stable functions**.
- Berry and Curien's **sequential algorithms on concrete data structures**.
- Bucciarelli and Ehrhard's **strongly stable functions**.

The French School

- Berry's **stable functions**.
- Berry and Curien's **sequential algorithms on concrete data structures**.
- Bucciarelli and Ehrhard's **strongly stable functions**.
- Curien, Plotkin and Winskel **Bistructures and Bidomains**.

The 1990's

The 1990's

Two major developments:

The 1990's

Two major developments:

- **Game Semantics**. Has proved to be a very fruitful and powerful approach for constructing fully abstract models of a wide range of programming languages (and logics). Its early stages very strongly and directly motivated by "The PCF paper" and related ideas (as well as by Linear Logic).

The 1990's

Two major developments:

- **Game Semantics**. Has proved to be a very fruitful and powerful approach for constructing fully abstract models of a wide range of programming languages (and logics). Its early stages very strongly and directly motivated by "The PCF paper" and related ideas (as well as by Linear Logic).
- **Loader's theorem**:
  Observation equivalence for finitary PCF (*i.e.* over the Booleans) is undecidable.

The 1990's

Two major developments:

- **Game Semantics**. Has proved to be a very fruitful and powerful approach for constructing fully abstract models of a wide range of programming languages (and logics). Its early stages very strongly and directly motivated by "The PCF paper" and related ideas (as well as by Linear Logic).
- **Loader's theorem**:
  Observation equivalence for finitary PCF (*i.e.* over the Booleans) is undecidable.
  The definitive result on PCF.

The 1990's

Two major developments:

- **Game Semantics**. Has proved to be a very fruitful and powerful approach for constructing fully abstract models of a wide range of programming languages (and logics). Its early stages very strongly and directly motivated by "The PCF paper" and related ideas (as well as by Linear Logic).
- **Loader's theorem**:
  Observation equivalence for finitary PCF (*i.e.* over the Booleans) is undecidable.
  The definitive result on PCF.

N.B. In many ways, PCF proves to be a (bad) singularity in the space of programming languages. Where more behaviour is observable, things can work much better. (E.g. Game Semantics gives fully abstract models without resorting to an "extensional collapse").

What's it all about?

What’s it all about?

From my paper in the Festschrift for Robin:

> *The importance of full abstraction for the semantics of programming languages is that it is one of the few quality filters we have. Specifically, it provides a clear criterion for assessing how definitive a semantic analysis of some language is. It must be admitted that to date the quest for fully abstract models has not yielded many obvious applications; but it has generated much of the deepest work in semantics. Perhaps it is early days yet.*

What's it all about?

From my paper in the Festschrift for Robin:

> *The importance of full abstraction for the semantics of programming languages is that it is one of the few quality filters we have. Specifically, it provides a clear criterion for assessing how definitive a semantic analysis of some language is. It must be admitted that to date the quest for fully abstract models has not yielded many obvious applications; but it has generated much of the deepest work in semantics. Perhaps it is early days yet.*

Since then, there has been a substantial development of **algorithmic game semantics** (A, Ong, Ghica, Murawski, Lazic, Dimowski, Walukiewicz, Ouaknine, . . .), with applications in verification and program analysis. The idea is to exploit the algorithmic nature of game semantics to get automata-theoretic representations of fully abstract models. This does put a considerably more practical gloss on the notion.

The Work Goes On

A dichotomy of tastes in the Semantics community?

The Work Goes On

A dichotomy of tastes in the Semantics community?

- Functions or Processes?

The Work Goes On

A dichotomy of tastes in the Semantics community?

- Functions or Processes?
- Intensions or Extensions?

The Work Goes On

A dichotomy of tastes in the Semantics community?

- Functions or Processes?
- Intensions or Extensions?

Here we come to the underlying attitudes which inform why we choose to study the things we do. Not often spoken about . . .

The Work Goes On

A dichotomy of tastes in the Semantics community?

- Functions or Processes?
- Intensions or Extensions?

Here we come to the underlying attitudes which inform why we choose to study the things we do. Not often spoken about . . .

Meanwhile, **there is a lot of good work going on** . . .

The Work Goes On

A dichotomy of tastes in the Semantics community?

- Functions or Processes?
- Intensions or Extensions?

Here we come to the underlying attitudes which inform why we choose to study the things we do. Not often spoken about . . .

Meanwhile, **there is a lot of good work going on** . . .

- Foundational work by Ong and Murawski interweaving semantics and complexity in a deep and beautiful way.
- Work by Ghica, Lazic et al on compositional software model checking and program analysis.

Full Completeness

Full Completeness

The aim: characterize — and expose the structure of — the “space of proofs” of a Logic.

Full Completeness

The aim: characterize — and expose the structure of — the “space of proofs” of a Logic.

This seems difficult! Proof systems can be presented in so many way! It seems hard to see any robust, invariant structure.

Full Completeness

The aim: characterize — and expose the structure of — the "space of proofs" of a Logic.

This seems difficult! Proof systems can be presented in so many way! It seems hard to see any robust, invariant structure.

It is perhaps useful to make an analogy with Geometry. A major concern of modern Geometry has been to find **instrinsic**, typically **coordinate-free**, descriptions of the geometric objects of study. We may view the rôle of **syntax** in Proof Theory as analogous to coordinates in Geometry; invaluable for computation, but an obstacle to finding the underlying invariant structure.

Full Completeness

The aim: characterize — and expose the structure of — the "space of proofs" of a Logic.

This seems difficult! Proof systems can be presented in so many way! It seems hard to see any robust, invariant structure.

It is perhaps useful to make an analogy with Geometry. A major concern of modern Geometry has been to find **instrinsic**, typically **coordinate-free**, descriptions of the geometric objects of study. We may view the rôle of **syntax** in Proof Theory as analogous to coordinates in Geometry; invaluable for computation, but an obstacle to finding the underlying invariant structure.

Progress in finding more intrinsic descriptions of proofs, their geometric structure, and their dynamics under Cut-elimination, has taken place in the study of **proof-nets in Linear Logic**. On the semantic side, the development of **Game Semantics and Full Completeness results**.

Two Views of Logic

Two Views of Logic

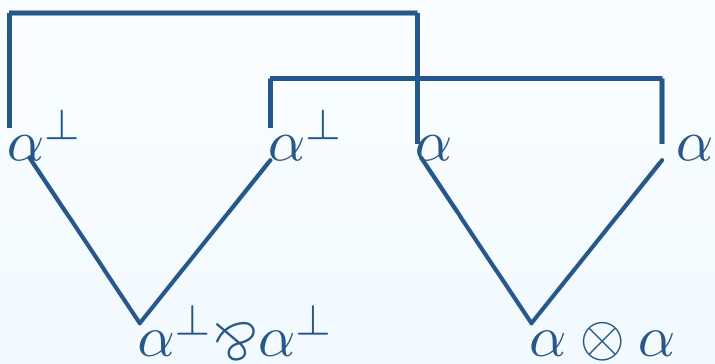
1. **The Descriptive View.** Logic is used to **talk about** structure. This is the view taken in Model Theory, and in most of the uses of Logic (Temporal logics, MSO etc.) in Verification. It is by far the more prevalent and widely-understood view.

Two Views of Logic

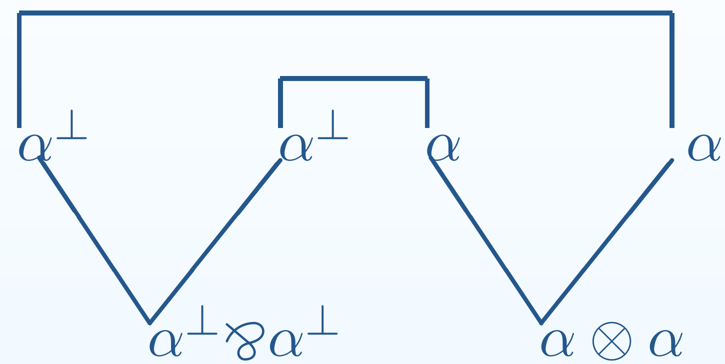
1. **The Descriptive View.** Logic is used to **talk about** structure. This is the view taken in Model Theory, and in most of the uses of Logic (Temporal logics, MSO etc.) in Verification. It is by far the more prevalent and widely-understood view.

2. **The Intrinsic View.** Logic is taken to **embody** structure. This is, implicitly or explicitly, the view taken in the Curry-Howard isomorphism, and more generally in Structural Proof Theory, and in (much of) Categorical Logic. In the Curry-Howard isomorphism, one is not using logic to **talk about** functional programming; rather, logic (in this aspect) **is** functional programming.

MLL: A Logical Paradise



Id



Twist

The essential information in a (cut-free) proof in MLL is the axiom links.

Accordingly, we define a **proof structure** on a sequent Γ to be a fixpoint-free involution f (so $f^2 = 1$ and $f(a) \neq a$) on its occurrences of literals such that if $f(a) = b$, $l(a) = l(b)^\perp$.

Assignment of Permutations to Sequent Proofs

Axiom

$$\frac{}{\vdash a, a^{\perp}}\ \mathsf{Id}$$

Multiplicatives

$$\frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\ \otimes \qquad \frac{\Gamma, A, B}{\vdash \Gamma, A \invamp B}\ \invamp$$

- Axiom: assign the transposition $a \leftrightarrow a^{\perp}$
- Tensor: assign the disjoint union of the two permutations
- Par: assign the same permutation!

MLL Proof Nets

Which proof structures really come from proofs in MLL?

Switching Graphs: A **switching** S of Γ assigns L or R to each occurrence of $\parr$. Given a sequent Γ, a proof structure f, and a switching S, the **switching graph** $G(\Gamma, f, S)$ has:

- subformula occurrences in Γ as vertices;
- an edge connecting A to $A \otimes B$ and an edge connecting B to $A \otimes B$ for each occurrence of $A \otimes B$;
- an edge connecting A to $A \parr B$ if S assigns L to $A \parr B$, and an edge connecting B to $A \parr B$ if S assigns R to $A \parr B$;
- an edge connecting literal occurrences a and b if $f(a) = b$.

The Danos-Regnier criterion: A proof-structure f for Γ is an MLL proof-net if for every switching S, $G(\Gamma, f, S)$ is **acyclic and connected**.

Results on Proof Nets

Every sequent proof in MLL canonically maps to a proof structure.

Proposition 1 (Soundness) *The proof structures arising from sequent proofs are proof nets.*

Theorem 2 (Sequentialization Theorem) *Every proof net arises from a sequent proof.*

This is the **Geometric Criterion**.

Understanding the Proof Net condition game-theoretically

We can think of a proof structure (set of axiom links) as a **copy-cat strategy**, and a switching as a counter-strategy. A proof structure will be a proof-net if its interaction with every counter-strategy yields a correct result.

Hence we define (Girard 1988):

$$f \perp g \quad \equiv \quad fg \text{ is cyclic}$$

i.e. $(fg)^k = 1$ where k is the cardinality of the underlying set (of literal occurrences), and this is true for no smaller value of k.

This condition is directly inspired by the **long trip condition**, the earlier version of the proof net correctness condition used by (Girard 1987).

We can then define

$$S^{\perp} = \{g \mid \forall f \in S.\ f \perp g\}.$$

Semantics of MLL Proofs

We now give a semantics of MLL proofs by specifying, for each formula A, a set S of permutations on the set of literal occurrences $|A|$, such that $S = S^{\perp\perp}$.
For a literal, we specify the unique permutation (the identity).

$$
\begin{array}{rcl}
S(A \otimes B) & = & \{f + g \mid f \in S(A) \ \wedge \ g \in S(B)\}^{\perp\perp} \\
S(A \parr B) & = & S(A^{\perp} \otimes B^{\perp})^{\perp}.
\end{array}
$$

Note that, for every formula A: $S(A^{\perp}) = S(A)^{\perp}$.

We extend this assignment to sequents Γ by treating Γ as the Par of its formulas.

Semantics: Soundness and Completeness

Proposition 3 (Semantic Soundness) *If f is the permutation assigned to a sequent proof of Γ, then $f \in S(\Gamma)$.*

Theorem 4 (Full Completeness) *If $f \in S(\Gamma)$ is a literal-respecting fixpoint-free involution, then f is a proof-net, and hence is the denotation of a sequent proof.*

This shows **the equivalence of the geometric and interactive criteria for proofs**.

The Plan: Given $\sigma \in S(\Gamma)$, we assume that for some switching S, $\mathbb{G}_{\Gamma}(\sigma, S)$ is not a tree. Then we construct a **counter-strategy** $\tau \in S(\Gamma)^{\perp}$ such that $\neg(\sigma \perp \tau)$. Contradiction.

Further Issues

Further Issues

- Cut Elimination.

Further Issues

- Cut Elimination.
- Uniformity. Getting the proof structure conditions to fall out automatically.

Further Issues

- Cut Elimination.
- Uniformity. Getting the proof structure conditions to fall out automatically.
- Free categories.

MALL: Proof Nets

Now we begin . . .

Now we begin . . .

MALL (Multiplicative Additive Linear Logic) is much harder than MLL!

Upgrading the beautiful MLL theory to MALL is surprisingly difficult:

Now we begin . . .

MALL (Multiplicative Additive Linear Logic) is much harder than MLL! Upgrading the beautiful MLL theory to MALL is surprisingly difficult:

- Much harder to find good notions of proof structure and proof net, and to prove Sequentialization

Now we begin . . .

MALL (Multiplicative Additive Linear Logic) is much harder than MLL! Upgrading the beautiful MLL theory to MALL is surprisingly difficult:

- Much harder to find good notions of proof structure and proof net, and to prove Sequentialization
- Harder to find good models (as discussed previously), and to prove Full Completeness.

Now we begin . . .

MALL (Multiplicative Additive Linear Logic) is much harder than MLL! Upgrading the beautiful MLL theory to MALL is surprisingly difficult:

- Much harder to find good notions of proof structure and proof net, and to prove Sequentialization
- Harder to find good models (as discussed previously), and to prove Full Completeness.

This is what we've been doing, building on previous work:

Now we begin . . .

MALL (Multiplicative Additive Linear Logic) is much harder than MLL! Upgrading the beautiful MLL theory to MALL is surprisingly difficult:

- Much harder to find good notions of proof structure and proof net, and to prove Sequentialization
- Harder to find good models (as discussed previously), and to prove Full Completeness.

This is what we've been doing, building on previous work:

- New Foundations for the Geometry of Interaction. A and Radha Jagadeesan. LiCS 1992.
- Concurrent Games and Full Completeness. A and Paul-Andre Mellies. LiCS 1999.

A seven-year itch.

New issues and structures in MALL

New issues and structures in MALL

- The additives involve **choice, temporality, causality**; whereas the multiplicatives are **atemporal, concurrent**. (Behavioural notions in fine particles of logic!)

New issues and structures in MALL

- The additives involve **choice, temporality, causality**; whereas the multiplicatives are **atemporal, concurrent**. (Behavioural notions in fine particles of logic!)
- We use domain-theoretic structure in a non-trivial way to get elegant semantic notions of proof structure and proof net, and to express the **geometric criterion** and prove Sequentialization. **Stability** (in Berry's sense) plays a key role.

New issues and structures in MALL

- The additives involve **choice, temporality, causality**; whereas the multiplicatives are **atemporal, concurrent**. (Behavioural notions in fine particles of logic!)
- We use domain-theoretic structure in a non-trivial way to get elegant semantic notions of proof structure and proof net, and to express the **geometric criterion** and prove Sequentialization. **Stability** (in Berry's sense) plays a key role.
- We use the same structure to express the **interactive criterion**, using **least fixpoints** to formulate interaction of proofs (Cut Elimination!).

New issues and structures in MALL

- The additives involve **choice, temporality, causality**; whereas the multiplicatives are **atemporal, concurrent**. (Behavioural notions in fine particles of logic!)
- We use domain-theoretic structure in a non-trivial way to get elegant semantic notions of proof structure and proof net, and to express the **geometric criterion** and prove Sequentialization. **Stability** (in Berry's sense) plays a key role.
- We use the same structure to express the **interactive criterion**, using **least fixpoints** to formulate interaction of proofs (Cut Elimination!).
- We show the equivalence of the geometric and interactive criteria.

New issues and structures in MALL

- The additives involve **choice, temporality, causality**; whereas the multiplicatives are **atemporal, concurrent**. (Behavioural notions in fine particles of logic!)
- We use domain-theoretic structure in a non-trivial way to get elegant semantic notions of proof structure and proof net, and to express the **geometric criterion** and prove Sequentialization. **Stability** (in Berry's sense) plays a key role.
- We use the same structure to express the **interactive criterion**, using **least fixpoints** to formulate interaction of proofs (Cut Elimination!).
- We show the equivalence of the geometric and interactive criteria.

So a lot of ideas developed in the PCF line of work play an important rôle!

HvG Proof Structures

Let Γ be a MALL sequent. Let W be the set of occurrences of Withs ($\&$) in Γ. Let $\mathcal{V}$ be the poset of partial functions from W into $\{0,1\}$, ordered by inclusion. Equivalently, $\mathcal{V} = \mathbb{B}_{\perp}^{W}$, a product of flat domains. Let $D(\Gamma)$ be the poset defined (as before) by:

$$D(A \& B) = D(A \oplus B) = (D(A) + D(B))_{\perp},$$

$$D(A \otimes B) = D(A \parr B) = D(A) \times D(B).$$

An **HvG proof structure** for Γ is a function

$$f : \mathsf{Max}(\mathcal{V}) \longrightarrow (\Sigma x \in \mathsf{Max}(D(\Gamma)))S(|x|)$$

which assigns to each **total Boolean valuation** of the With-occurrences a pair (x, π), where x is a maximal element of $D(\Gamma)$ (an “additive resolution of Γ”), and π is a permutation on the corresponding set of literal occurrences.
Note that if there are no additives, this reduces to the “old” notion for MLL. If there are no Withs, we simply get a resolution of all $\oplus$, and an assignment of a set of axiom links.

Monotone Extensions

Our definition of HvG proof structures is deliberately phrased to provoke the question: what about **partial** With valuations?
We shall define a MALL proof structure to be a **monotone** function

$$f : \mathcal{V} \longrightarrow (\Sigma x \in D(\Gamma))S^{\partial}(|x|)$$

which to each (partial) Boolean valuation of the With-occurrences assigns a pair (x, π) where $x \in D(\Gamma)$ and π is a **partial** permutation on the set of literal occurrences defined at x. We require that the restriction to $\mathsf{Max}(\mathcal{V})$ is an HvG proof structure.

Note that a given HvG proof structure f can have many monotone extensions. There will always be a **greatest one** $\hat{f}$, defined by

$$\hat{f}(v) = \bigwedge\{f(v') \mid v \leq v' \in \mathsf{Max}(\mathcal{V})\}.$$

This is the "maximally parallel extension", and can be seen as the canonical representative of the HvG proof structure.

Stable Extensions

We can also consider further properties of the monotone function, e.g. **stability** (preservation of bounded meets). This “semantic” property is the essence of Girard’s **monomial condition**.

Note that the maximal extension will **not** in general be stable! (Think of monotone extensions of the logical or function on the Booleans: the maximal extension is parallel or!)

The Dependency Relation

Recall that an equivalent definition of stability is the **minimum data property**: for every input v, and $y \leq f(v)$, there is a minimum value $M(f, v, y) \leq v$ such that $y \leq f(M(f, v, y))$.

Let f be a stable proof structure for Γ, and $v \in \mathsf{Max}(\mathcal{V})$. Let $f(v) = (x, \pi)$. Given a With occurrence w and literals a, b in $|x|$ with $\pi(a) = b$, we say that **the axiom link** $a \frown b$ **depends on** w **in** v if $v'(w)\downarrow$, where $v' = M(f, v, (p_a \sqcup p_b, \{(a, b)\})$.
(Here $p_a \sqsubseteq x$ is the **prime** corresponding to the literal occurrence a).

Now given $v \in \mathsf{Max}(\mathcal{V})$ with $f(v) = (x, \pi)$, define a switching S to be an assignment of L or R to every occurrence of $\invamp$ in x, and a choice of a **jump** for every occurrence w of a With in x, where a jump is either **normal** — the premise of w specified by $v(w)$ — or **proper** — an occurrence or link l depending on w in v.

Stable MALL Proof Nets

We can then define a switching graph $G(\Gamma, f, v, S)$ with:

- the vertices given by the subformula occurrences in x;
- an edge connecting A to $A \otimes B$ and an edge connecting B to $A \otimes B$ for each occurrence of $A \otimes B$;
- an edge connecting A to $A \parr B$ if S assigns L to $A \parr B$, and an edge connecting B to $A \parr B$ if S assigns R to $A \parr B$;
- an edge connecting literal occurrences a and b if $\pi(a) = b$;
- an edge connecting each $\oplus$ to its unique premise in x;
- an edge connecting each With occurrence to its jump as specified by S.

We say that f is a **stable MALL proof net** if for every $v \in \mathsf{Max}(\mathcal{V})$ and switching S, $G(\Gamma, f, v, S)$ is connected and acyclic.

Assignment of Proof Structures to MALL Sequent Proofs

Multiplicative part: as for MLL

Additives

$$\frac{\vdash \Gamma, A}{\vdash \Gamma A \oplus B}\ \oplus\text{L} \qquad \frac{\vdash \Gamma, B}{\vdash \Gamma A \oplus B}\ \oplus\text{R} \qquad \frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \& B}\ \&$$

- $\oplus$: compose with the injection
- With: a case statement on $v(w)$. Suppose the proof structures assigned to the two premises of the rule are f and g. We write a valuation in $\mathcal{V}(\Gamma)$ as (v, b), where $b \in \mathbb{B}_\perp$ is the value assigned to the With occurrence appearing in the conclusion of the rule. Then we assign the proof structure h to the conclusion, where:

$$\begin{array}{rcl} h(v, 0) & = & f(v) \\ h(v, 1) & = & g(v) \\ h(v, \perp) & = & \perp. \end{array}$$

Discussion

Note that the last equation in this definition is **the only place where any latitude appears** in the definition of the assignment of proof structures to sequent proofs. The above definition can be written as a conditional:

$$h(v, b) = \mathbf{if}\ b\ \mathbf{then}\ f(v)\ \mathbf{else}\ g(v).$$

This is the usual **sequential conditional**. (The maximally parallel extension of the HvG proof structure would do a **parallel conditional** here, violating stability.)

Results

Proposition 5 *For every sequent proof, the corresponding proof structure given by the above assignment is sequential.*

Theorem 6 (Soundness) *For every sequent proof, its denotation as a proof structure is a stable proof net.*

The major result on proof nets is the Sequentialization Theorem (Gir87,DR,Gir91,Gir95).

Theorem 7 (Sequentialization) *For every stable proof net f, there is a sequent proof Π such that $g \sqsubseteq f$ (in the pointwise order), where $g = [\![\Pi]\!]$ is the sequential proof net assigned to Π.*
N.B. This implies that $f^{\mathsf{m}} = g^{\mathsf{m}}$.

MALL: Strategies and Full Completeness

Projection onto Withs

Firstly, we define a function

$$\mathsf{out} : D(\Gamma) \longrightarrow \mathcal{V}$$

which extracts the (partial) assignment to Withs from an element of $D(\Gamma)$.
We then define

$$\mathbf{p} : \mathcal{E}(\Gamma) \longrightarrow \mathcal{V}(\Gamma) :: (d, \pi) \mapsto \mathsf{out}(d).$$

By the way, note that we should always have

$$\mathbf{p}(f(v)) \leq v$$

as a general sanity condition.
Another condition:

$$f = f \circ \mathbf{p} \circ f$$

takes the place of Girard's "technical condition".

Pre-Strategies and Orthogonality

We define a **pre-strategy** for a MALL sequent Γ to be a monotone function

$$f : \mathcal{E}(\Gamma) \longrightarrow \mathcal{E}(\Gamma).$$

Note that a proof structure $f : \mathcal{V}(\Gamma) \longrightarrow \mathcal{E}(\Gamma)$ induces a pre-strategy

$$\bar{f} = f \circ \mathbf{p} : \mathcal{E}(\Gamma) \longrightarrow \mathcal{E}(\Gamma).$$

The Orthogonality Relation

Now given prestrategies $f, g : \mathcal{E}(\Gamma) \rightarrow \mathcal{E}(\Gamma)$ we define

$$F : \mathcal{E}(\Gamma)^2 \longrightarrow \mathcal{E}(\Gamma)^2 :: (x, y) \mapsto (g(y), f(x)).$$

We then define $\langle f \mid g \rangle = ((x, \pi), (y, \rho))$ to be the **least fixed point** of F.

We say that $f \perp g$ if $x = y \in \mathsf{Max}(D(\Gamma))$, and $\pi \perp \rho$ (defined as for MLL).

Thus f is orthogonal to g if each resolves all the other's additive choices, and we end up in a maximal state (complete additive resolution) with a set of axiom links (π) and "counter-links" (ρ) satisfying the usual multiplicative condition.

Just as before, we can define $S^{\perp}$, where S is a set of prestrategies.

Semantics of MALL Proofs

Just as for MLL, we define for each MALL formula A a set $S(A)$ of prestrategies such that $S = S^{\perp\perp}$. In particular:

$$S(A \oplus B) = (\{\mathsf{inl}(f) \mid f \in S(A)\} \cup \{\mathsf{inr}(g) \mid g \in S(B)\})^{\perp\perp}$$

$$S(A \& B) = S(A^{\perp} \oplus B^{\perp})^{\perp}.$$

Proposition 8 (Semantic Soundness) *If f is the stable proof structure assigned to a sequent proof of Γ, then $\bar{f} \in S(\Gamma)$.*

Theorem 9 (Full Completeness) *If f is a stable proof structure, then f is a proof-net iff $\bar{f} \in S(\Gamma)$.*

This shows the equivalence of the **geometric** and **interactive** conditions.

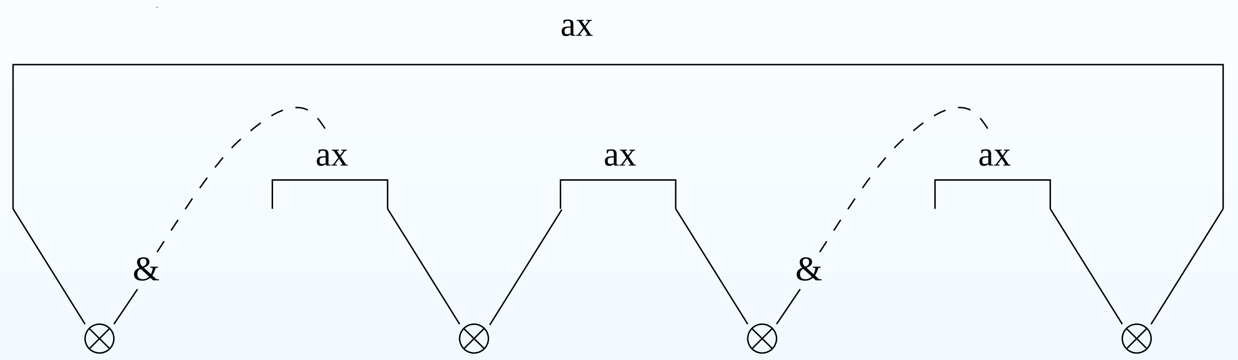
Proving Full Completeness

Same strategy as for MLL. Given $f \in S(\Gamma)$, we assume that for some valuation v and switching S, $G(\Gamma, f, S)$ is not a tree. Then we construct a **counter-strategy** $g \in S(\Gamma)^{\perp}$ such that $\neg(f \perp g)$.

The argument is more involved than for MLL.

- For acyclicity, we show that **cycles in switching graphs imply deadlocks**, and hence we get stuck at a non-maximal element, violating orthogonality.
- Once we have acyclicity, we can reduce connecteness to the Multiplicative case.

Acyclicity



Important Lemma (Mellies):

We can take cycles to be **oriented** (jumps “face” the same way).

We construct a counter-strategy which at each displayed tensor waits for information from the predecessor in the cycle. Thus we get stuck in a circular dependency, and cannot make progress.

Compositionality and Uniformity

We can make a category of such strategies. There is an elegant definition of composition of strategies, and we can interpret Cut-Elimination.

This follows A and Jagadeesan, New Foundations for GoI. The closure operator formulation of A and Mellies is a variant.

We can also consider **uniform families of strategies**, and recover the conditions on proof structures systematically.

Questions, Questions

- Understand HvG proof-net condition, relate it to the semantics.
- Analyze full range of possibilities, sequential–parallel.
- Connections to Ludics nets
- Exponentials
- And much more . . .