A Geometry of Interaction and Game Semantics Tutorial

Laurent Regnier

Institut de Mathématiques de Luminy

Geometry of Computation 2006

The logical viewpoint

Three logical levels:

- Formula: truth
- Proof: provability
- Cut elimination: coherence (subformula property)

The logical viewpoint

Three logical levels:

- Formula: truth
- Proof: provability
- Cut elimination: coherence (subformula property)

The logical viewpoint

Three logical levels:

- Formula: truth
- Proof: provability
- Cut elimination: coherence (subformula property)

The logical viewpoint

Three logical levels:

- Formula: truth
- Proof: provability
- Cut elimination: coherence (subformula property)

Curry-Howard

Model for sequential programming language, *e.g.*, (typed) lambda calculus: *use modularity (compositionality)*

- Type: space
- Program: morphisms
- Execution: introducing time in the model

Curry-Howard

Model for sequential programming language, *e.g.*, (typed) lambda calculus: *use modularity (compositionality)*

- Type: space
- Program: morphisms
- Execution: introducing time in the model

Curry-Howard

Model for sequential programming language, *e.g.*, (typed) lambda calculus: *use modularity (compositionality)*

- Type: space
- Program: morphisms
- Execution: introducing time in the model

Semantics

Embedding syntax into more general structure

- Scott continuity: finitess of computation
- Stability: (inverse) determinism
- Sequentiality: determinism

Semantics

Embedding syntax into more general structure

- Scott continuity: finitess of computation
- Stability: (inverse) determinism
- Sequentiality: determinism

What about execution?

Syntaxes for execution

- Logic: cut elimination, *ie*, beta-reduction
- Programming language: abstract machines

Head linear reduction

A machine for (weak) *head linear reduction*:

$$(\lambda\vec{x}x_i)\vec{u} \succ u_i$$

$$(\lambda\vec{x}vw)\vec{u} \succ (\lambda\vec{x}v)\vec{u}((\lambda\vec{x}w)\vec{u})$$

- KAM: closures and stack
- PAM: pointed sequences (hyper lazy KAM)
- Execution = sequence of occurrences of variables

Head linear reduction

A machine for (weak) *head linear reduction*:

$$(\lambda\vec{x}x_i)\vec{u} \succ u_i$$
$$(\lambda\vec{x}vw)\vec{u} \succ (\lambda\vec{x}v)\vec{u}((\lambda\vec{x}w)\vec{u})$$

- KAM: closures and stack
- PAM: pointed sequences (hyper lazy KAM)
- Execution = sequence of occurrences of variables

Head linear reduction

A machine for (weak) *head linear reduction*:

$$(\lambda\vec{x}x_i)\vec{u} \succ u_i$$
$$(\lambda\vec{x}vw)\vec{u} \succ (\lambda\vec{x}v)\vec{u}((\lambda\vec{x}w)\vec{u})$$

- KAM: closures and stack
- PAM: pointed sequences (hyper lazy KAM)
- Execution = sequence of occurrences of variables

Head linear reduction

A machine for (weak) *head linear reduction*:

$$(\lambda\vec{x}x_i)\vec{u} \succ u_i$$
$$(\lambda\vec{x}vw)\vec{u} \succ (\lambda\vec{x}v)\vec{u}((\lambda\vec{x}w)\vec{u})$$

- KAM: closures and stack
- PAM: pointed sequences (hyper lazy KAM)
- Execution = sequence of occurrences of variables

Game in mathematics

- Game theory (economics)
- Gentzen (coherence of arithmetics: sequent calculus proof = winning strategy)
- Descriptive set theory (determination axioms)
- Program verification
- Game semantics

Game semantics

- Two players: Environment (O) and Program (P)
- Execution = alternating sequence of moves (play)
- Program = strategy
- Type = set of plays

AJM games

- Move = finite sequence of numbers (plus multiplicative information)
- Strategy = function on moves (memory freeness)
- Equivalence between strategies: renumbering

Theorem

AJM strategy of M = GoI of M

HO games

- Play = pointed sequence (à la PAM)
- Strategy = function on views (innocence)

Theorem

Strategy = tree of views = Böhm tree

Proof.

HO play = PAM run □

Pointifixion

- AJM play $\rightsquigarrow$ HO play: $\vec{\imath}, i$ points on $\vec{\imath}$
- AJM strategy (memory free) $\rightsquigarrow$ HO strategy (innocent)

Geometry of interaction

- A reversible abstract machine (IAM)
- An interpretation of programs/proofs by operators
- An algebraic characterization of *execution paths*
- A localization of beta-reduction (sharing graphs)
- A generalization of multiplicative *experiments*
- An interpretation into a traced monoidal category

Geometry of interaction

- A reversible abstract machine (IAM)
- An interpretation of programs/proofs by operators
- An algebraic characterization of *execution paths*
- A localization of beta-reduction (sharing graphs)
- A generalization of multiplicative *experiments*
- An interpretation into a traced monoidal category

Geometry of interaction

- A reversible abstract machine (IAM)
- An interpretation of programs/proofs by operators
- An algebraic characterization of *execution paths*
- A localization of beta-reduction (sharing graphs)
- A generalization of multiplicative *experiments*
- An interpretation into a traced monoidal category

Geometry of interaction

- A reversible abstract machine (IAM)
- An interpretation of programs/proofs by operators
- An algebraic characterization of *execution paths*
- A localization of beta-reduction (sharing graphs)
- A generalization of multiplicative *experiments*
- An interpretation into a traced monoidal category

Geometry of interaction

- A reversible abstract machine (IAM)
- An interpretation of programs/proofs by operators
- An algebraic characterization of *execution paths*
- A localization of beta-reduction (sharing graphs)
- A generalization of multiplicative *experiments*
- An interpretation into a traced monoidal category

Geometry of interaction

- A reversible abstract machine (IAM)
- An interpretation of programs/proofs by operators
- An algebraic characterization of *execution paths*
- A localization of beta-reduction (sharing graphs)
- A generalization of multiplicative *experiments*
- An interpretation into a traced monoidal category

The Interaction Abstract Machine

- Program = bideterministic (reversible) automaton
- State = (B, S) + location in the graph
  - B = *box stack* of *exponential signatures*
  - S = *balanced stack* of exponential signatures + multiplicative constants P and Q
  - exponential signature = binary tree with leaves in $\{\square, R, S\}$
- Transitions = partial transformations on (B, S)

Theorem

$KAM \subset IAM$

Execution formula

- $M : A$ and $x : A \vdash N : B$ yields:

$$\pi = \begin{pmatrix} \pi_{AA} & 0 & 0 \\ 0 & \pi_{A^\perp A^\perp} & \pi_{A^\perp B} \\ 0 & \pi_{BA^\perp} & \pi_{BB} \end{pmatrix} \qquad \sigma = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- $$(1 - \sigma^2)\pi \sum_{k \geq 0} (\sigma\pi)^k (1 - \sigma^2)$$

Execution formula

- $M : A$ and $x : A \vdash N : B$ yields:

$$\pi = \begin{pmatrix} \pi_{AA} & 0 & 0 \\ 0 & \pi_{A^\perp A^\perp} & \pi_{A^\perp B} \\ 0 & \pi_{BA^\perp} & \pi_{BB} \end{pmatrix} \qquad \sigma = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- $$(1-\sigma^2)\pi \sum_{k\geq 0} (\sigma\pi)^k (1-\sigma^2)$$

The GoI equationnal theory

- Monoid with 0 generated by p, q, d, r, s, t
- Involution: $0^* = 0$, $1^* = 1$, $(uv)^* = v^*u^*$
- Morphism: $!(0) = 0$, $!(1) = 1$, $!(u)!(v) = !(uv)$, $!(u)^* = !(u^*)$
- Annihilation equations: $x^*y = \delta_{xy}$ (x, y generators)
- Commutation equations:
  - $!(u)d = du$
  - $!(u)x = x!(u)$ for $x = r, s$
  - $!(u)t = t!(!(u))$

The theorem AB^*

- Orientate equations $\rightsquigarrow$ rewriting system
- Normal forms $= 0$ or AB^*
- Inverse semigroup structure

Models of the equationnal theory

- Partial isometries on the Hilbert space
- Small models: partial injections on $\mathbb{N}$
- Partial transformations on an algebra of first order terms (clauses model, consistent semantics)

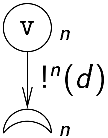
The GoI interpretation of lambda-calculus

Given M and n define the oriented graph $\mathcal{G}_n(M)$:

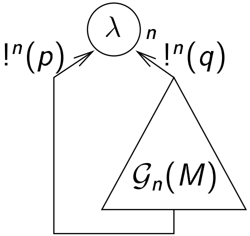
- Nodes: lambda and app, box nodes
- Edges: labelled with weight
- One exiting edge per free variable plus one entering edge for M.

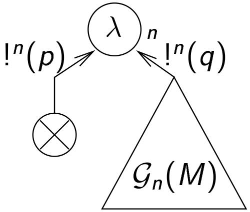
GoI of $M = \mathcal{G}_0(M)$

Variable case: $\mathcal{G}_n(x)$

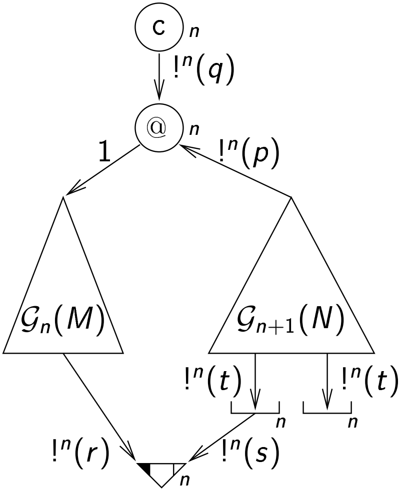


Abstraction case: $\mathcal{G}_n(\lambda x M)$





Application case: $\mathcal{G}_n(MN)$



Execution paths

Definition

Execution paths = invariant of beta-reduction = virtual redexes

Theorem

Execution paths = Regular paths = Legal paths

Corollary

Balanced execution paths = redex families