

[LE,RO]



[LE] 1

[RO] 1

[LO] [RE]

plain

An Explicit Framework for Interaction Nets

Marc de Falco

Institut de Mathématiques de Luminy
Marseille, France

Rewriting Techniques and Applications 2009

Interaction nets (Lafont):

- Model of computation
 - derived from multiplicative proof-nets of linear logic
 - relying on simple graph rewriting
 - Turing complete.
 - Strongly (one-step) confluent.
- Used when terms are too restrictive:
 - optimal reduction for λ -calculus (sharing graphs)
 - concurrency (differential interaction nets)

Goal: explicit syntax for straightforward implementation

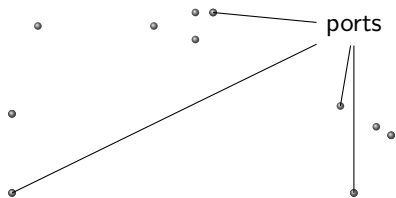
Usual presentation of interaction nets

Two parts: statics (nets) and dynamics (reduction by means of rules)

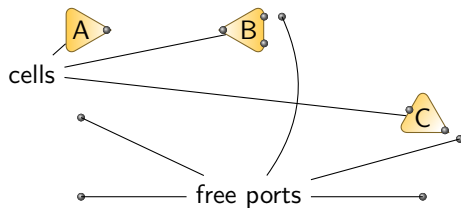
Interaction Nets: Statics



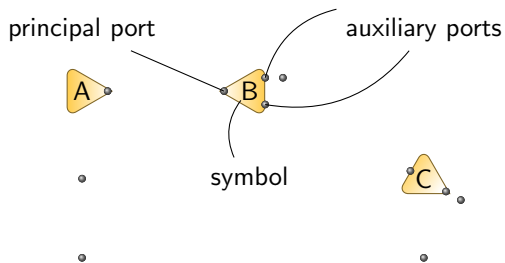
Interaction Nets: Statics



Interaction Nets: Statics

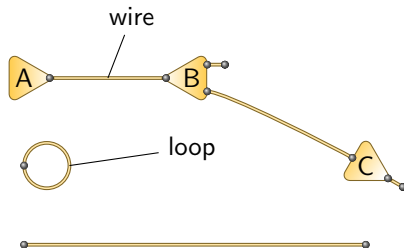


Interaction Nets: Statics



Arity of a symbol: number of auxiliary ports.

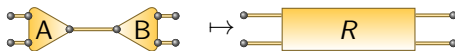
Interaction Nets: Statics



Each port belong to **exactly one** wire (*Port invariant*).
No loop from a cell port.

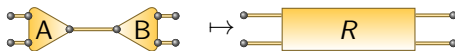
Interaction Nets: Dynamics

Rules of the form:

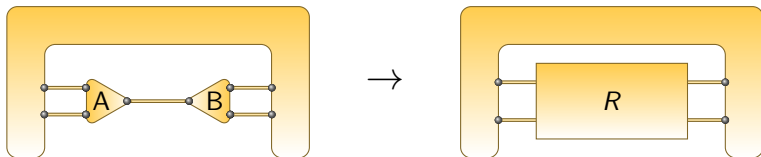


Interaction Nets: Dynamics

Rules of the form:



Derived reduction:



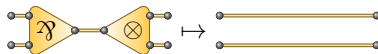
Hidden mechanism : port fusion

- Are interaction nets as simple as they look?

Hidden mechanism : port fusion

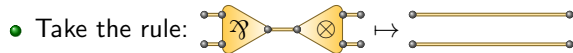
- Are interaction nets as simple as they look?

- Take the rule:

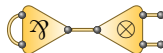


Hidden mechanism : port fusion

- Are interaction nets as simple as they look?



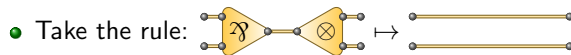
- And the net:



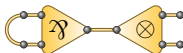
We should be able to apply the reduction.

Hidden mechanism : port fusion

- Are interaction nets as simple as they look?



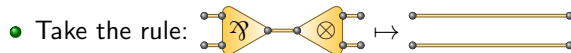
- And the net:



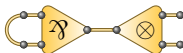
We should be able to apply the reduction. We *pull apart* the redex.

Hidden mechanism : port fusion

- Are interaction nets as simple as they look?



- And the net:



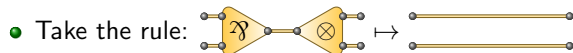
We should be able to apply the reduction. We *pull apart* the redex.

- The reduced net being:

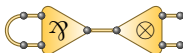


Hidden mechanism : port fusion

- Are interaction nets as simple as they look?



- And the net:



We should be able to apply the reduction. We *pull apart* the redex.

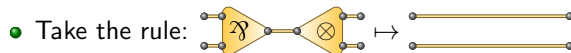
- The reduced net being:



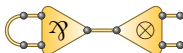
We *fuse* the unnecessary ports.

Hidden mechanism : port fusion

- Are interaction nets as simple as they look?



- And the net:



We should be able to apply the reduction. We *pull apart* the redex.

- The reduced net being:



We *fuse* the unnecessary ports.

- Reduction process:

starting net $\xrightarrow{\text{pulling apart}}$ not a net $\xrightarrow{\text{substitution}}$ not a net $\xrightarrow{\text{port fusion}}$ final net

Approaches to compute port fusion

- Term syntax (Lafont, Fernández-Mackie): cells as functional symbol and wires as paired occurrences of variables.
Port fusion computed via an equivalence on variables.
implicit treatment.
- Calculus of interaction nets (Fernández-Mackie): machine for reducing nets with concrete rules handling the equivalence.
external treatment.
- We seek an explicit treatment of port fusion.

Extending de Bruijn indices

- λ -calculus: integers for occurrences + ad hoc techniques
- our proposition for interaction nets:
integers for ports + Girard's Geometry of Interaction
- Surprising concrete link between linear logic and interaction nets through technical issues.

Graph-Hypergraph presentation

- Usual structure to define the statics of nets over ports (Mazza, ...).
- **Graph** for wires + **directed hypergraph** for cells + disjointness property



- Problems:
 - Huge class of structures for very few nets.
 - No graph-theoretic way to compute port-fusion.

Towards actions

- Shift of perspective : group actions (i.e. permutations) on ports.
- Natural disjointness of orbits.
- Action induced by wires: unique neighbour along wires \Rightarrow order 2 action.
- Action induced by cells: get next port around a cell.
- Graph-Hypergraph restricted to representations of actions.

Our proposition for statics

- Actions over \mathbb{N} : partial permutations.
- σ labelled by L when $\exists l : \text{Orbs}(\sigma) \mapsto L$.
- σ has pointed orbits when labelled by $\text{dom}(\sigma)$ with $\forall o \in \text{Orbs}(\sigma), l(o) \in o$.
- Countable set of symbols \mathcal{S} with arity function α .

Definition

An interaction net is an ordered pair $R = (\sigma_w, \sigma_c)$ where:

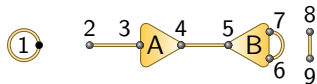
- σ_w w -permutation (order 2 partial permutation), we set $\text{dom}(\sigma_w) = P(R) \uplus \text{fix}(\sigma_w)$.
- σ_c partial permutation s.t. $\text{dom}(\sigma_c) \subseteq P(R)$, with pointed orbits and labelled by \mathcal{S} in such a way that $\forall o \in \text{Orbs}(\sigma_c), |o| = \alpha(l(o))$.

Example

- $R = (\sigma_w, \sigma_c)$ with

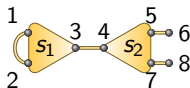
$$\sigma_w = (1)(2\ 3)(4\ 5)(6\ 7)(8\ 9) \text{ and } \sigma_c = (\overset{\bullet}{4}\ 3)_A(\overset{\bullet}{5}\ 6\ 7)_B$$

-



Port fusion by composition

- Our original net $R = (\sigma_w, \sigma_c)$:



- Right part of the rule $R' = (\tau_w, \tau_c)$



- Correspondance f from $\text{dom}(\sigma_w)$ to $\text{dom}(\tau_w)$.
- Computing the resulting action ρ_w :



$$\rho_w(6) = (\sigma_w f^{-1} \tau_w f \sigma_w f^{-1} \tau_w f \sigma_w)(6) = ((\sigma_w + \tau_w)[(f + f^{-1})(\sigma_w + \tau_w)]^4)(6)$$

A formula seems to appear.

The Execution formula for pinjs

- Originally introduced by Girard with operators, here we work with partial injections of integers (pinjs).
- $f : A \uplus B \rightarrow C \uplus D$ and $g : D \rightarrow B$ are pinjs, we set

$$\text{Ex}(f, g) = \sum_{i=0}^{\infty} [f(gf)^i] \upharpoonright_C^A$$

where $f \upharpoonright_C^A$ graph is f graph $\cap A \times C$.

- This is well-defined and bounded when $\text{dom}(f)$ is finite.

Property

$$\text{Ex}(\text{Ex}(f, g), h) = \text{Ex}(f, g + h) = \text{Ex}(\text{Ex}(f, h), g).$$

- Girard's usual interpretation: localized Church-Rosser.
- Strong confluence should be a corollary.

Definition

Let σ and τ be disjoint w -permutations and let f be a partial injection with $\text{dom}(f) \subseteq \text{dom}(\sigma)$ and $\text{codom}(f) \subseteq \text{dom}(\tau)$.

We call the Ex_0 -composition of σ and τ along f the partial permutation

$$\sigma \overset{f}{\rightsquigarrow} \tau = \text{Ex}(\sigma + \tau, f + f^{-1})$$

- In the previous example we get $\rho_w = \sigma_w \overset{f}{\rightsquigarrow} \tau_w$.
- ... but something is missing:



$\sigma_w \overset{f}{\rightsquigarrow} \tau_w = 0$, we can't see loops!

- First solution: this is not a bug, this is a feature.
- Second solution: we can detect them and define an extension $\sigma_w \overset{f}{\rightsquigarrow} \tau_w$.

Definition

Let $R = (\sigma_w, \sigma_c)$ and $R' = (\sigma'_w, \sigma'_c)$ be two **disjoint** interaction nets, and f be a pinj from $P_f(R)$ to $P_f(R')$.

We call *gluing of R and R' along f* the net $R \overset{f}{\leftarrow\rightarrow} R' = (\sigma_w \overset{f}{\leftarrow\rightarrow} \sigma'_w, \sigma_c + \sigma'_c)$.

Definition

Let R be a net, we call *cutting of R* a triple (R_1, f, R_2) such that $R = R_1 \overset{f}{\leftarrow\rightarrow} R_2$. Any net R' appearing in a cutting of R is called a *subnet of R* , noted $R' \subseteq R$.

- Gluing computes *port fusion*.
- Cutting *pulls apart* a subnet.

Property (Associativity of gluing)

$$R \overset{f+g}{\rightsquigarrow} (S \overset{h}{\rightsquigarrow} T) = (R \overset{f}{\rightsquigarrow} S) \overset{g+h}{\rightsquigarrow} T = (R \overset{g}{\rightsquigarrow} T) \overset{f+h}{\rightsquigarrow} S.$$

Corollary

If $R_0 = R \overset{f}{\rightsquigarrow} (S \overset{g}{\rightsquigarrow} T)$ then $\exists f_S, f_T$ s.t. $R_0 = (R \overset{f_S}{\rightsquigarrow} S) \overset{g+f_T}{\rightsquigarrow} T$.

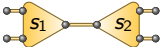
Property

\subseteq is an ordering of interaction nets.

Contexts and interfaces

- Interfaces: linearly ordered subset of free ports.
- Contexts: R^I with I an interface of R .
- Implicit gluing function between two interfaces of same length: $\rho(I, I')$.
- Context gluing: $R^I \rightsquigarrow R^{I'} = R \overset{\rho(I, I')}{\rightsquigarrow} R'$.

Reduction


- Rule for (s_1, s_2) : $\mathcal{R} = (R_r^{I_r}, R_p^{I_p})$ with
 - I_r (resp. I_p) contains all the free ports of R_r (resp. R_p)
 - $|I_r| = |I_p|$
 - R_r has the representation 

- Reduction: $R^I \rightsquigarrow \alpha(R_r)^{\alpha(I_r)} \xrightarrow{\mathcal{R}} R^I \rightsquigarrow \beta(R_p)^{\beta(I_p)}$
with α, β renamings

Theorem

We have strong confluence up to renaming.

Proof.

(involved) corollary of associativity of Ex. 

- Concrete bridge from MLL proofnets to interaction nets thanks to a quotient: Ex-collapse.
- Concrete implementation: a tool for handling interaction nets based on a kernel using this presentation.
- Algebraic generalization through DPO: morphism of interaction nets, existence of a double-pushout for a large class of rules.
- Same framework for multi-wires, multi-ports,