Differential Nets with Promotion

Introduction and current results

Paolo Tranquilli

paolo.tranquilli@ens-lyon.fr

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon

23 October 2009

Outline

1. The System
   - Why Differential?
   - The Nets

2. The Results
   - Confluence and termination
   - Curry-Howard correspondence

Outline

1. The System
   - Why Differential?
   - The Nets
2. The Results
   - Confluence and termination
   - Curry-Howard correspondence

Differentials: from semantics to syntax

Thomas Ehrhard.
On Köthe sequence spaces and linear logic.
Mathematical. Structures in Comp. Sci., 12:579–623, 2002.

Thomas Ehrhard.
Finiteness spaces.
Mathematical. Structures in Comp. Sci., 15(4):615–646, 2005.

$\Downarrow \quad \Downarrow \quad \Downarrow$

Thomas Ehrhard and Laurent Regnier.
The differential lambda-calculus.
Theor. Comput. Sci., 309(1):1–41, 2003.

Thomas Ehrhard and Laurent Regnier.
Differential interaction nets. (promotion free!)
Theor. Comput. Sci., 364(2):166–195, 2006.

Differentials: from semantics to syntax

Thomas Ehrhard.
On Köthe sequence spaces and linear logic.
Mathematical. Structures in Comp. Sci., 12:579–623, 2002.

Thomas Ehrhard.
Finiteness spaces.
Mathematical. Structures in Comp. Sci., 15(4):615–646, 2005.

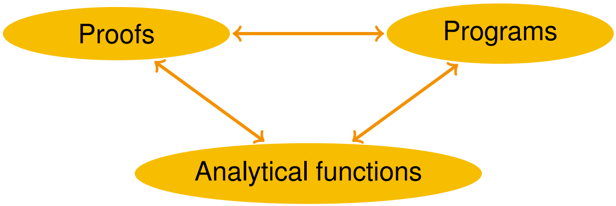
$\Downarrow \quad \Downarrow \quad \Downarrow$

Thomas Ehrhard and Laurent Regnier.
The differential lambda-calculus.
Theor. Comput. Sci., 309(1):1–41, 2003.

Thomas Ehrhard and Laurent Regnier.
Differential interaction nets. (promotion free!)
Theor. Comput. Sci., 364(2):166–195, 2006.

A third actor for Curry-Howard?

- Analysis may be regarded as a third pole for Curry-Howard:

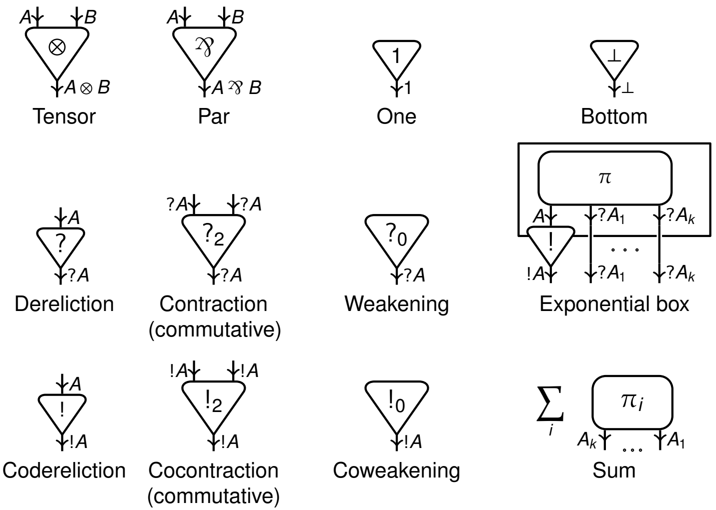


- For example, linearity: programs using inputs just once, proofs using hypotheses just once, linear functions.
- We concentrate on proofs, via their representation as nets.

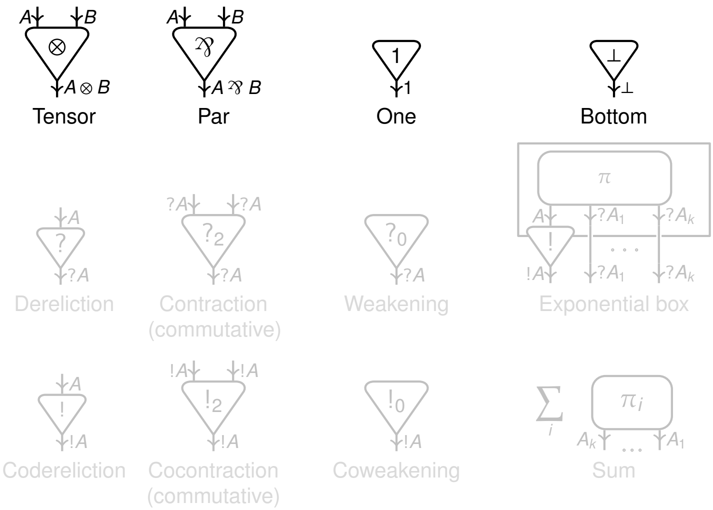
Outline

1. The System
   - Why Differential?
   - The Nets
2. The Results
   - Confluence and termination
   - Curry-Howard correspondence

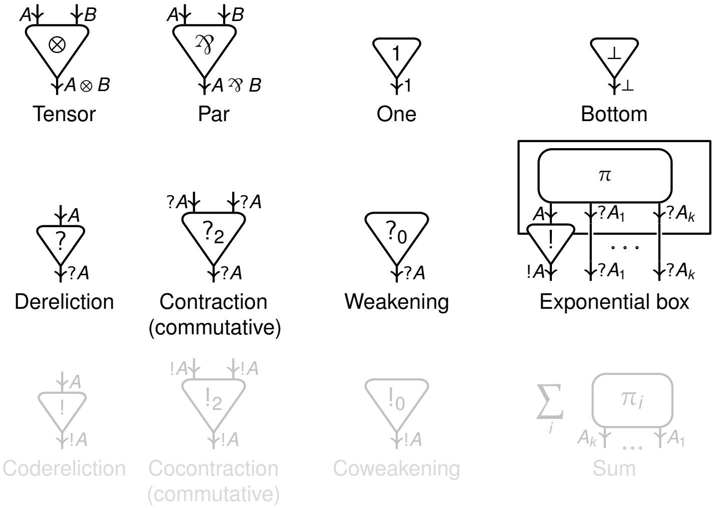
The nets: Family picture



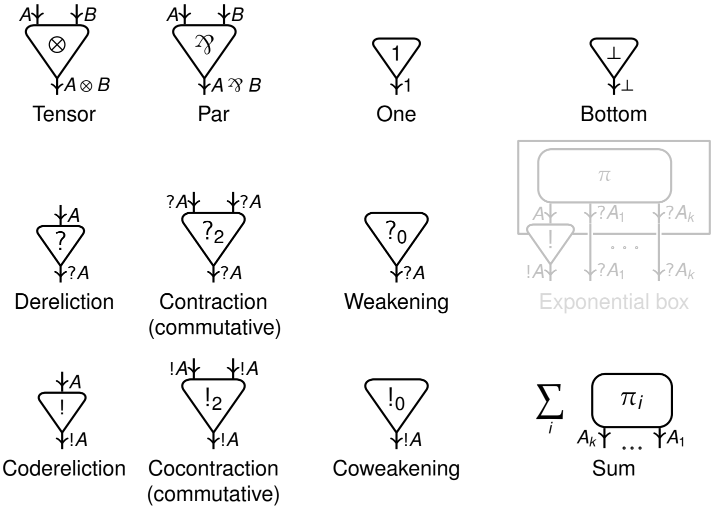
The nets: MLL


Tensor
Par
One
Bottom
Dereliction
Contraction (commutative)
Weakening
Exponential box
Codereliction
Cocontraction (commutative)
Coweakening
Sum

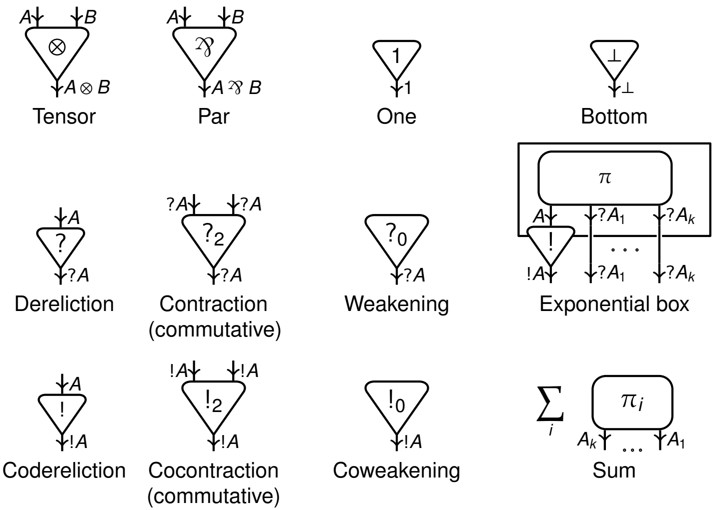
The nets: MELL



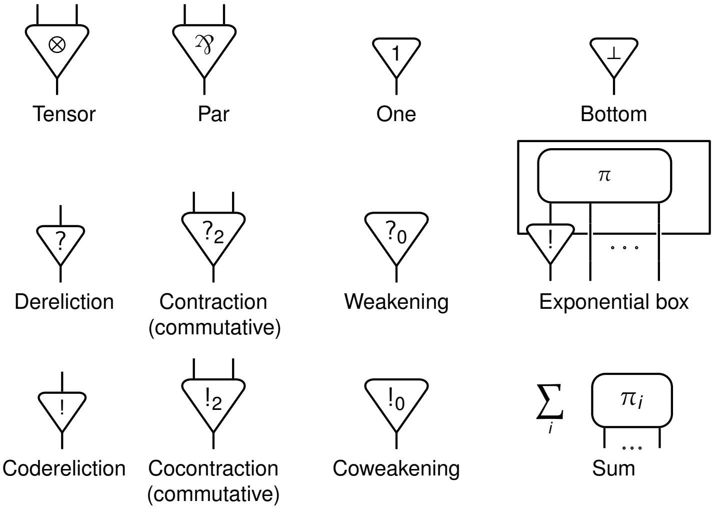
The nets: Differential Interaction Nets



The nets: Differential Nets (DiLL)



The nets: Differential Nets (DiLL)



Codereliction as derivative

Q: What is the derivative of a function in analysis?

A: Its best linear aproximation.

$$\Downarrow \qquad \Downarrow \qquad \Downarrow$$

Q: What is the derivative of a proof/program?

A: Its best linear aproximation,
i.e. the "closest" proof/program that uses its hypothesis/input exactly once.

$$\left.\frac{\partial f(x)}{\partial x}\right|_{x=0} \cdot u \rightsquigarrow$$

Codereliction as derivative

Q: What is the derivative of a function in analysis?

A: Its best linear aproximation.

$$\Downarrow \qquad \Downarrow \qquad \Downarrow$$

Q: What is the derivative of a proof/program?

A: Its best linear aproximation,
i.e. the “closest” proof/program that uses its hypothesis/input exactly once.

$$\left.\frac{\partial f(x)}{\partial x}\right|_{x=0} \cdot u \rightsquigarrow$$

Codereliction as derivative

Q: What is the derivative of a function in analysis?

A: Its best linear aproximation.

$\Downarrow \quad \Downarrow \quad \Downarrow$

Q: What is the derivative of a proof/program?

A: Its best linear aproximation,
i.e. the “closest” proof/program that uses its hypothesis/input exactly once.

$$\left.\frac{\partial f(x)}{\partial x}\right|_{x=0} \cdot u \rightsquigarrow$$

Codereliction as derivative

Q: What is the derivative of a function in analysis?

A: Its best linear aproximation.

$\Downarrow \quad \Downarrow \quad \Downarrow$

Q: What is the derivative of a proof/program?

A: Its best linear aproximation,
i.e. the “closest” proof/program that uses its hypothesis/input exactly once.

$$\left.\frac{\partial f(x)}{\partial x}\right|_{x=0} \cdot u \leadsto$$

Codereliction as derivative

Q: What is the derivative of a function in analysis?

A: Its best linear aproximation.

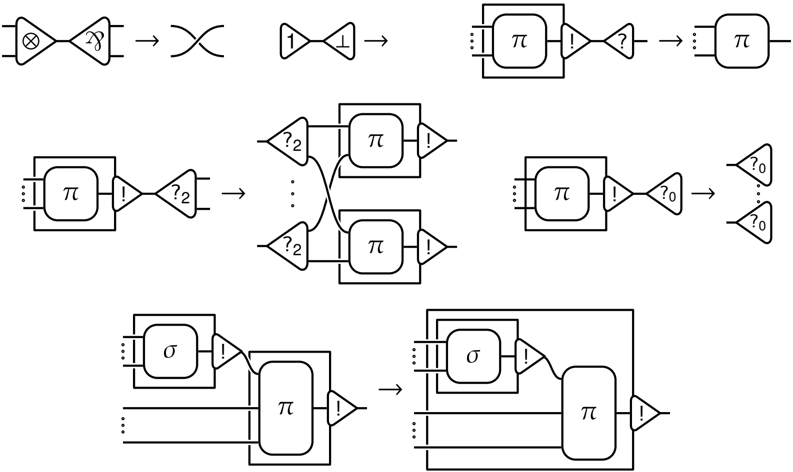
$\Downarrow \quad \Downarrow \quad \Downarrow$

Q: What is the derivative of a proof/program?

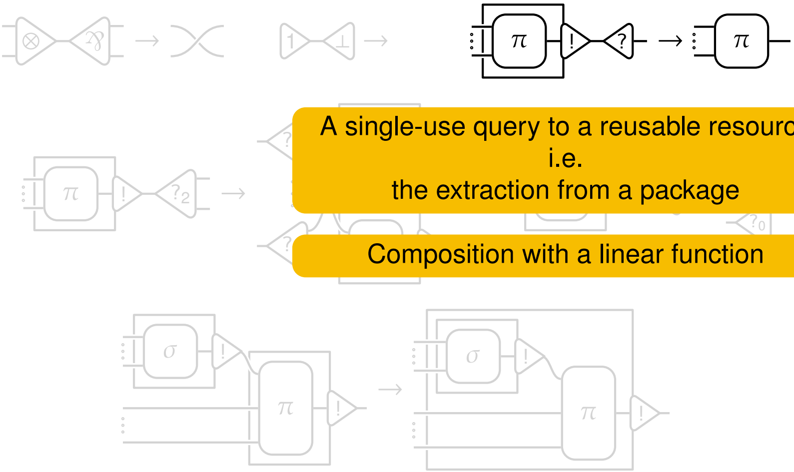
A: Its best linear aproximation,
i.e. the "closest" proof/program that uses its
hypothesis/input exactly once.

$$\left.\frac{\partial f(x)}{\partial x}\right|_{x=0} \cdot u \rightsquigarrow$$

The known reductions



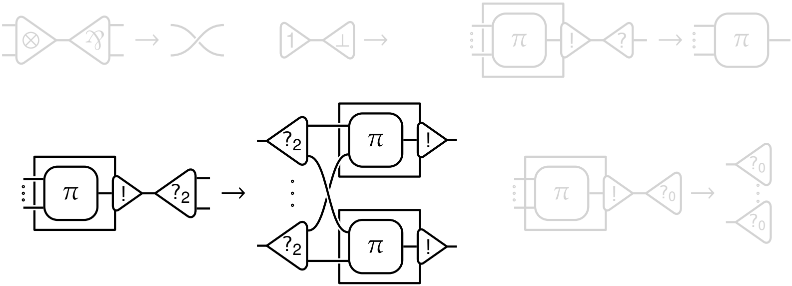
The known reductions



A single-use query to a reusable resource
i.e.
the extraction from a package

Composition with a linear function

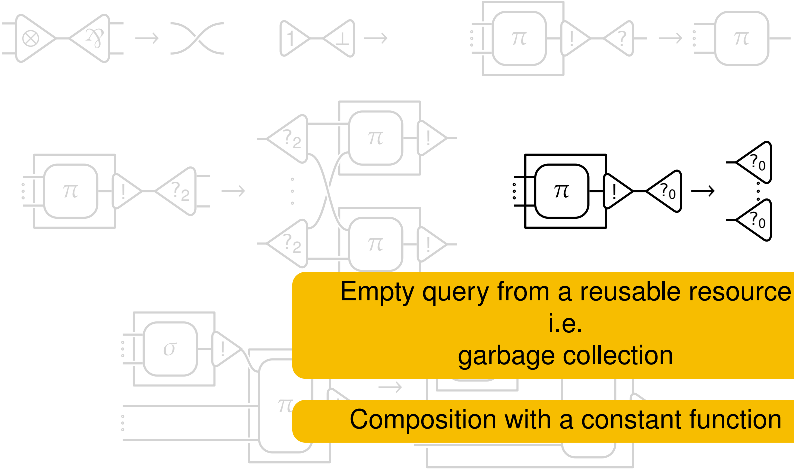
The known reductions



Two queries from a reusable resource
i.e.
the duplication of a package

Composition on a shared variable

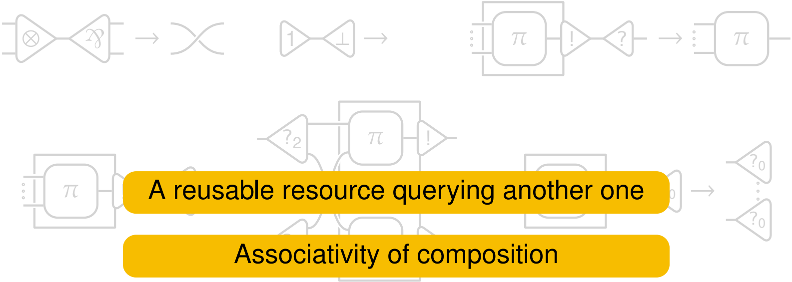
The known reductions



Empty query from a reusable resource
i.e.
garbage collection

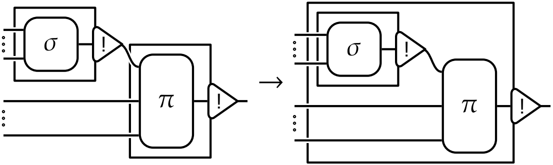
Composition with a constant function

The known reductions

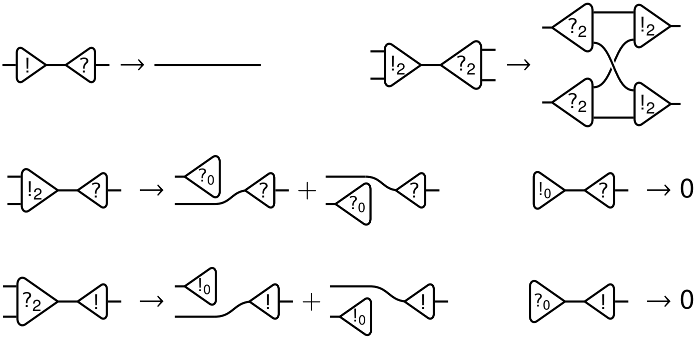


A reusable resource querying another one

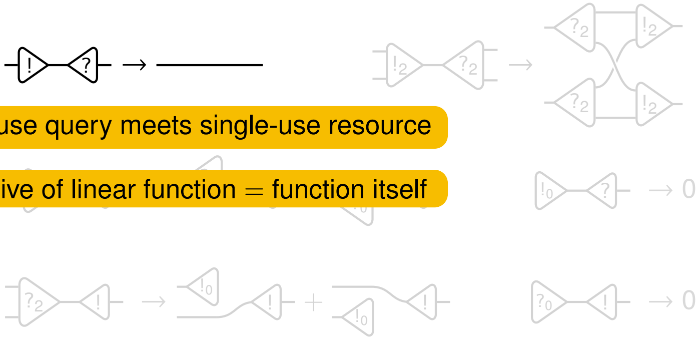
Associativity of composition



The new reductions – Differential Interaction Nets



The new reductions – Differential Interaction Nets



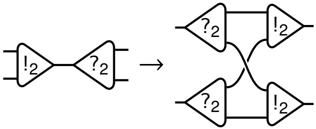
single-use query meets single-use resource

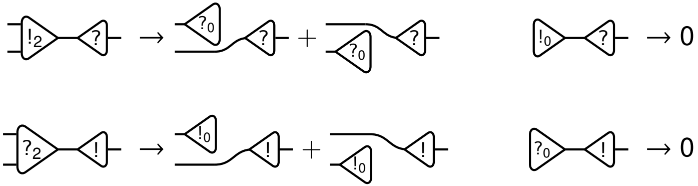
derivative of linear function $=$ function itself

The new reductions – Differential Interaction Nets

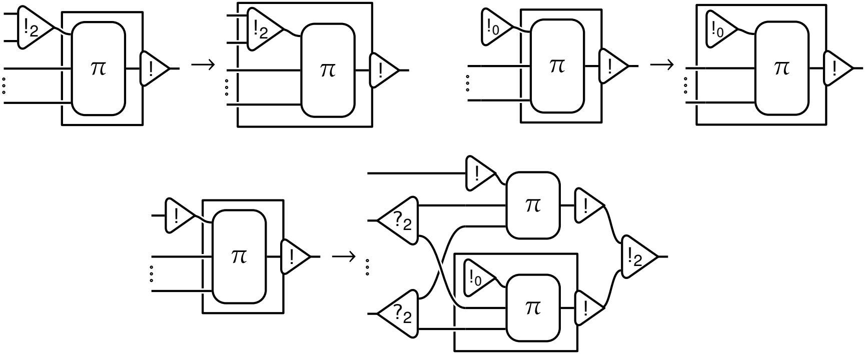
routing!

properties of sharing and sums
linearity
derivative of a shared variable

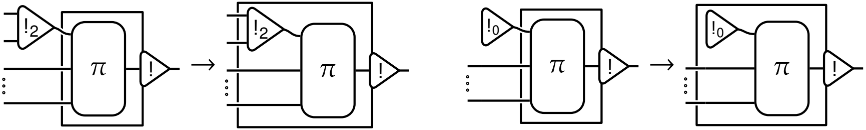




The new reductions – Enter promotion

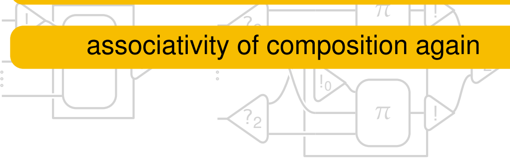


The new reductions – Enter promotion



commutation of routing and packaging

associativity of composition again

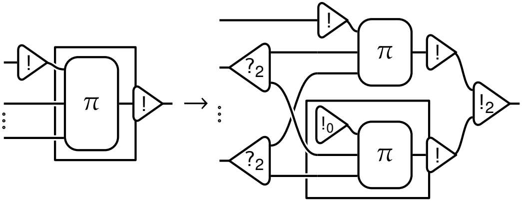


The new reductions – Enter promotion

reusable resource asks for a single-use one

the chain rule

$$\left.\frac{\partial f(g(x))}{\partial x}\right|_{x=0} = \left.\frac{\partial f(y)}{\partial y}\right|_{y\quad g(0)} \cdot \left.\frac{\partial g(x)}{\partial x}\right|_{x=0}$$



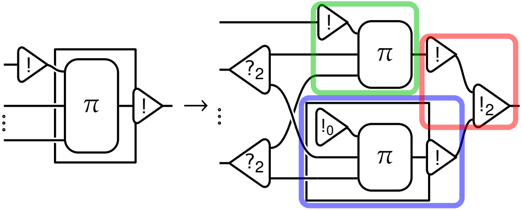
DiLL's very own pandora's box

The new reductions – Enter promotion

reusable resource asks for a single-use one

the chain rule

$$\left.\frac{\partial f(g(x))}{\partial x}\right|_{x=0} = \boxed{\left.\frac{\partial f(y)}{\partial y}\right|}_{y=\boxed{g(0)}} \cdot \boxed{\left.\frac{\partial g(x)}{\partial x}\right|_{x=0}}$$



DiLL's very own pandora's box

A note on reductions and sums

$$\pi \to \quad \pi_1$$

- In LL reduction is deterministic.
- In DiLL reduction is nondeterministic.
- Reduction rules extended by context closure as usual.
- Sums duplicate the context, without spreading outside boxes.
- As to reducing sums, two flavours
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \lambda_2 + \cdots + \lambda_k$;
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \cdots + \mu_h + \lambda_{h+1} + \cdots + \lambda_k$, with $h \geq 1$.

A note on reductions and sums

$$\pi \;\to\; \pi_1 + \cdots + \pi_k$$

- In LL reduction is deterministic.
- In DiLL reduction is nondeterministic.
- Reduction rules extended by context closure as usual.
- Sums duplicate the context, without spreading outside boxes.
- As to reducing sums, two flavours
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \lambda_2 + \cdots + \lambda_k$;
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \cdots + \mu_h + \lambda_{h+1} + \cdots + \lambda_k$, with $h \geq 1$.

A note on reductions and sums

$$\omega[\ \pi] \to \omega[\ \pi_1\ + \cdots +\ \pi_k\ ]$$

- In LL reduction is deterministic.
- In DiLL reduction is nondeterministic.
- Reduction rules extended by context closure as usual.
- Sums duplicate the context, without spreading outside boxes.
- As to reducing sums, two flavours
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \lambda_2 + \cdots + \lambda_k$;
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \cdots + \mu_h + \lambda_{h+1} + \cdots + \lambda_k$, with $h \geq 1$.

A note on reductions and sums

$$\omega[\ \pi] \to \omega[\ \ \pi_1] + \cdots + \omega[\pi_k\ \ ]$$

with π not in a box.

- In LL reduction is deterministic.
- In DiLL reduction is nondeterministic.
- Reduction rules extended by context closure as usual.
- Sums duplicate the context, without spreading outside boxes.
- As to reducing sums, two flavours
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \lambda_2 + \cdots + \lambda_k$;
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \cdots + \mu_h + \lambda_{h+1} + \cdots + \lambda_k$, with $h \geq 1$.

A note on reductions and sums

$$\omega[!\pi] \to \omega[!(\pi_1 \; + \cdots + \quad \pi_k)]$$

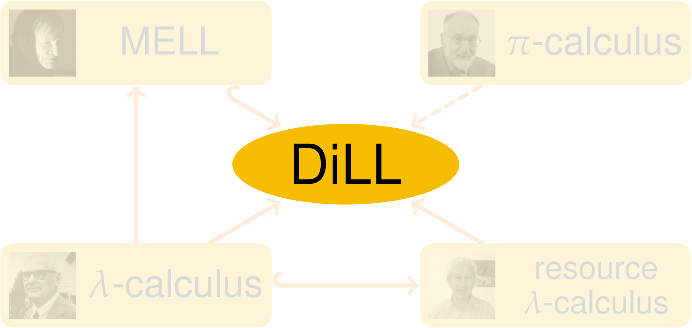
- In LL reduction is deterministic.
- In DiLL reduction is nondeterministic.
- Reduction rules extended by context closure as usual.
- Sums duplicate the context, without spreading outside boxes.
- As to reducing sums, two flavours
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \lambda_2 + \cdots + \lambda_k$;
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \cdots + \mu_h + \lambda_{h+1} + \cdots + \lambda_k$, with $h \geq 1$.

A note on reductions and sums

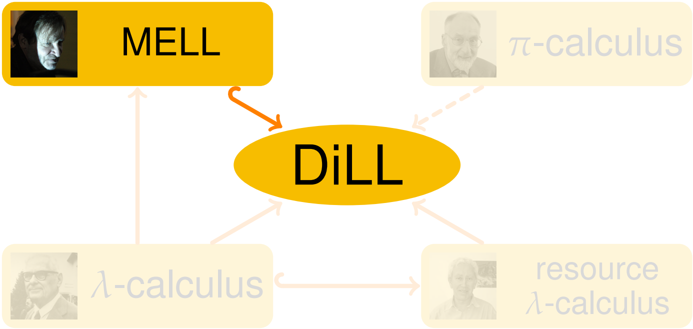
$$\omega[!\pi] \to \omega[!(\pi_1 + \cdots + \pi_k)]$$

- In LL reduction is deterministic.
- In DiLL reduction is nondeterministic.
- Reduction rules extended by context closure as usual.
- Sums duplicate the context, without spreading outside boxes.
- As to reducing sums, two flavours
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \lambda_2 + \cdots + \lambda_k$;
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \cdots + \mu_h + \lambda_{h+1} + \cdots + \lambda_k$, with $h \geq 1$.

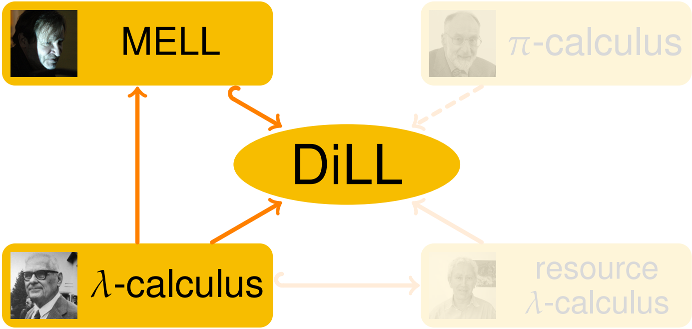
Quite an expressive system



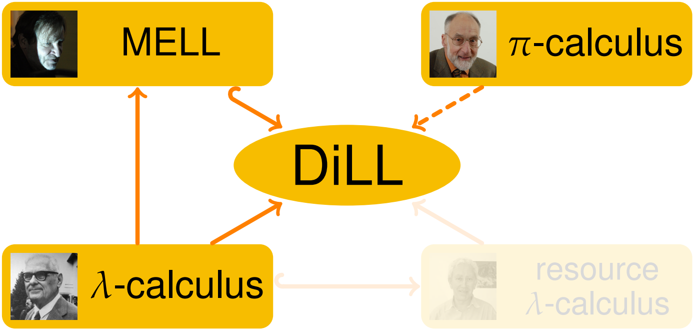
Quite an expressive system



Quite an expressive system

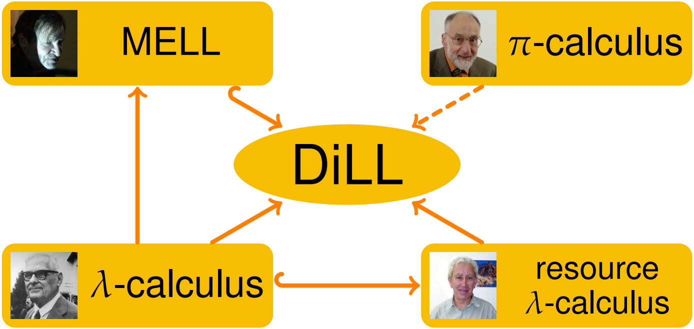


Quite an expressive system



Thomas Ehrhard and Olivier Laurent.
Interpreting a finitary pi-calculus in differential interaction nets.
CONCUR, *LNCS* vol. 4703, pages 333–348, 2007.

Quite an expressive system



Gérard Boudol.
The lambda-calculus with multiplicities.
INRIA Research Report 2025, 1993.

P. T.
Intuitionistic differential nets and lambda calculus.
To appear in *Theoretical Computer Science*, 2008.

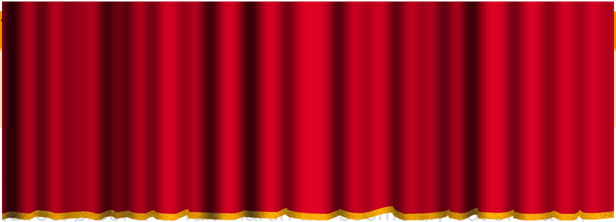
Outline

1. The System
   - Why Differential?
   - The Nets
2. The Results
   - Confluence and termination
   - Curry-Howard correspondence

What do we want?

Theorem

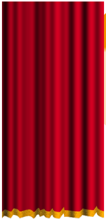
Reduction is confluent.



What do we want?

Theorem

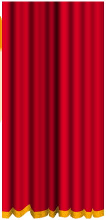
Reduction is confluent.



Theorem (Finite developments)

Reduction not reducing new redexes is strongly normalizing.

Local confluence of developments $\Rightarrow$ strongly confluent parallel reduction.
(direct proof à la Tait-Martin Löf seemingly out of reach)



What do we want?

Theorem

Reduction of switching acyclic untyped LL nets is confluent.

Theorem (Finite developments)

Reduction not reducing new redexes is strongly normalizing.

Local confluence of developments $\Rightarrow$ strongly confluent parallel reduction.
(direct proof à la Tait-Martin Löf seemingly out of reach)

Michele Pagani and Lorenzo Tortora de Falco.
Strong normalization property for second order linear logic.
To appear on *Theoretical Computer Science*, 2008.

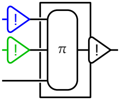
Nondeterminism and confluence?

Q: How come we speak of confluence with nondeterminism around?

A: confluence ensures nondeterministic choice is internal, i.e. it is not triggered by what we reduce.

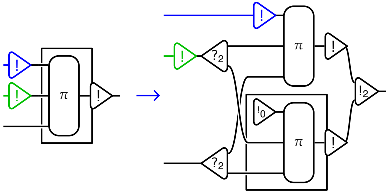
Reducing two different redexes gives the same set of nondeterministic choices in the end.

The need for associativity and neutrality



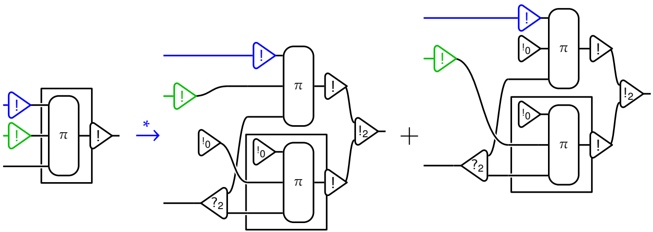
- Contractions and cocontractions need to be swapped to have confluence (associativity).
- Other confluence diagrams require merging (co)weakenings with (co)contractions (neutrality).

The need for associativity and neutrality



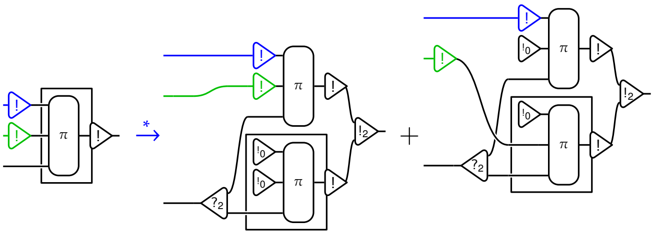
- Contractions and cocontractions need to be swapped to have confluence (associativity).
- Other confluence diagrams require merging (co)weakenings with (co)contractions (neutrality).

The need for associativity and neutrality



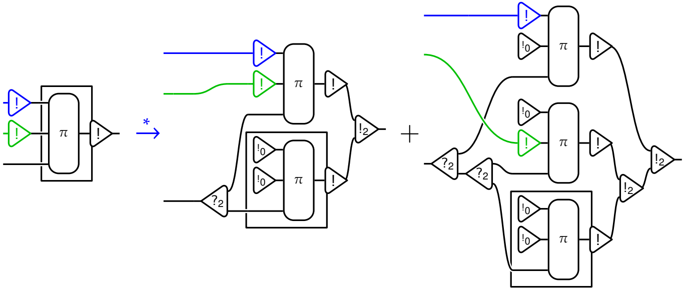
- Contractions and cocontractions need to be swapped to have confluence (associativity).
- Other confluence diagrams require merging (co)weakenings with (co)contractions (neutrality).

The need for associativity and neutrality



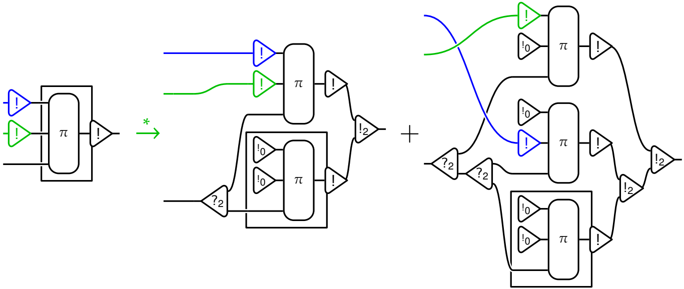
- Contractions and cocontractions need to be swapped to have confluence (associativity).
- Other confluence diagrams require merging (co)weakenings with (co)contractions (neutrality).

The need for associativity and neutrality



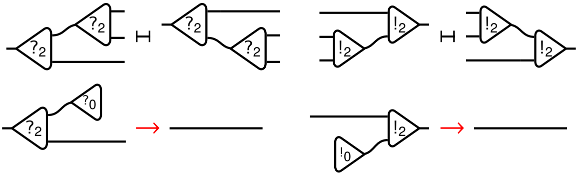
- Contractions and cocontractions need to be swapped to have confluence (associativity).
- Other confluence diagrams require merging (co)weakenings with (co)contractions (neutrality).

The need for associativity and neutrality



- Contractions and cocontractions need to be swapped to have confluence (associativity).
- Other confluence diagrams require merging (co)weakenings with (co)contractions (neutrality).

Associative equivalence and neutral reduction

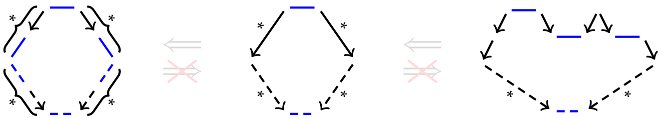


- associative equivalence $\sim\ =\ \mapsfrom^*$;
- neutral reduction (if reversed arbitrary (co)contraction trees can be generated).

Compulsory!

Reduction modulo equivalence

- Confluence properties in presence of an equivalence relation $\sim$:



confluence of $\to / \sim \, = \, \sim\to\sim$

confluence modulo

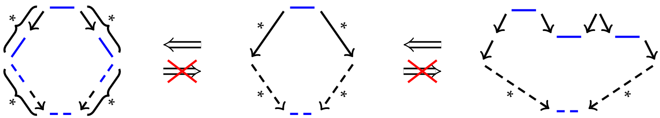
Church-Rosser modulo (CR$\sim$)
$(\to \cup \leftarrow \cup \sim)^* \subseteq \xrightarrow{*}\sim\xleftarrow{*}$

- Strong normalization modulo (SN$\sim$) is defined by SN of $\to/\sim \, = \, \sim\to\sim$.

Terese.
Term Rewriting Systems, volume 55 of *Cambridge Tracts in TCS*.
Cambridge University Press, 2003.

Reduction modulo equivalence

- Confluence properties in presence of an equivalence relation $\sim$:



confluence of $\to / \sim \; = \; \sim\to\sim$

confluence modulo

Church-Rosser modulo (CR$\sim$)
$(\to \cup \leftarrow \cup \sim)^* \subseteq \xrightarrow{*}\sim\xleftarrow{*}$

- Strong normalization modulo (SN$\sim$) is defined by SN of $\to/\sim \; = \; \sim\to\sim$.

Terese.
Term Rewriting Systems, volume 55 of *Cambridge Tracts in TCS*.
Cambridge University Press, 2003.

What do we want?

Theorem

Reduction of switching acyclic untyped LL nets is confluent.

Theorem (Finite developments)

Reduction not reducing new redexes is strongly normalizing.

Local confluence of developments $\Rightarrow$ strongly confluent parallel reduction.
(direct proof à la Tait-Martin Löf seemingly out of reach)

Michele Pagani and Lorenzo Tortora de Falco.
Strong normalization property for second order linear logic.
To appear on *Theoretical Computer Science*, 2008.

What do we want? We have it!

Theorem

Reduction of switching acyclic untyped DiLL nets is CR modulo ~.

Theorem (Finite developments)

Reduction not reducing new redexes is strongly normalizing modulo ~.

Local confluence of developments ⇒ strongly CR modulo ~ parallel reduction.
(direct proof à la Tait-Martin Löf seemingly out of reach)

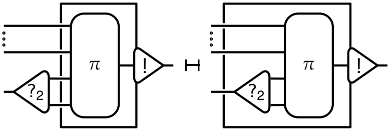
P. T.
Confluence of pure differential nets with promotion.
CSL'09, volume 5771 of *LNCS*, pages 500–514. 2009.

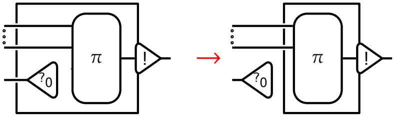
details

While we are at it. . .

- As we need to consider reduction modulo, why not add other optional (yet useful) equivalences?
- We introuduce push and bang sum equivalences, together with pull and bang zero reductions.
- Again reductions cannot be reversed to prevent deranged behaviour.
- Optional, but must be taken together to have CR$\sim$.

Push equivalence and pull reduction – the rules

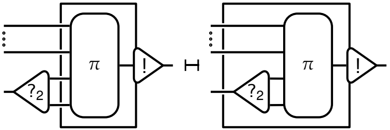


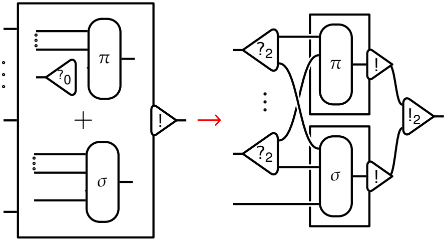


with $\pi \neq 0$.

studied in LL relating to λ-calculi with explicit subsitutions.

Push equivalence and pull reduction – the rules

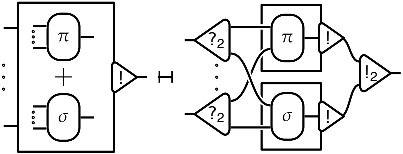




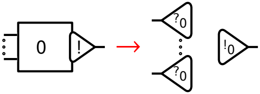
with $\pi \neq 0$.

studied in LL relating to λ-calculi with explicit subsitutions.

Bang sum equivalence and bang zero reduction



with $\sigma, \pi \neq 0$,



- Derived from LL's exponential isomorphism $!A \otimes !B \cong !(A \,\&\, B)$ (and $1 \cong !\top$).
- It means we can (if we want) work without sums in boxes.

What do we want?

Theorem

Reduction in the typed case is strongly normalizing.

What do we want?

Theorem

Reduction in the typed case is strongly normalizing.

Theorem (“conservation” according to Barendregt)

Non erasing reduction is perpetual,
i.e. it never terminates on non SN terms.

$\neg e$ stands for non erasing reduction.

Michele Pagani and Lorenzo Tortora de Falco.
Strong normalization property for second order linear logic.
To appear on *Theor. Comput. Sci.*, 2008.

What do we want?

Theorem

Reduction in the typed case is strongly normalizing.

Theorem ("standardization" according to Girard)

Non erasing reduction is perpetual,
i.e. it never terminates on non SN terms.

$\neg e$ stands for non erasing reduction.

Michele Pagani and Lorenzo Tortora de Falco.
Strong normalization property for second order linear logic.
To appear on *Theor. Comput. Sci.*, 2008.

What do we want?

Theorem

Reduction in the typed case is strongly normalizing.

Theorem ("striction" according to Danos)

Non erasing reduction is perpetual,
i.e. it never terminates on non SN terms.

$\neg e$ stands for non erasing reduction.

Michele Pagani and Lorenzo Tortora de Falco.
Strong normalization property for second order linear logic.
To appear on *Theor. Comput. Sci.*, 2008.

What do we want?

Theorem

Reduction 2nd order LL is strongly normalizing.

Theorem (cons.|stand.|strict.)

For π untyped switching acyclic LL net,
$\pi \in \mathrm{SN} \iff \pi \in \mathrm{WN}_{\neg e}$.

$\neg e$ stands for non erasing reduction.

Michele Pagani and Lorenzo Tortora de Falco.
Strong normalization property for second order linear logic.
To appear on *Theor. Comput. Sci.*, 2008.

What do we want? We have it!

Theorem

Reduction simply typed DiLL is strongly normalizing.

Theorem (cons.|stand.|strict. with Pagani)

*For π untyped * switching acyclic DiLL net,*
$\pi \in \mathrm{SN} \iff \pi \in \mathrm{WN}_{\neg e}$.

$\neg e$ stands for non erasing reduction.

Michele Pagani, P. T.
The conservation theorem for differential nets with promotion.
Manuscript in preparation, result contained in my thesis.

Michele Pagani.
The cut-elimination theorem for differential nets with promotion.
TLCA'09, volume 5608 of *LNCS*, pages 219–233. 2009.

* actually closely so but not completely, but we will just. . .

. . . sweep it under the carpet

(urban art by Banksy)

details

Intermezzo – Summing up the state of affairs

	CR~	FD	Cons.	Stand.	WN	SN
Untyped DiLL	[Tr09]	[Tr09]	[PaTr09]	[PaTr09]	No	No
Second order DiLL	$\Downarrow$ Yes	$\Downarrow$ Yes	$\Downarrow$ Yes	$\Downarrow$ Yes	**?**	**?**
Propositional DiLL	$\Downarrow$ Yes	$\Downarrow$ Yes	$\Downarrow$ Yes	$\Downarrow$ Yes	[Pa09]	[PaTr09]

FD: finite developments; Cons.: conservation;
Stand.: standardization (reduction can be ordered in ascending depth).

[Tr09] P. T.
Confluence of pure differential nets with promotion.
CSL'09, volume 5771 of *LNCS*, pages 500–514. 2009.

[PaTr09] Michele Pagani, P. T.
The conservation theorem for differential nets with promotion.
Manuscript in preparation, result contained in my thesis.

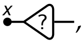
[Pag09] Michele Pagani.
The cut-elimination theorem for differential nets with promotion.
TLCA'09, volume 5608 of *LNCS*, pages 219–233. 2009.

Outline

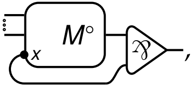
1. The System
   - Why Differential?
   - The Nets
2. The Results
   - Confluence and termination
   - Curry-Howard correspondence

LL proofnets and λ-calculus

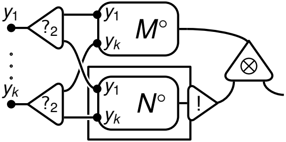
Curry-Howard $\rightsquigarrow$ translation of λ-terms into proof nets

$x^\circ :=$ 

,

$(\lambda x.M)^\circ :=$ 

,

$(MN)^\circ :=$ 

Theorem

$$M \xrightarrow{*} N \textit{ iff } M^\circ \xrightarrow{*} N^\circ.$$

Enter resource calculus

Q: What calculus corresponds in this way to differential nets?

A: Resource calculus.

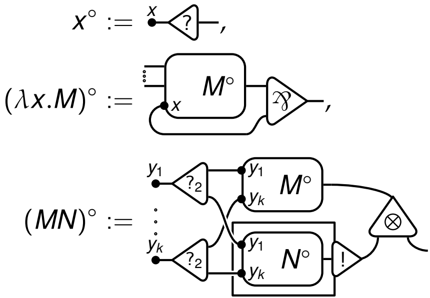
- arguments can be ephemeral, i.e. one-use, or perpetual, i.e. reusable, and mixed together.

$$M ::= x \mid \lambda x.M \mid MP, \qquad P ::= [M_1, \dots, M_h, N_1^!, \dots, N_k^!]$$

- wrt Boudol's original calculus, we have
  - non lazy reduction *à la* differential λ-calculus (with sums).
  - non-affinity over ephemeral resources (i.e. $(\lambda d.I)[N] \to 0$).

Differential nets and resource calculus

Translation of resource terms into differential nets

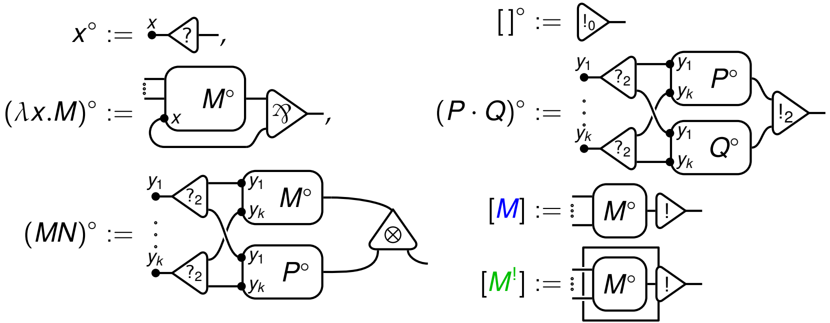


Theorem

$$M \xrightarrow{*} N \text{ iff } M^\circ \xrightarrow{*} N^\circ.$$

Differential nets and resource calculus

Translation of resource terms into differential nets



Theorem

$M \xrightarrow{*} N$ *iff* $M^\circ \xrightarrow{*} N^\circ$.

Consequences

- Confluence of differential nets $\Rightarrow$ confluence of resource calculus

P. T.
Intuitionistic differential nets and lambda-calculus.
To appear in *Theoretical Computer Science*, 2008.

- SN of typed differential nets $\Rightarrow$ SN of typed resource calculus
- More on resource calculus (and direct proofs of confluence, standardization, characterization of solvability)

Michele Pagani and P. T.
Parallel reduction in resource lambda-calculus.
To appear in APLAS'09, 2009.

Michele Pagani and Simona Ronchi della Rocca
Solvability in resource lambda-calculus.
Submitted for publication, 2009.

Concluding remarks

- Has that anything to do with complexity?
- The hope is that LL's ICC methods caould be somehow adapted to DiLL and from there to concurrent and nondeterministic computation.
- Also, the ephemeral resources introduced by codereliction "smell" of bounded complexity, is there more underneath?
- In any case, an affine version seems necessary: bounded resources are ok, exact ones are cumbersome.

Thanks

Questions?

Concluding remarks

- Has that anything to do with complexity?
- The hope is that LL's ICC methods caould be somehow adapted to DiLL and from there to concurrent and nondeterministic computation.
- Also, the ephemeral resources introduced by codereliction "smell" of bounded complexity, is there more underneath?
- In any case, an affine version seems necessary: bounded resources are ok, exact ones are cumbersome.

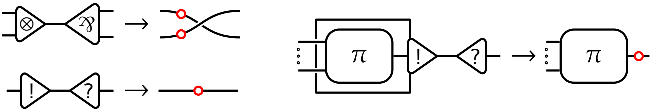
Thanks

Questions?

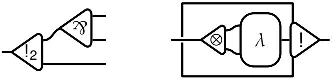
Stating the finite developments theorem

DiLL$^\circ$: the system blocking "new" redexes and exponential clashes.

- "New" redexes are blocked via redefining reductions:



- Exponential clashes are "type mismatches", as for example



(this defines an embedding π° from DiLL to DiLL$^\circ$.)

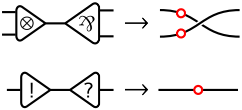
Theorem (Finite developments)

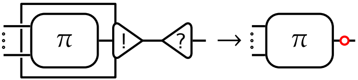
Untyped switching acyclic DiLL$^\circ$ *nets are SN$\sim$.*

Stating the finite developments theorem

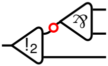
DiLL$^\circ$: the system blocking “new” redexes and exponential clashes.

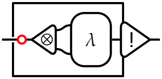
- “New” redexes are blocked via redefining reductions:





- Exponential clashes are “type mismatches”, as for example





(this defines an embedding π° from DiLL to DiLL$^\circ$.)

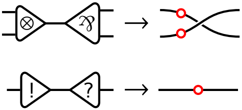
Theorem (Finite developments)

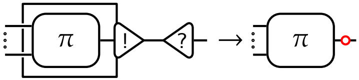
Untyped switching acyclic DiLL$^\circ$ *nets are SN$\sim$.*

Stating the finite developments theorem

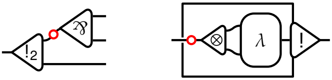
DiLL$^\circ$: the system blocking “new” redexes and exponential clashes.

- “New” redexes are blocked via redefining reductions:





- Exponential clashes are “type mismatches”, as for example



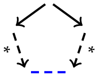
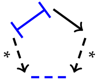
(this defines an embedding π° from DiLL to DiLL$^\circ$.)

Theorem (Finite developments)

Untyped switching acyclic DiLL$^\circ$ *nets are SN*$\sim$*.*

Proving CR$\sim$

1. We check in DiLL$^\circ$:

local confluence $\sim$ & local coherence $\sim$

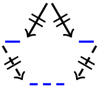
& SN$\sim$ $\Rightarrow$ DiLL$^\circ$ is CR$\sim$

(by Huet generalizing Newman's lemma)

2. In DiLL, define $\pi ⇻ \pi'$ if $\pi^\circ \xrightarrow{*} \pi'$ in DiLL$^\circ$ (forgetting $\circ$ in π').

$$\rightarrow \subseteq ⇻ \subseteq \xrightarrow{*} \Rightarrow ⇻^* = \xrightarrow{*} \quad \text{and} \quad (⇼ \cup \sim)^* = (\leftrightarrow \cup \sim)^*$$

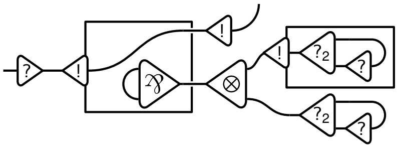
3. DiLL$^\circ$ CR$\sim$ $\Rightarrow$ $⇻$ strongly CR modulo



$\Rightarrow$ DiLL is CR$\sim$

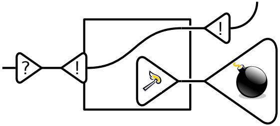
back

"Lazarus" clashes



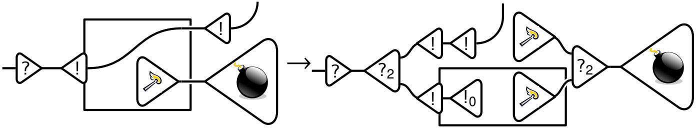
In LL Lazarus clashes do not negate standardization.

“Lazarus” clashes



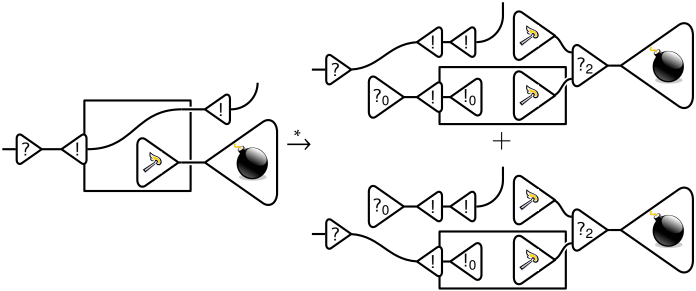
In LL Lazarus clashes do not negate standardization.

"Lazarus" clashes



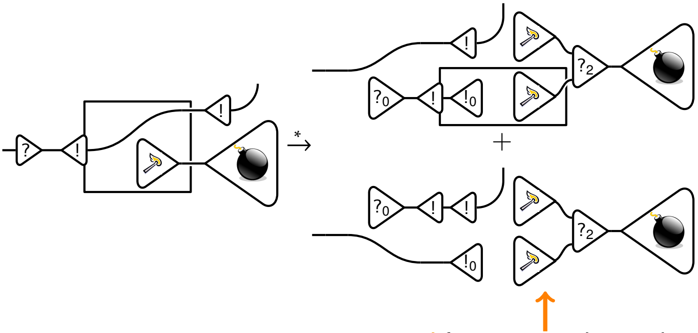
In LL Lazarus clashes do not negate standardization.

“Lazarus” clashes



In LL Lazarus clashes do not negate standardization.

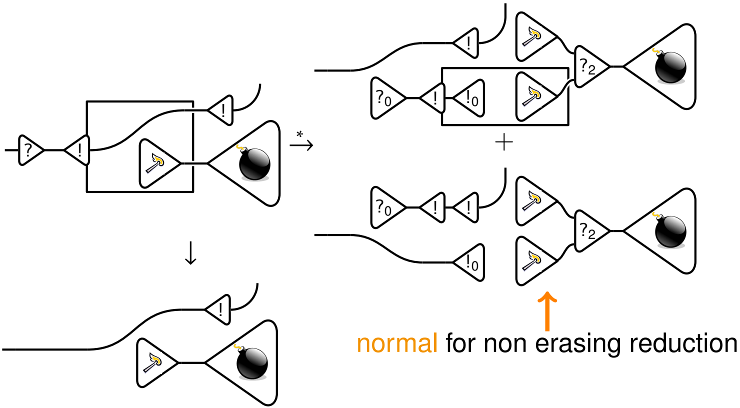
"Lazarus" clashes



normal for non erasing reduction

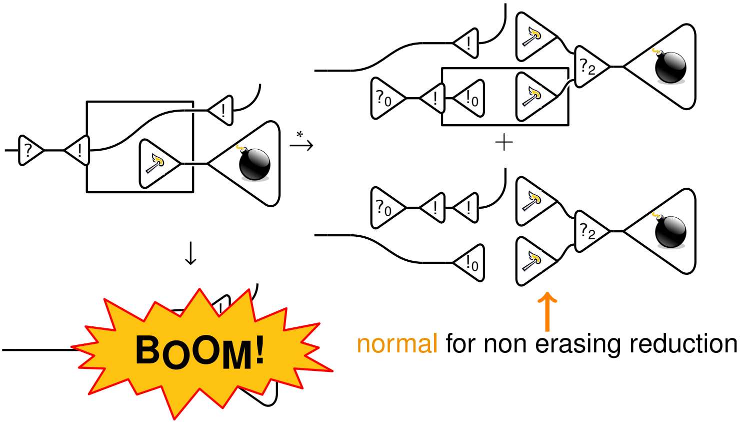
In LL Lazarus clashes do not negate standardization.

“Lazarus” clashes



In LL Lazarus clashes do not negate standardization.

"Lazarus" clashes

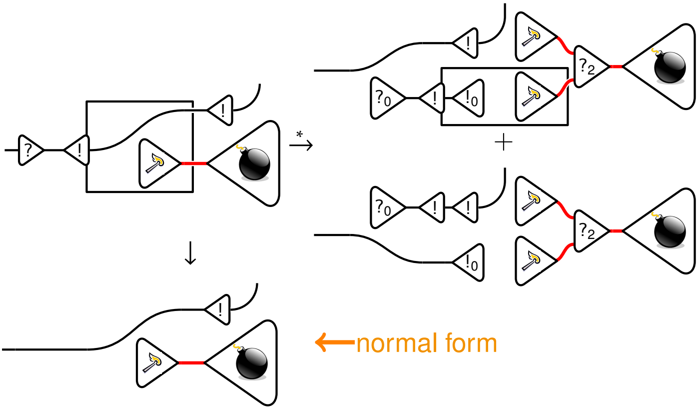


In LL Lazarus clashes do not negate standardization.

Lax typing

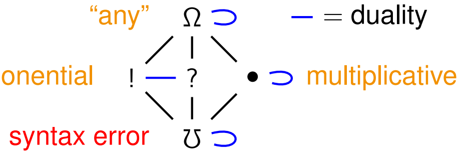
- We introuduce a very weak form of typing.
- It does not disallow nets, but assigns a special syntax error type to exponential clashes.
- We block the reduction of “syntax errror”-typed cuts.
- Any typing system avoiding clashes can be embedded in this one.

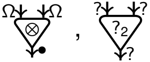
“Lazarus” clashes



Lax typing

- 5 ordered types:



- expected typing rules, for example 
- All type mismatches are allowed, but resulting type is the inf of the two.
- ℧-typed cuts are blocked.
- Types are preserved during reduction.

Lax typing does not disallow nets, but provides a mechanism to annotate exponential clashes and prevent them from resurrecting.

back