Confluence and normalization of differential nets with boxes

Paolo Tranquilli

ptranqui@pps.jussieu.fr

Preuves Programmes et Systèmes
Université Paris Diderot

Dipartimento di Matematica
Università Roma Tre

12 March 2009

Outline

Outline

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.

$\Downarrow$ $\Downarrow$ $\Downarrow$

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

Computer Science
- Programs
- Single use of resources
- Nondeterminism
- Depletable resources

Proof Theory
- Proofs
- No structural rules
- Costructural rules, sum
- Coderelicition

Analysis
- Analytical functions
- Linearity
- Sums
- Differentials: linear aproximation

A third actor for Curry-Howard

Computer Science
- Programs
- Single use of resources
- Nondeterminism
- Depletable resources

Proof Theory
- Proofs
- No structural rules
- Costructural rules, sum
- Codereliction

Analysis
- Analytical functions
- Linearity
- Sums
- Differentials: linear aproximation

A third actor for Curry-Howard

Computer Science
- Programs
- Single use of resources
- Nondeterminism
- Depletable resources

Proof Theory
- Proofs
- No structural rules
- Costructural rules, sum
- Codereliction

Analysis
- Analytical functions
- Linearity
- Sums
- Differentials: linear aproximation

A third actor for Curry-Howard

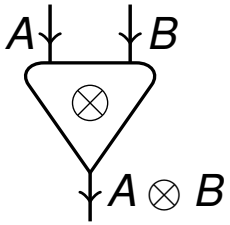
Computer Science
- Programs
- Single use of resources
- Nondeterminism
- Depletable resources

Proof Theory
- Proofs
- No structural rules
- Costructural rules, sum
- Codereliction

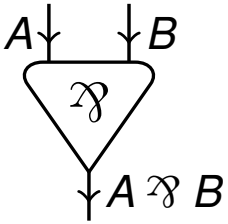
Analysis
- Analytical functions
- Linearity
- Sums
- Differentials: linear aproximation

Outline

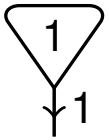
The nets: Family picture



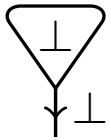
Tensor



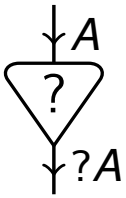
Par



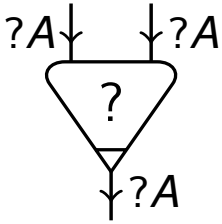
One



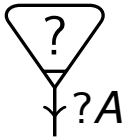
Bottom



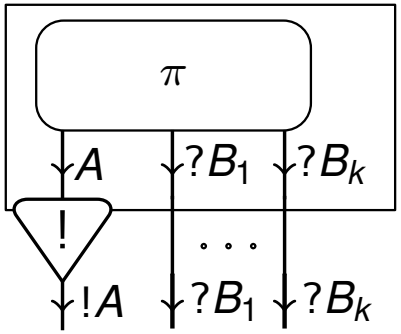
Dereliction



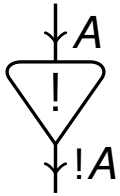
Contraction
(commutative)



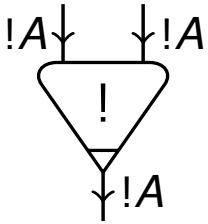
Weakening



Exponential box

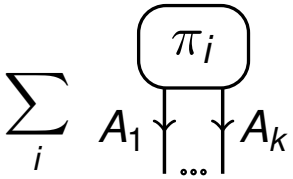


Codereliction



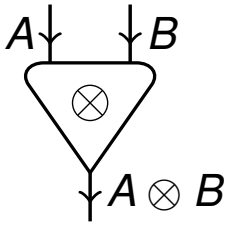
Cocontraction
(commutative)

Coweakening

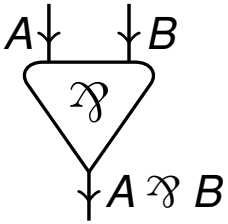


Sum

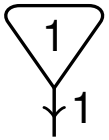
The nets: MLL



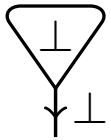
Tensor



Par



One



Bottom

Dereliction

Contraction (commutative)

Weakening

Exponential box

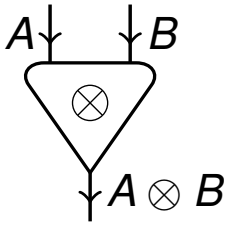
Codereliction

Cocontraction (commutative)

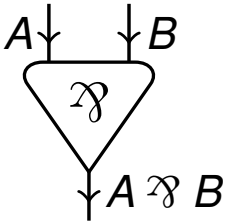
Coweakening

Sum

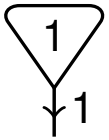
The nets: MELL



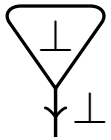
Tensor



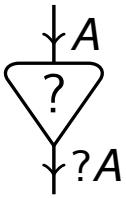
Par



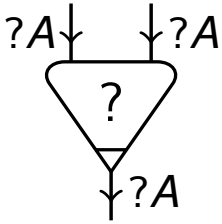
One



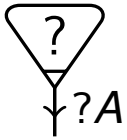
Bottom



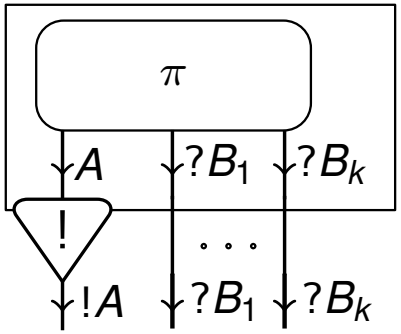
Dereliction



Contraction
(commutative)



Weakening



Exponential box

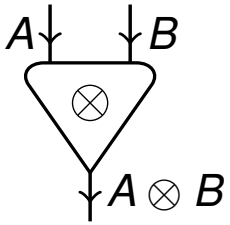
Codereliction

Cocontraction
(commutative)

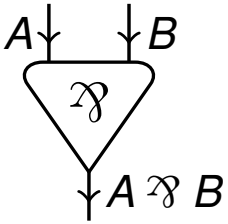
Coweakening

Sum

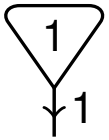
The nets: Differential Interaction Nets



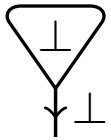
Tensor



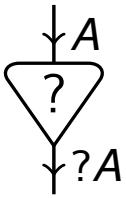
Par



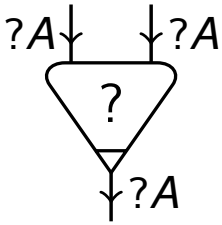
One



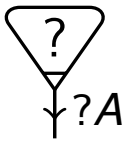
Bottom



Dereliction

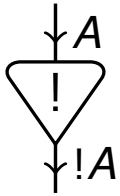


Contraction (commutative)

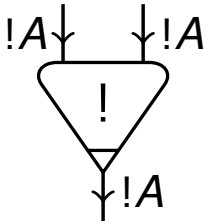


Weakening

Exponential box

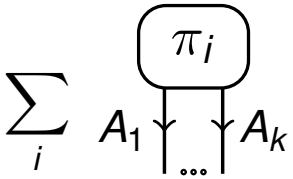


Codereliction



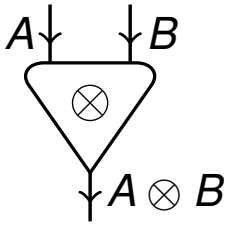
Cocontraction (commutative)

Coweakening

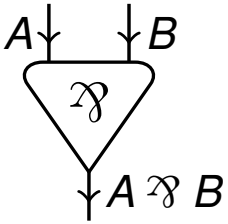


Sum

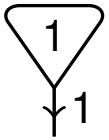
The nets: Differential Nets (DiLL)



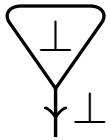
Tensor



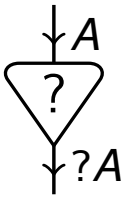
Par



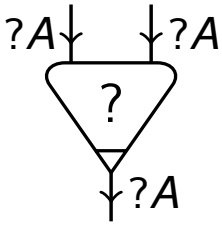
One



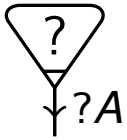
Bottom



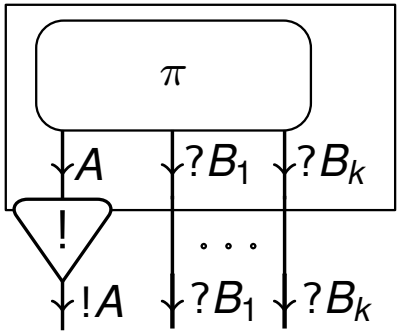
Dereliction



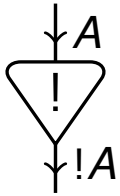
Contraction (commutative)



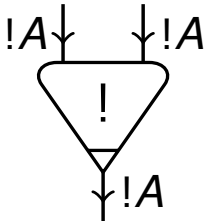
Weakening



Exponential box

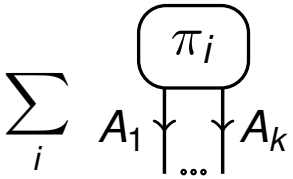


Codereliction



Cocontraction (commutative)

Coweakening

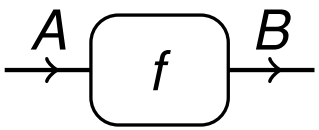


Sum

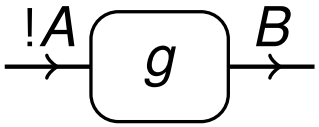
Developing ternary Curry-Howard

Let us develop the idea at the level of syntax

$f : A \multimap B \quad \rightsquigarrow \quad$ linear functions $\quad \rightsquigarrow$



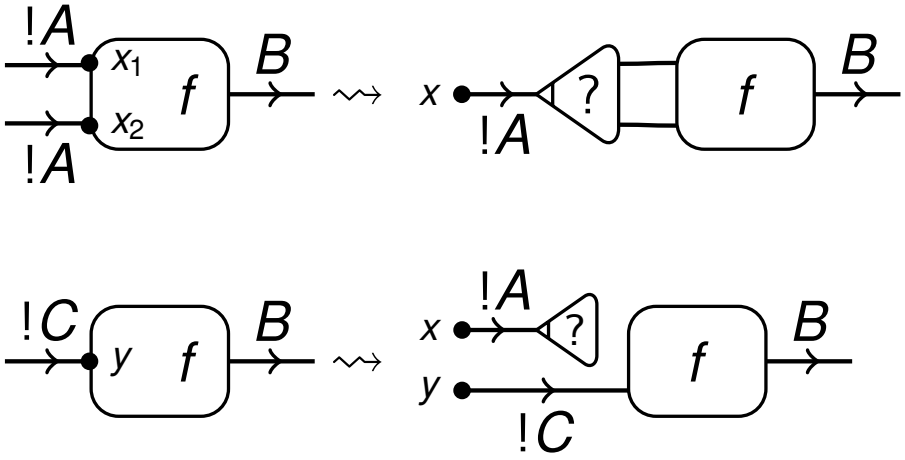
$g : A \to B \quad \rightsquigarrow \quad$ analytic functions $\quad \rightsquigarrow$



The usual suspects / Contraction and weakening

Analysis: sharing of variables, constant functions

Nets:

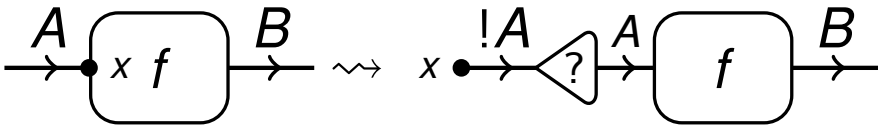


Programs: duplication (joining queries) and erasing (dummy variables, empty query).

The usual suspects / Dereliction

Analysis: Declaring a linear function to be analytical

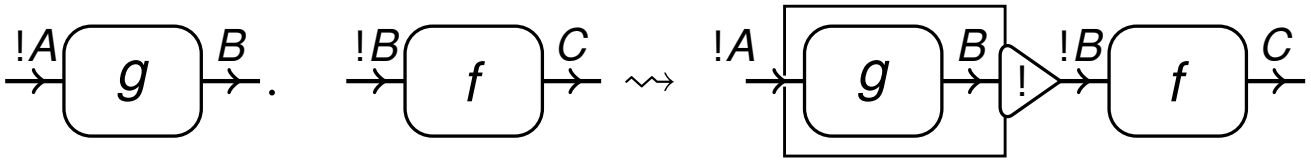
$$f \cdot x = 0 + f \cdot x + \sum_{n=2}^{\infty} \mathbf{0}(x^n)$$

Nets: 

Programs: query of a single use of an input (occurrences of variables).

The usual suspects / Promotion

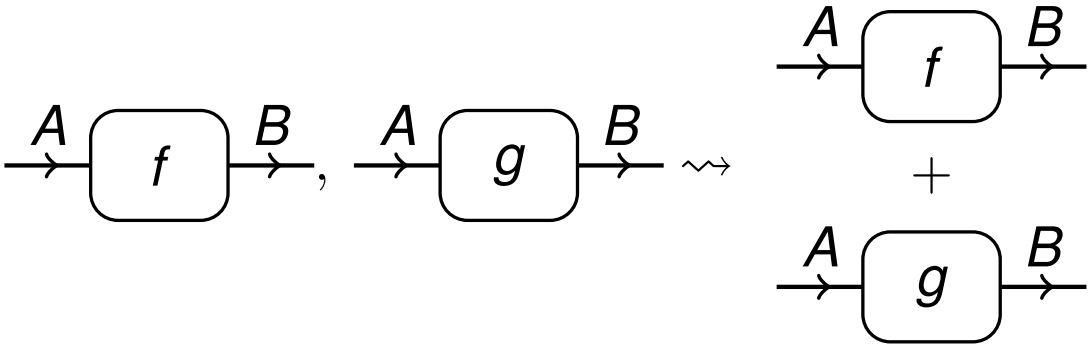
Analysis: Composition of analytical functions.

Nets: 

Programs: reusable packages.

The unusual suspects / Sum

Analysis: $(f+g)(x) := f(x) + g(x)$.

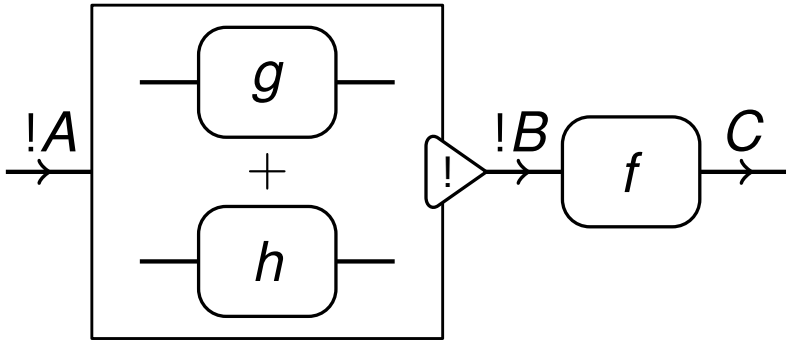
Nets: 

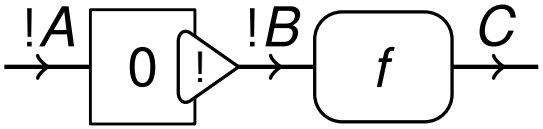
Programs: Nondeterminism (or independent parallelism).

- There is also the function/net/process 0, of any type.
- We call sums of nets polynets.

Sums and promotion

Sums and 0 do not exit boxes.
Evaluation function $f, x \mapsto f(x)$ linear in f but not in x.

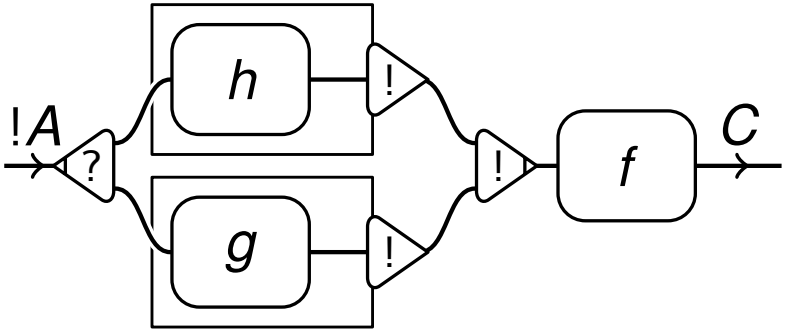


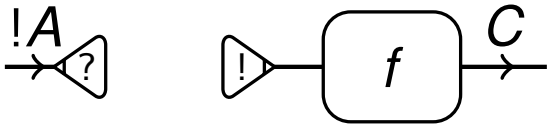


However we need more atomic constructs, and we introduce cocontractions and coweakenings.

Sums and promotion

Sums and 0 do not exit boxes.
Evaluation function $f, x \mapsto f(x)$ linear in f but not in x.

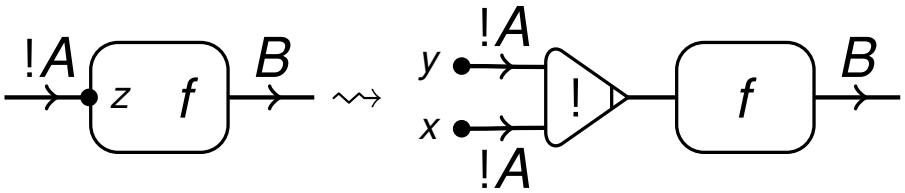




However we need more atomic constructs, and we introduce cocontractions and coweakenings.

The unusual supspects / Cocontraction

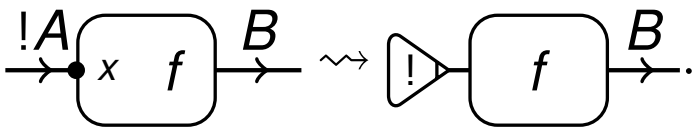
Analysis: $f(x+y)$ out of $f(z)$.

Nets: 

Programs: joining resources.

The unusual suspects / Coweakening

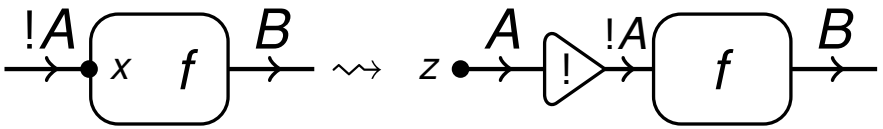
Analysis: Evaluation in 0.

Nets: 

Programs: Empty resource.

The unusual suspects / Codereliction

Analysis: Derivative in 0, giving a linear function: $\left.\frac{\partial f(x)}{\partial x}\right|_{x=0} \cdot z$.

Nets: 

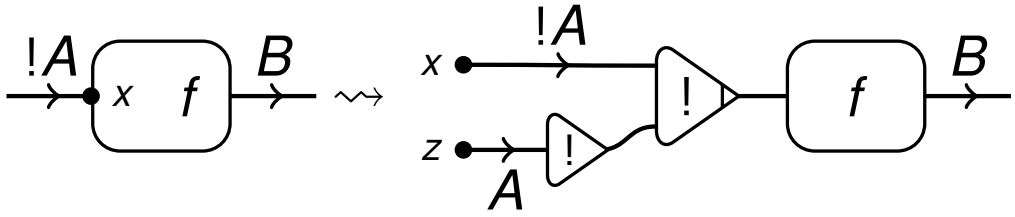
Programs: a single use resource.

General partial derivation

Analysis:

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

Nets:



Programs: linear substitution.

Outline

Reductions

$$\pi \quad \longrightarrow \quad \pi_1$$

- 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 $h \geq 1$.

Reductions

$$\pi \longrightarrow \quad \pi_1 + \cdots + \quad \pi_k$$

- 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 $h \geq 1$.

Reductions

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

- 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 $h \geq 1$.

Reductions

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

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 $h \geq 1$.

Reductions

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

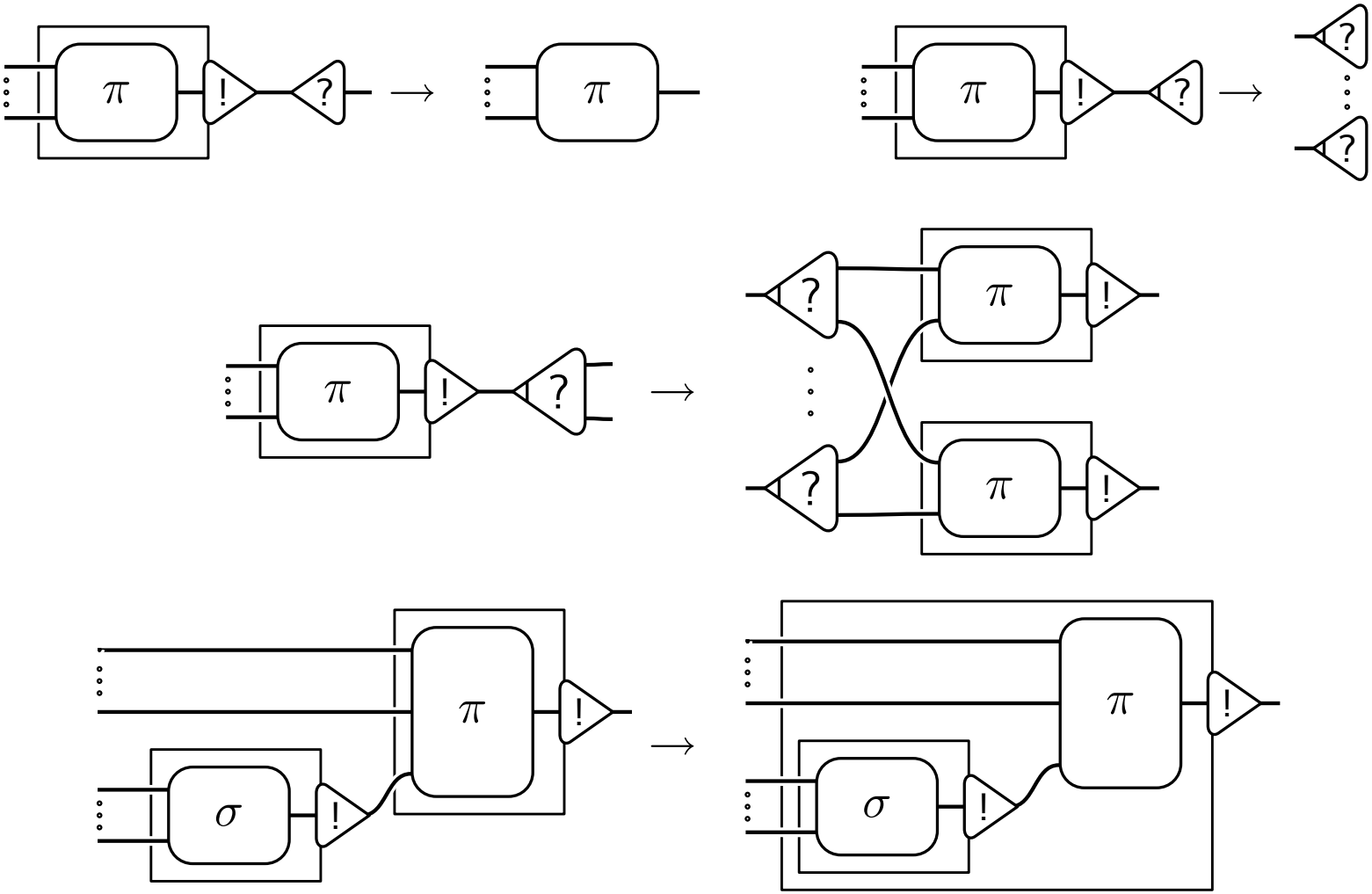
- In LL reduction is deterministic.
- In DiLL reduction is nondeterministic.
- Reduction rules extended by context closure as usual.
- Sums duplicate the context, without spreading outside boxes.
- As to reducing sums, two flavours
  - $\lambda_1 + \cdots + \lambda_k \to \mu_1 + \lambda_2 + \cdots +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

Reductions

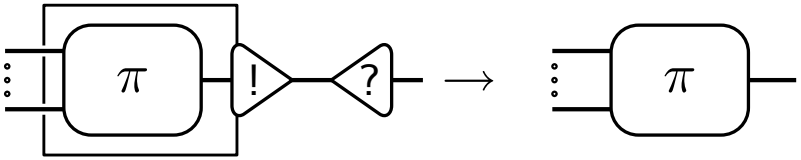
$$\omega[!\pi] \to \omega[!(\pi_1 \ + \cdots + \quad \pi_k)]$$

- 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 $h \geq 1$.

The known reductions



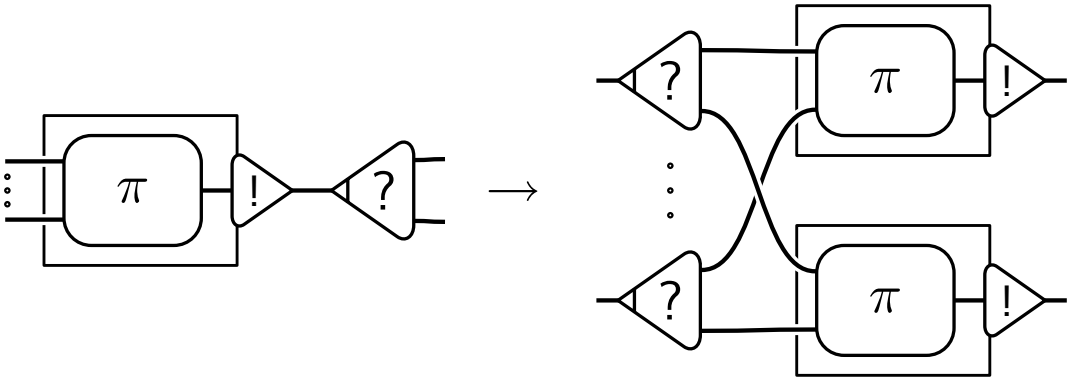
The known reductions



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

Composition with an analytical function
which is linear

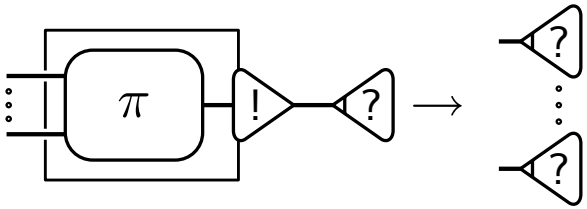
The known reductions



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

Composition on a shared variable

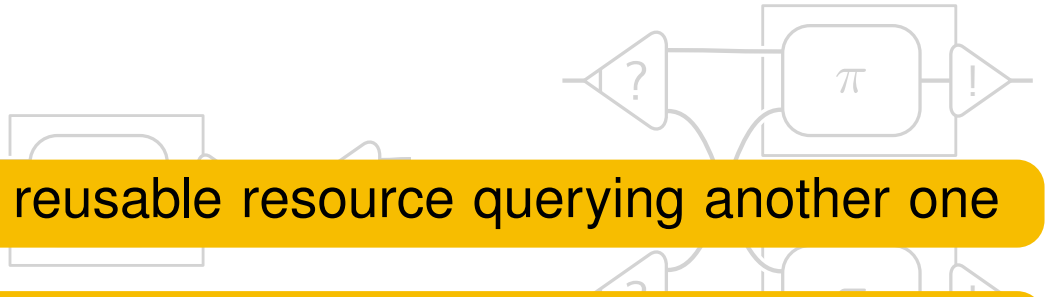
The known reductions



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

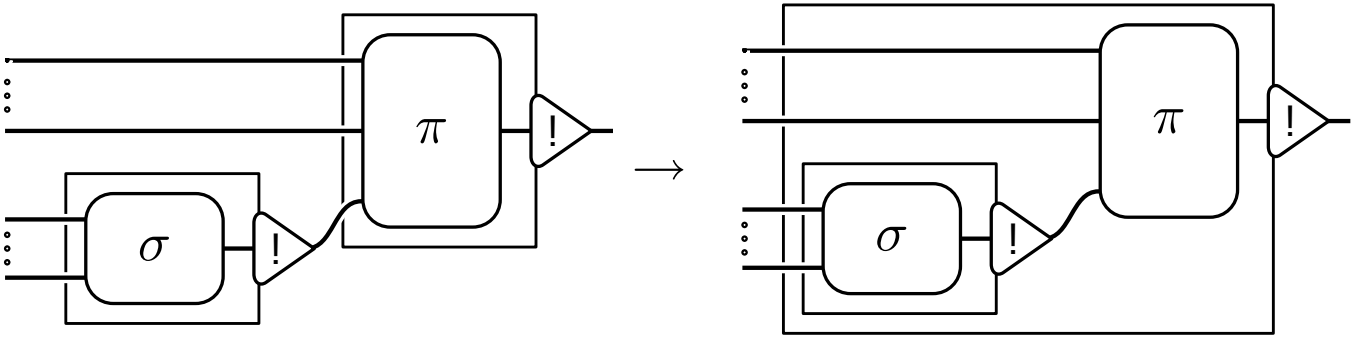
Composition with a constant function

The known reductions

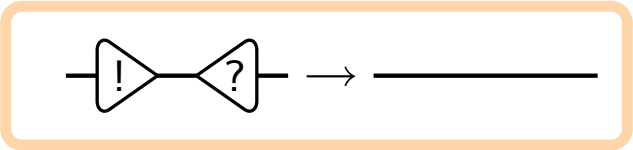


A reusable resource querying another one

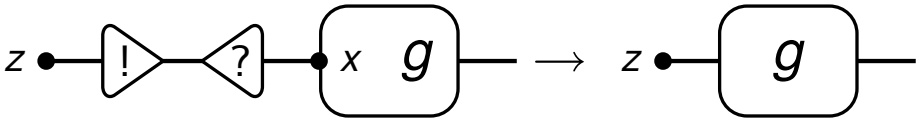
Associativity of composition



Codereliction vs dereliction

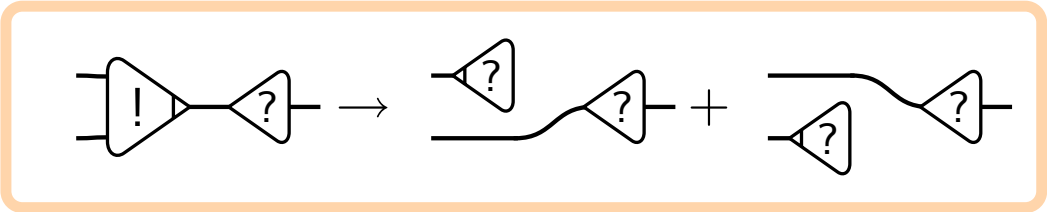


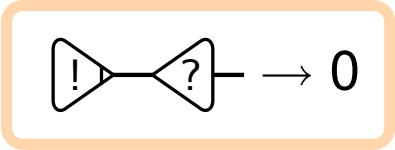
Analysis: $\dfrac{\partial g \cdot x}{\partial x} = g \implies \left.\dfrac{\partial g \cdot x}{\partial x}\right|_{x=0} \cdot z = g \cdot z$:



Programs: a single-use query encounters a single-use resource and is resolved.

Cocontranction and coweakening vs dereliction





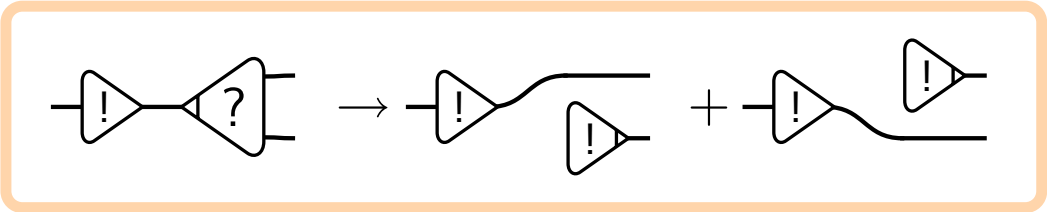
Analysis: linearity!

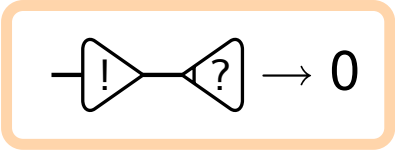
$$g \cdot (x + y) = g \cdot x + g \cdot y, \qquad g \cdot 0 = 0.$$

Programs: a single-use query presented with two resources $\rightsquigarrow$ nondeterministic choice.
Or presented with an empty one $\rightsquigarrow$ nondeterministic dead end.

Codereliction vs contraction and coweakening

Completely simmetric!





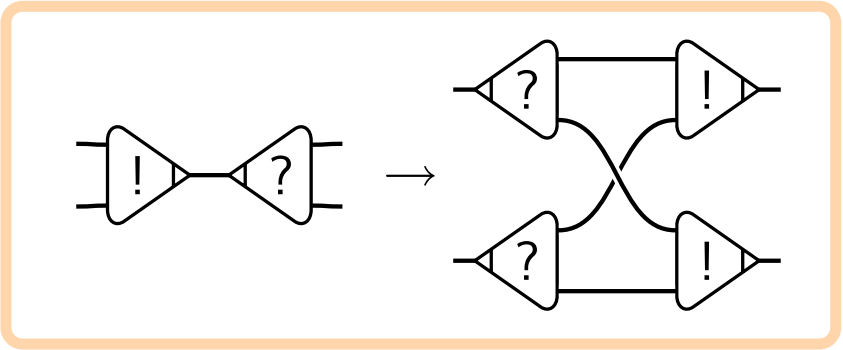
Analysis: laws of derivation!

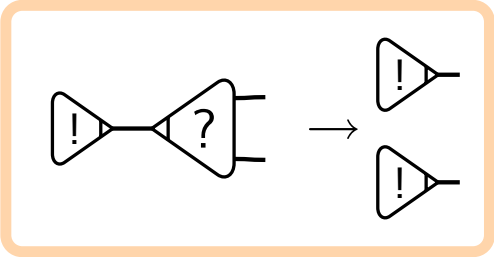
$$\frac{\partial f(x,x)}{\partial x}\Big|_{x=0} = \Big(\frac{\partial f(y,z)}{\partial y}, \frac{\partial f(y,z)}{\partial z}\Big)\Big|_{y,z=x} \cdot \frac{\partial (x,x)}{\partial x}\Big|_{x=0} = \frac{\partial f(y,0)}{\partial y}\Big|_{y=0} + \frac{\partial f(0,z)}{\partial z}\Big|_{z=0}$$

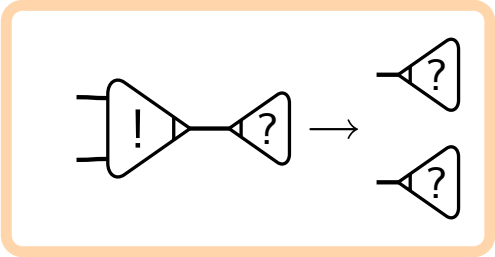
and $\frac{\partial k}{\partial x} = 0$.

Programs: two queries meets a single-use resource $\rightsquigarrow$ nondeterministic choice.
Or an empty query meets a single use resource $\rightsquigarrow$ an error (not affine).

Routing: (co)contractions and (co)weakenings





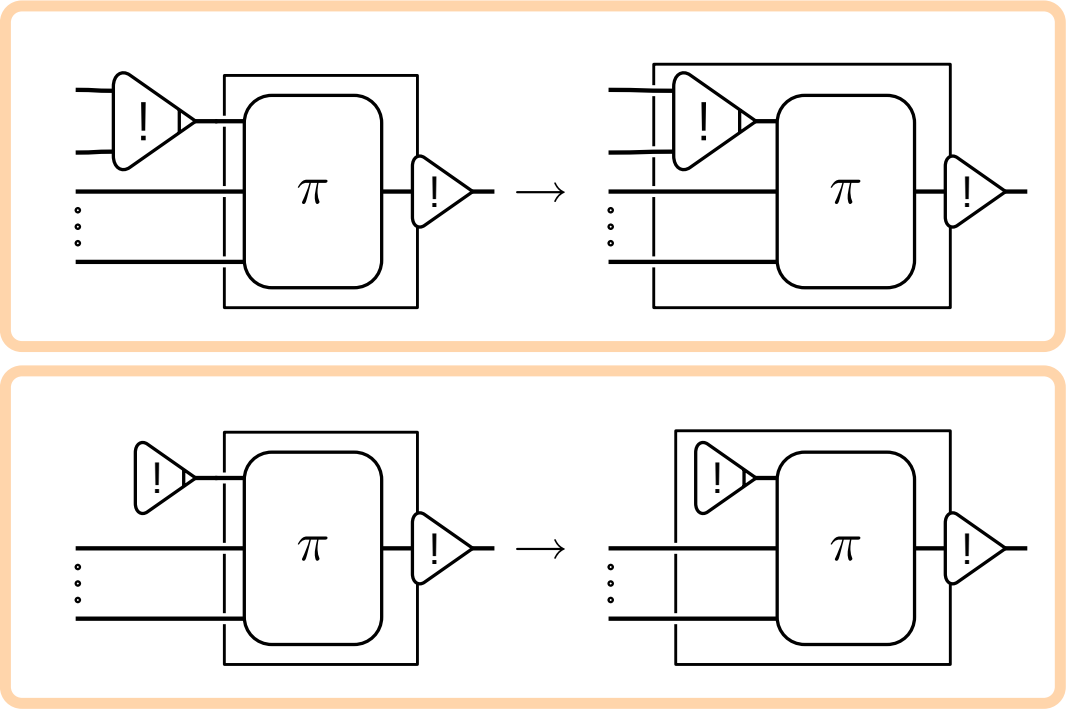


Analysis: Commutation of sum and sharing.

Programs: routing of queries and resources.

Cocontraction and coweakening vs box

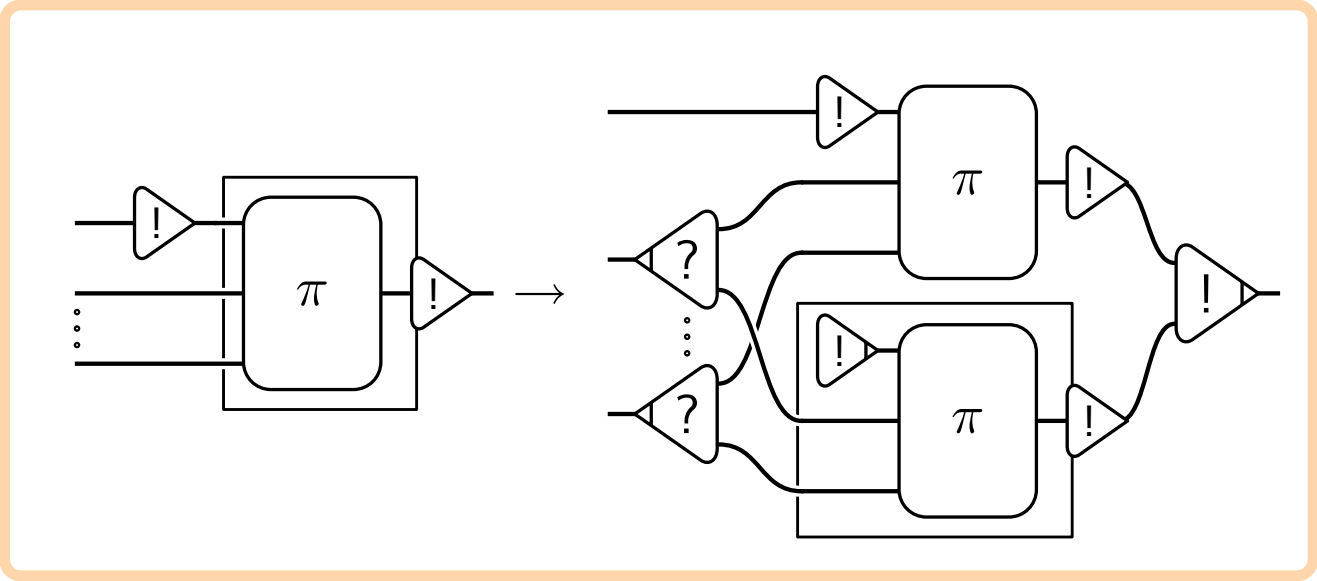
Recall that cocontractions and coweakenings are, in a way, boxes.



analysis: plain composition.

programs: operations on resources transported inside a package.

Pandora's box: codereliction vs box

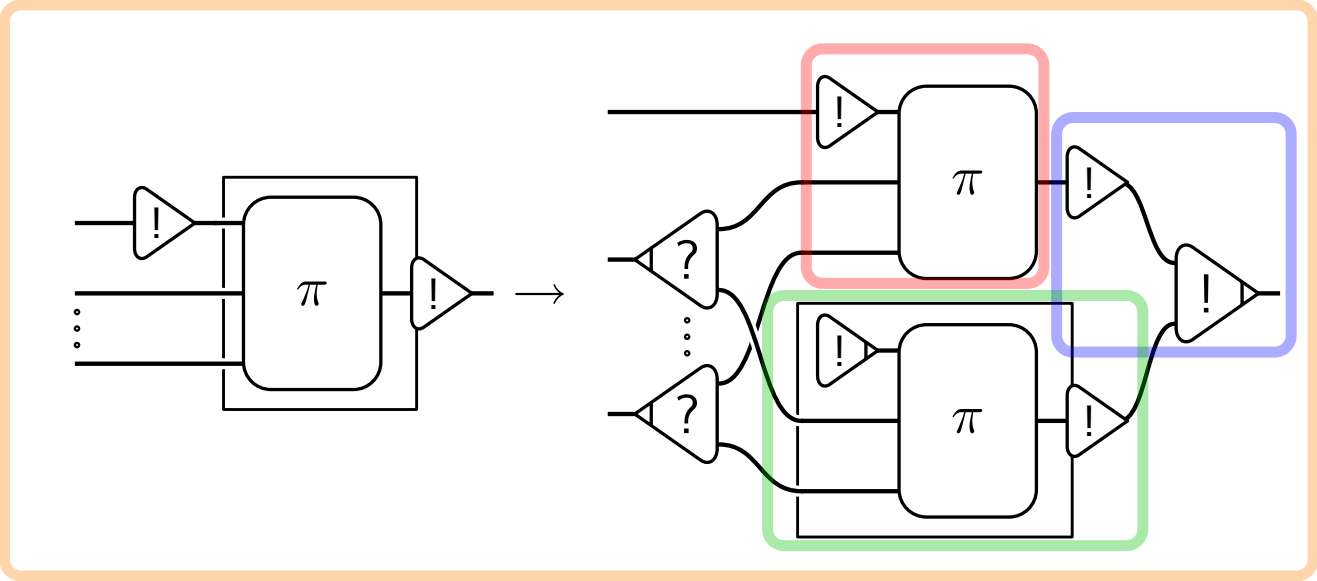


Analysis: $\frac{\partial f(g(x))}{\partial x}\Big|_{x=0} \cdot z = \frac{\partial f(y)}{\partial y}\Big|_{y=g(0)} \cdot \frac{\partial g(x)}{\partial x}\Big|_{x=0} \cdot z$

The first element is general partial derivation, which we have already seen.

Programs: reusable package gets a single-use resource $\rightsquigarrow$ a single-use copy gets the resource, others an empty one.

Pandora's box: codereliction vs box

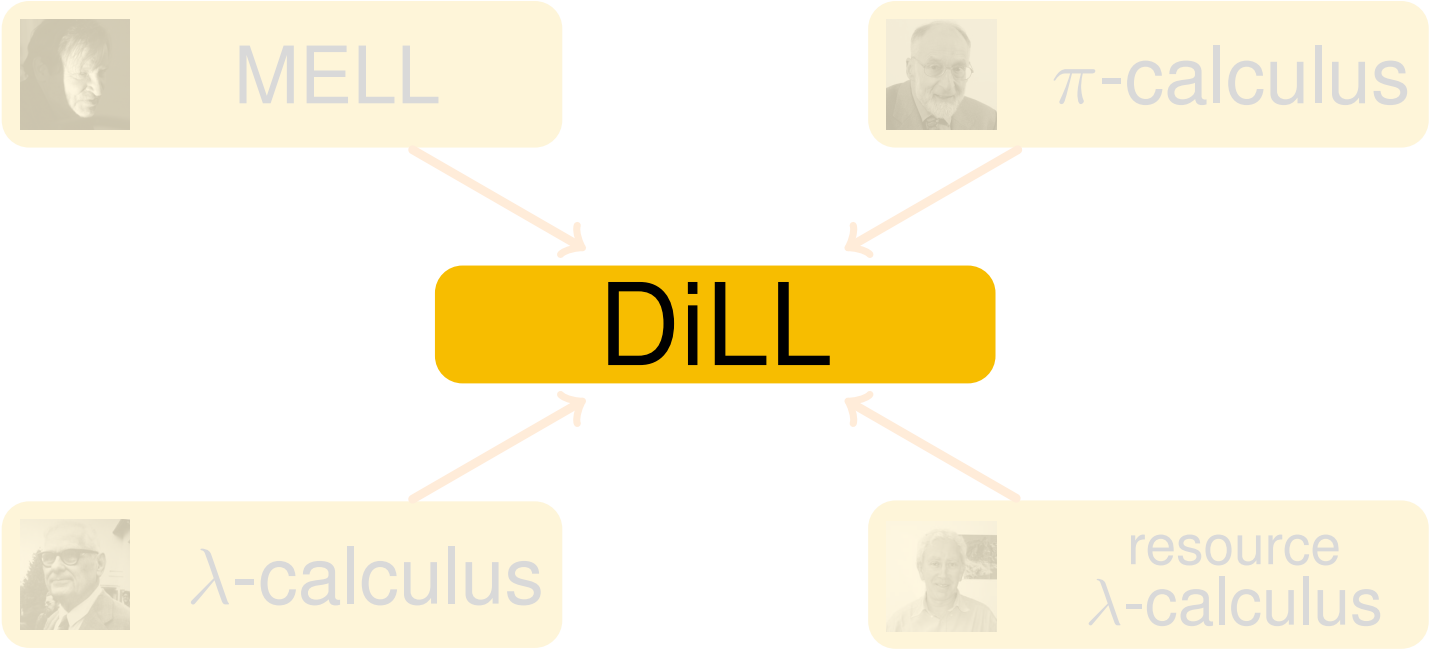


Analysis: $\frac{\partial f(g(x))}{\partial x}\Big|_{x=0} \cdot z = \frac{\partial f(y)}{\partial y}\Big|_{y=g(0)} \cdot \frac{\partial g(x)}{\partial x}\Big|_{x=0} \cdot z$

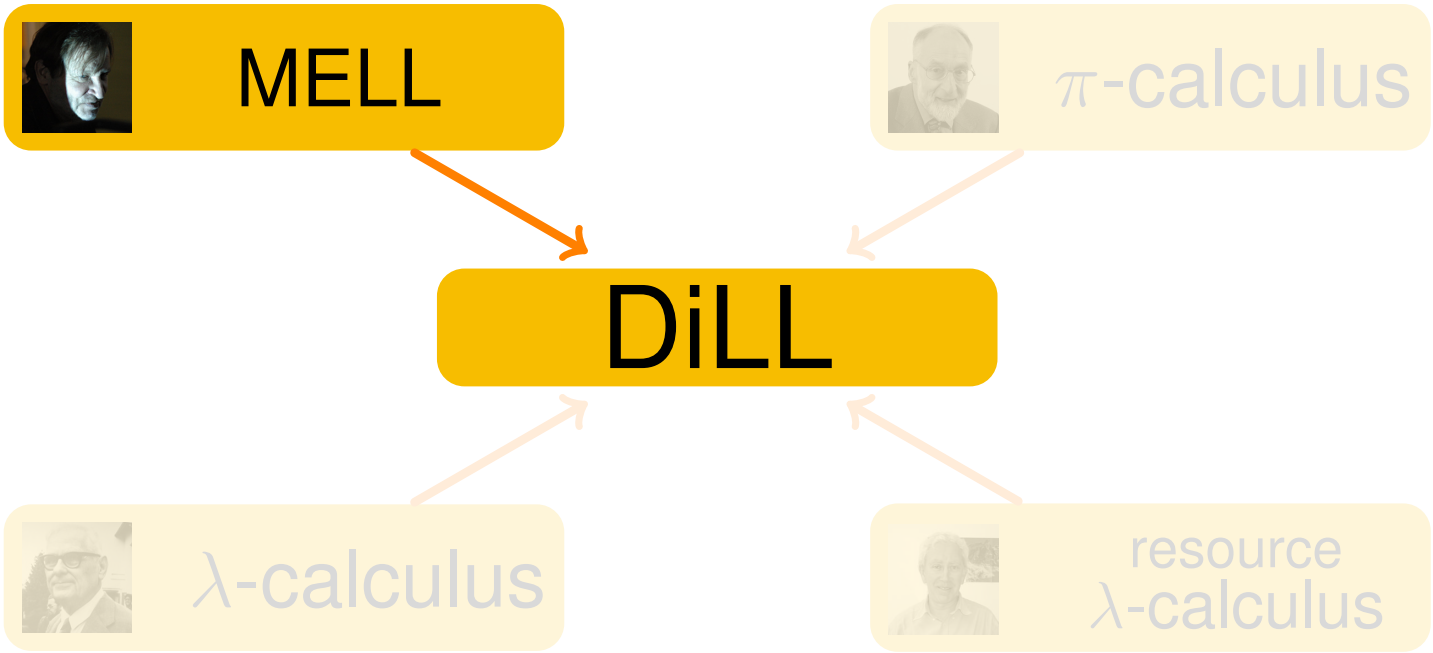
The first element is general partial derivation, which we have already seen.

Programs: reusable package gets a single-use resource $\rightsquigarrow$ a single-use copy gets the resource, others an empty one.

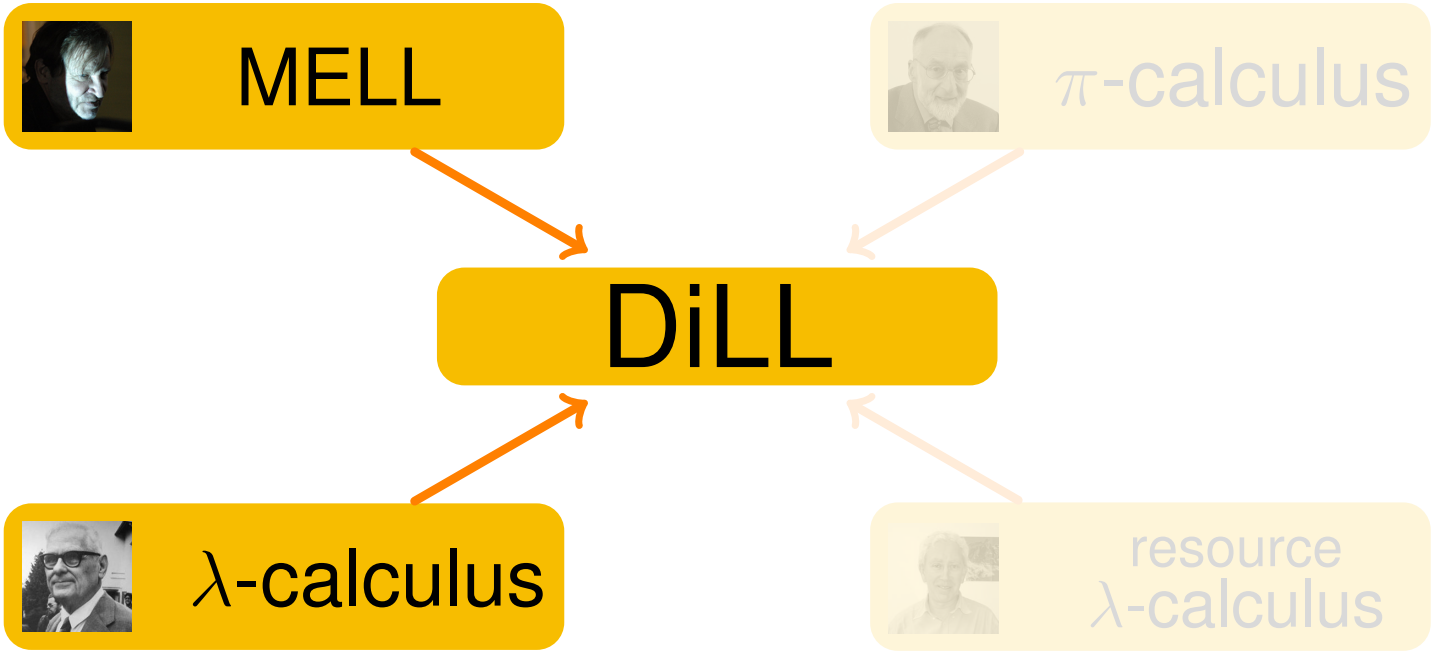
Quite an expressive system



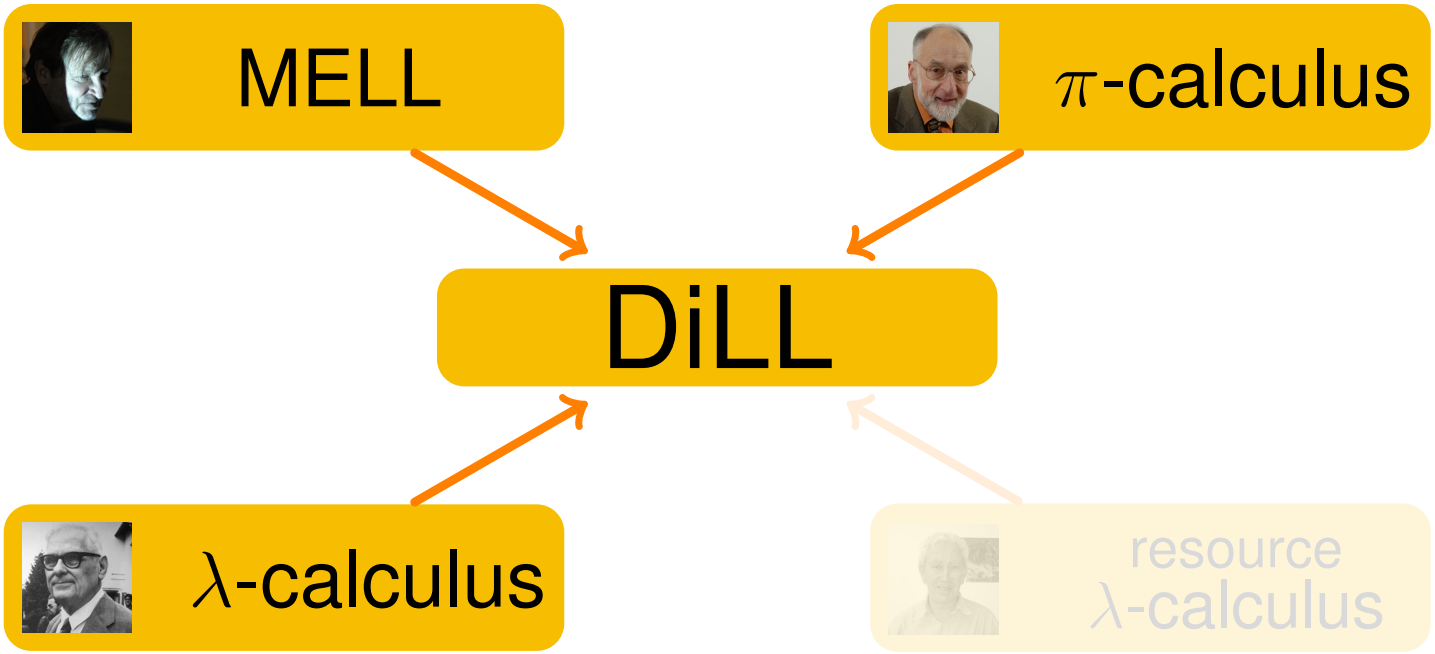
Quite an expressive system



Quite an expressive system

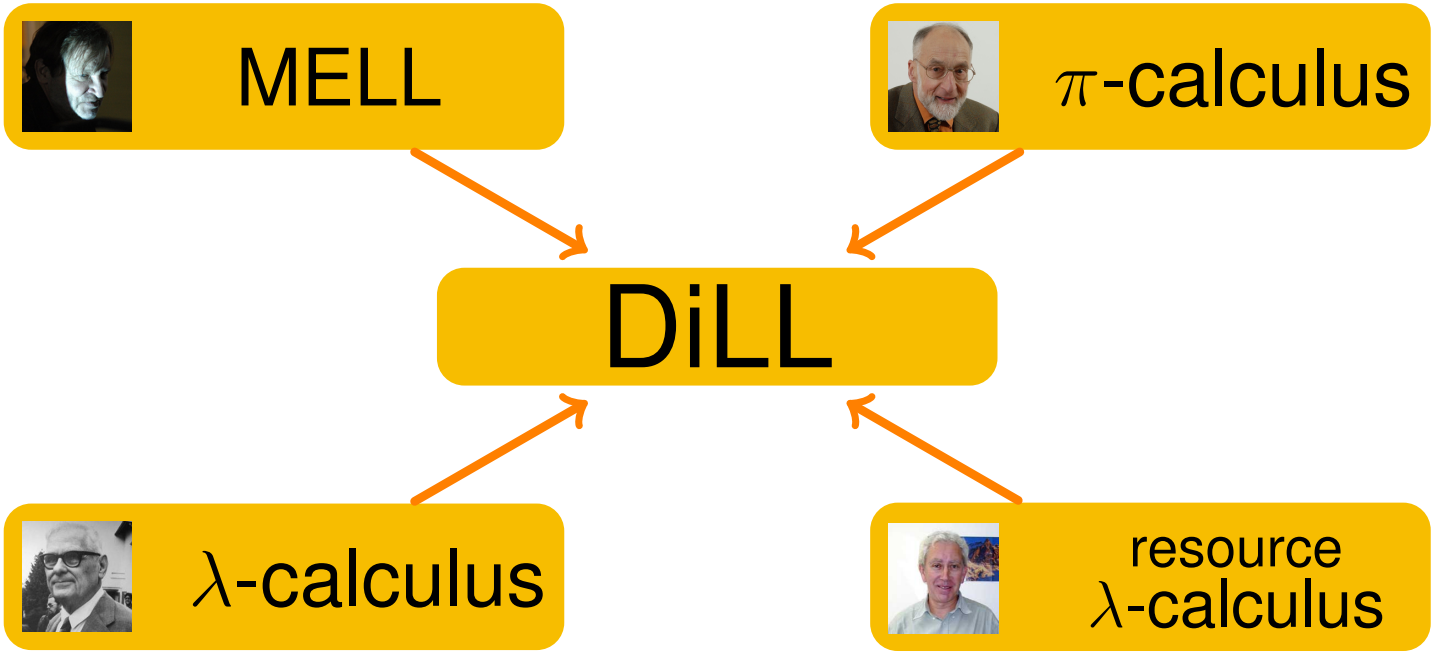


Quite an expressive system



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.

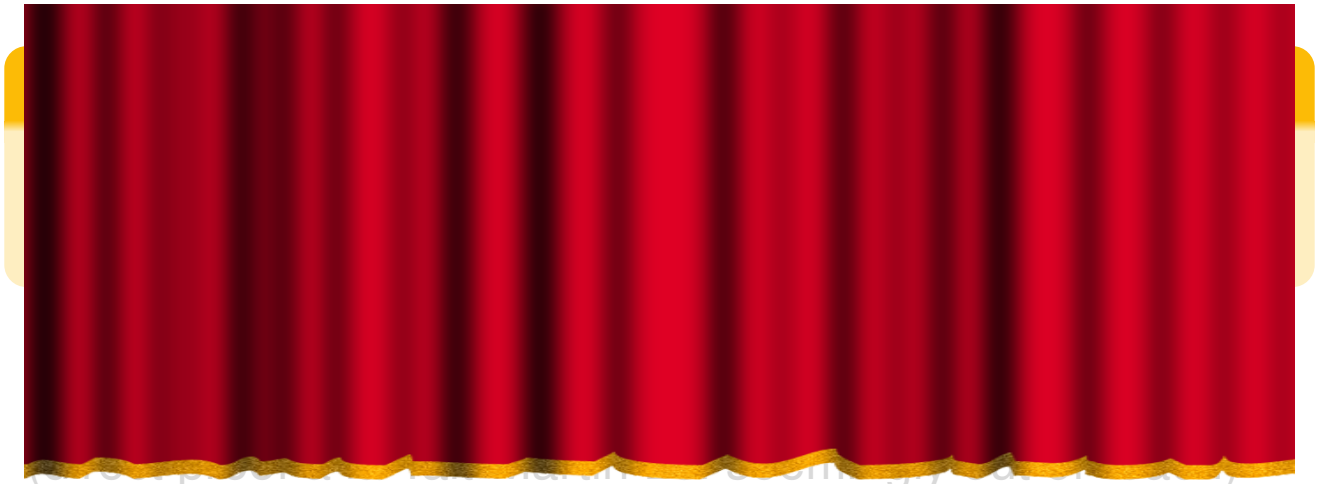
Paolo Tranquilli.
Intuitionistic differential nets and lambda calculus.
Conditionally accepted to *Theor. Comput. Sci.*, 2008.

Outline

What do we want?

Theorem

Reduction in the untyped case is confluent.



What do we want?

Theorem

Reduction in the untyped case is confluent.

Theorem (Finite developments)

Reduction not reducing new redexes is strongly normalizing.

Local confluence of developments $\Longrightarrow$ 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" cuts is strongly normalizing.

Local confluence of developments $\Longrightarrow$ 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 *Theor. Comput. Sci.*, 2008.

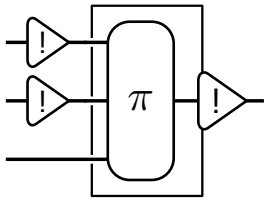
Nondeterminism and confluence?

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

A: confluence ensures nondeterministic choice is not triggered by what we reduce.

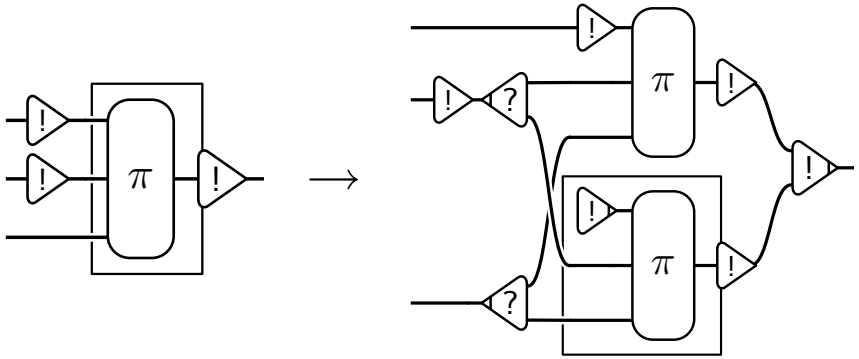
Reducing two different redexes gives the same set of nondeterministic choices in the end.

We need associativity and neutrality



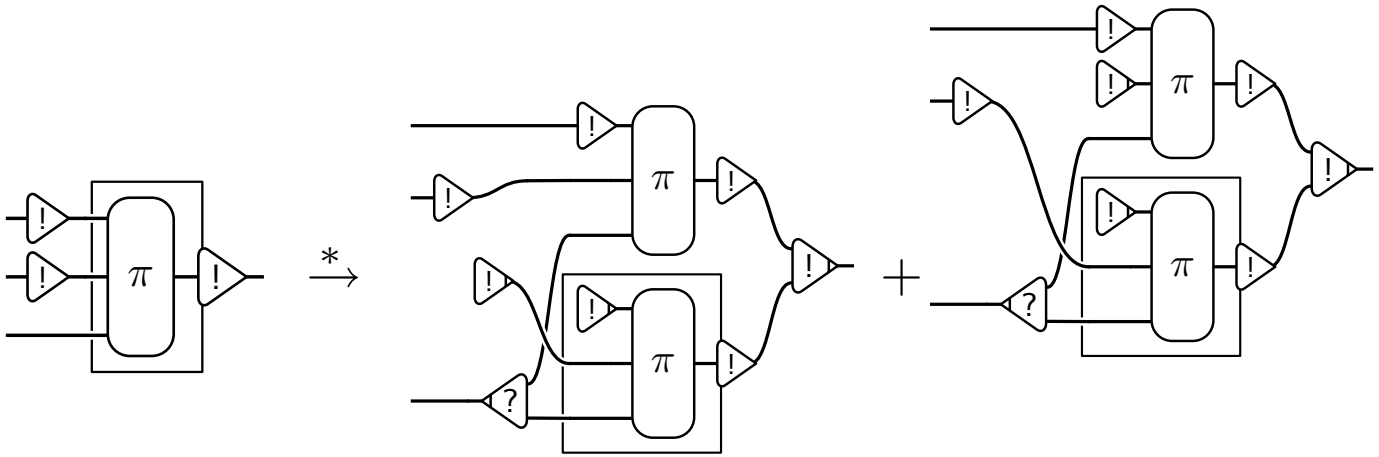
- Contractions and cocontractions need to be swapped to have confluence (associativity).
- Other confluence diagrams require merging (co)weakenings with (co)contractions (neutrality).

We need associativity and neutrality



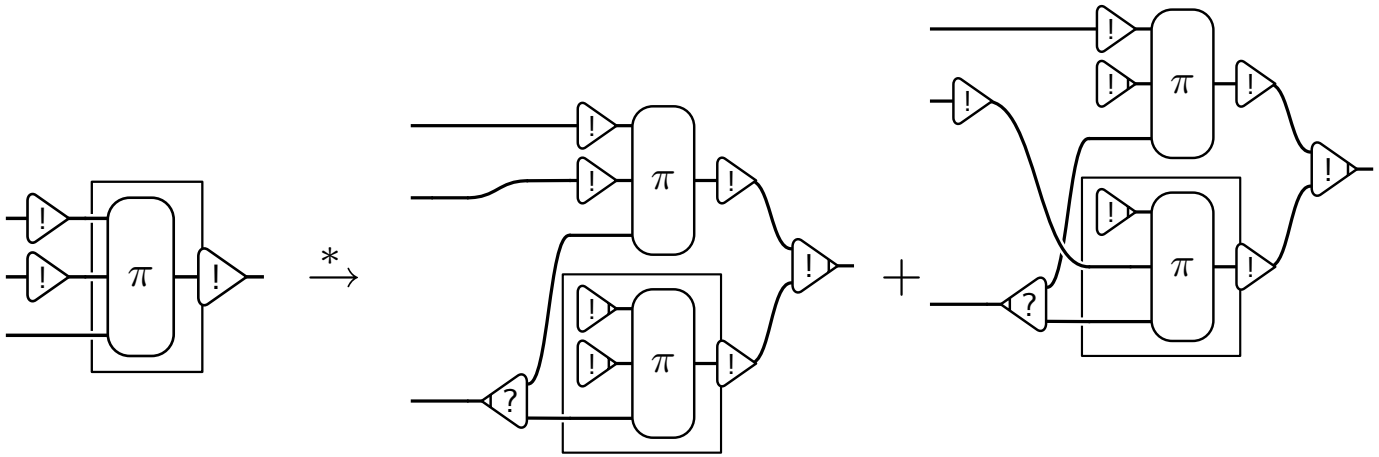
- Contractions and cocontractions need to be swapped to have confluence (associativity).
- Other confluence diagrams require merging (co)weakenings with (co)contractions (neutrality).

We need associativity and neutrality



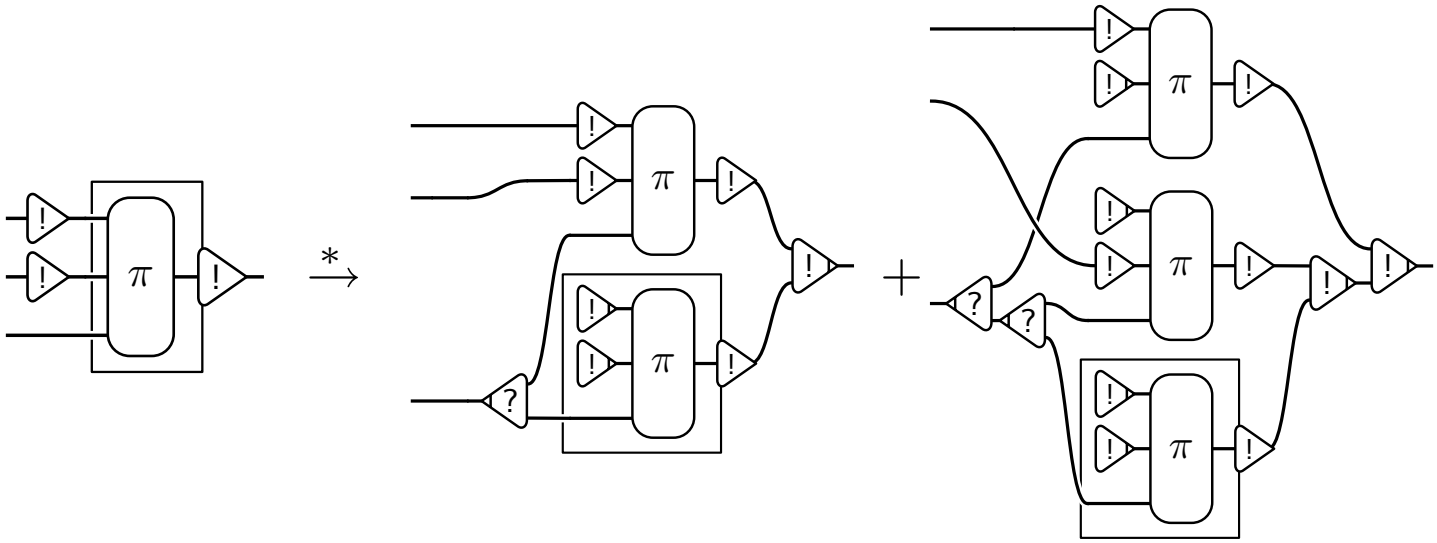
- Contractions and cocontractions need to be swapped to have confluence (associativity).
- Other confluence diagrams require merging (co)weakenings with (co)contractions (neutrality).

We need associativity and neutrality



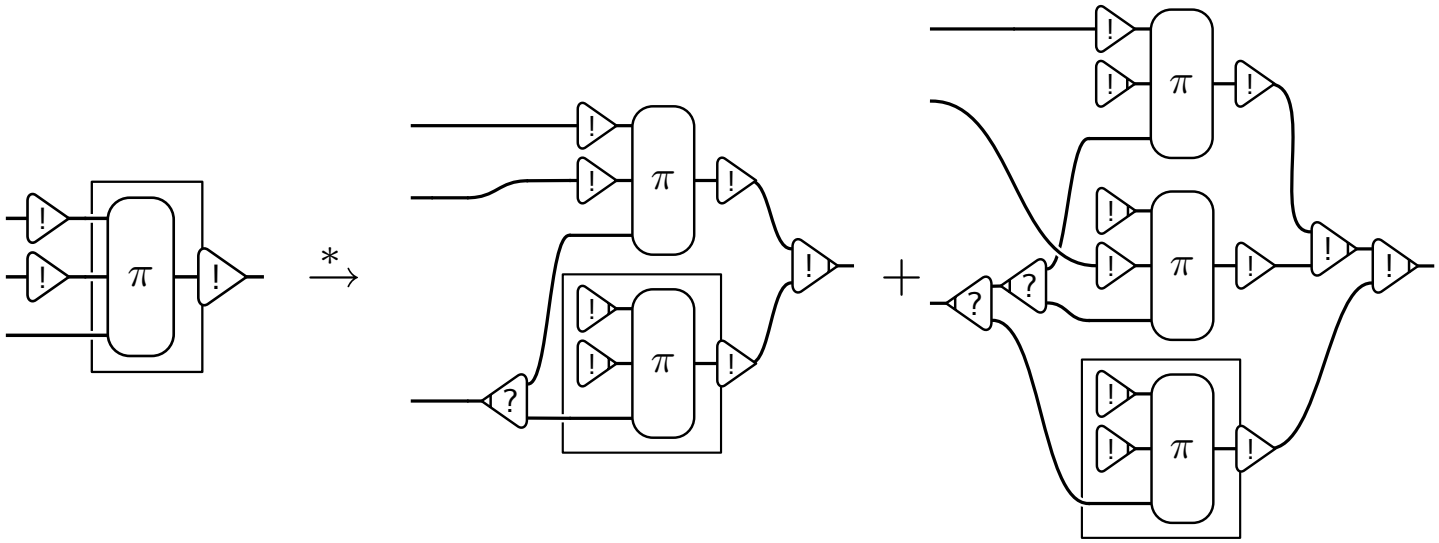
- Contractions and cocontractions need to be swapped to have confluence (associativity).
- Other confluence diagrams require merging (co)weakenings with (co)contractions (neutrality).

We need associativity and neutrality



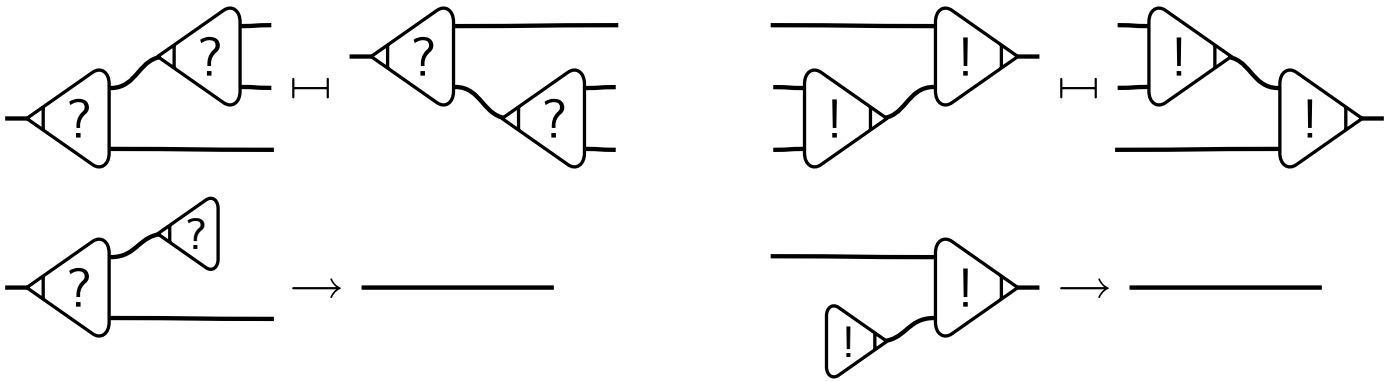
- Contractions and cocontractions need to be swapped to have confluence (associativity).
- Other confluence diagrams require merging (co)weakenings with (co)contractions (neutrality).

We need 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).

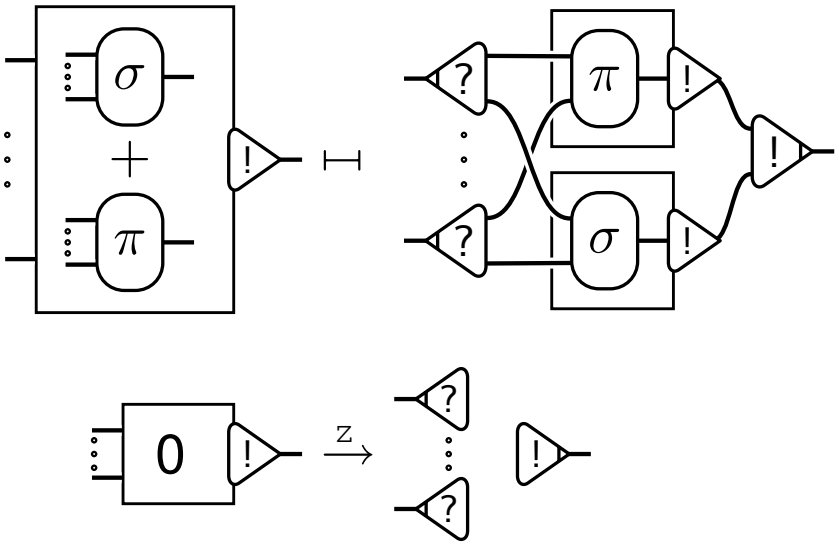
Associative equivalence and neutral reduction



- associative equivalence $\sim = \vdash^*$;
- neutral reduction (cannot be reversed).

Compulsory!

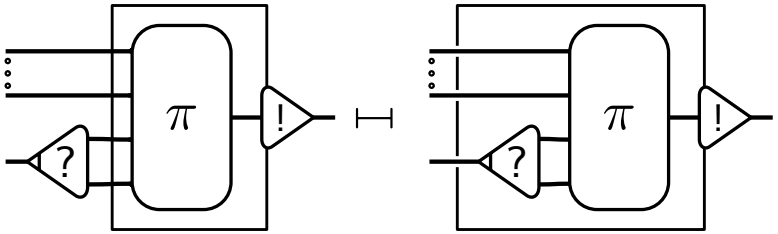
Bang sum equivalence and bang zero reduction

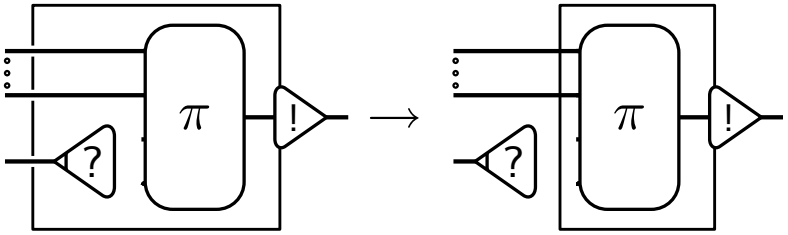


with $\sigma, \pi \neq 0$,

A reusable nondeterministic resource is equivalent to joining the various reusable resources.

Push equivalence and pull reduction

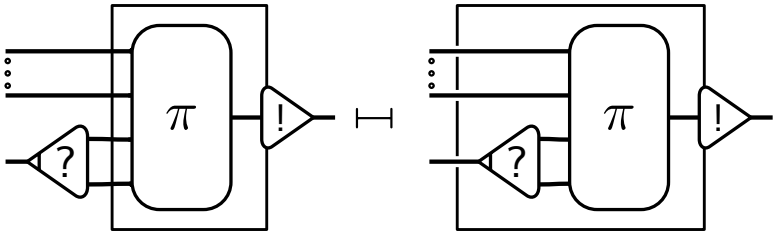


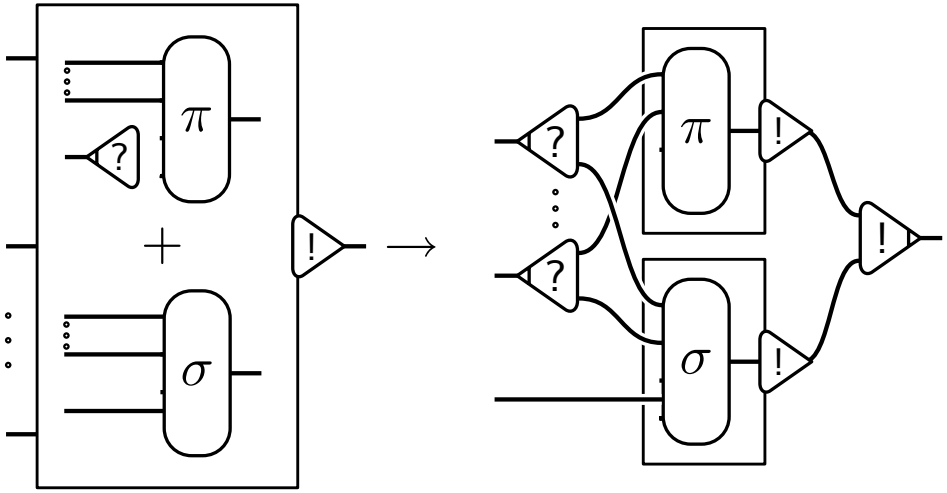


with $\pi \neq 0$.

These latter equivalences and reductions are optional (but inseparable).

Push equivalence and pull reduction





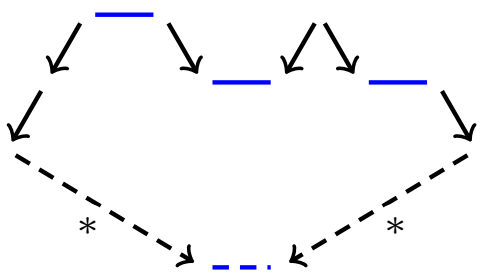
with $\pi \neq 0$.

These latter equivalences and reductions are optional (but inseparable).

Reduction modulo equivalence

- Presents problematics sometimes underestimated.
- For confluence, strongest property is Church Rosser modulo. CR$\sim$ holds if

$$(\rightarrow \cup \leftarrow \cup \sim)^* \subseteq \xrightarrow{*}\sim\xleftarrow{*}, \quad \text{for ex.}$$



It is not equivalent to confluence modulo ($\xleftarrow{*}\sim\xrightarrow{*} \subseteq \xrightarrow{*}\sim\xleftarrow{*}$), or to confluence of $A/\sim$!

- $t \in A$ is SN$\sim$ if it is SN for $\sim\rightarrow\sim$.

Enno Ohlebusch.
Church-Rosser theorems for abstract reduction modulo an equivalence relation.
In *RTA*, *LNCS* vol. 1379, pages 17–31, 1998.

What do we want?

Theorem

Reduction of switching acyclic untyped LL nets is confluent.

Theorem (Finite developments)

Reduction not reducing "new" cuts is strongly normalizing.

Local confluence of developments $\Longrightarrow$ 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 *Theor. Comput. Sci.*, 2008.

What do we want? We got it!

Theorem

Reduction of switching acyclic untyped DiLL nets is CR modulo $\sim$.

Theorem (Finite developments)

Reduction not reducing "new" cuts is strongly normalizing modulo $\sim$.

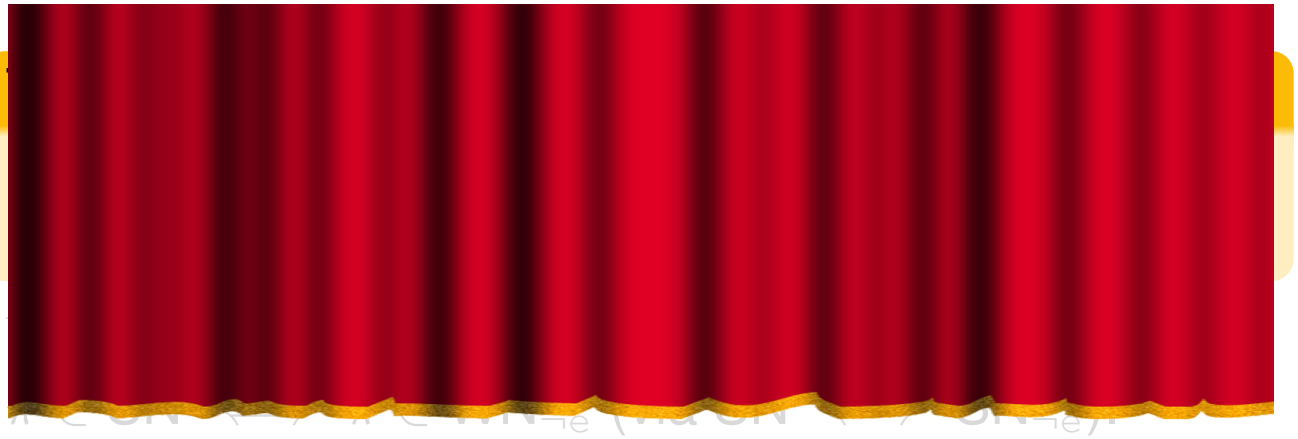
Local confluence of developments $\Longrightarrow$ strongly confluent parallel reduction.
(direct proof à la Tait-Martin Löf seemingly out of reach)

Outline

What do we want?

Theorem

Reduction is strongly normalizing in the typed case.



What do we want?

Theorem

Reduction is strongly normalizing in the typed case.

Theorem ("conservation" according to Barendregt)

Non erasing reduction preserves infinite ones.

$\overset{\neg e}{\to}$ stands for non erasing reduction. It follows that $\pi \in \mathrm{SN} \iff \pi \in \mathrm{WN}_{\neg e}$ (via $\mathrm{SN} \iff \mathrm{SN}_{\neg e}$).

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 is strongly normalizing in the typed case.

Theorem ("standardization" according to Girard)

Non erasing reduction preserves infinite ones.

$\overset{\neg e}{\to}$ stands for non erasing reduction. It follows that $\pi \in \mathrm{SN} \iff \pi \in \mathrm{WN}_{\neg e}$ (via $\mathrm{SN} \iff \mathrm{SN}_{\neg e}$).

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 is strongly normalizing in the typed case.

Theorem ("striction" according to Danos)

Non erasing reduction preserves infinite ones.

$\overset{\neg e}{\to}$ stands for non erasing reduction. It follows that $\pi \in \mathrm{SN} \iff \pi \in \mathrm{WN}_{\neg e}$ (via $\mathrm{SN} \iff \mathrm{SN}_{\neg e}$).

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 is strongly normalizing in the 2nd order typed case.

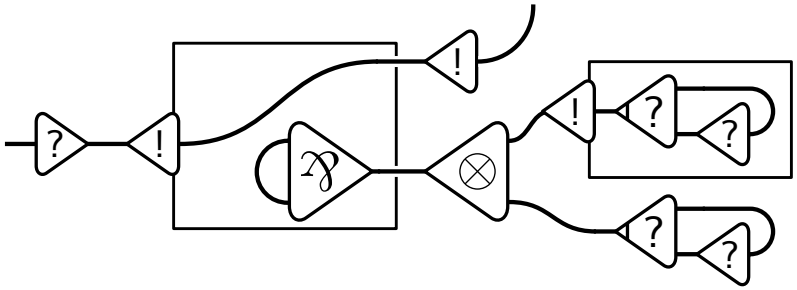
Theorem (standstrictcons?)

For π untyped switching acyclic LL net,
if $\pi \overset{\neg e}{\to} \pi'$, then $\pi' \in \mathrm{SN}_{\neg e} \implies \pi \in \mathrm{SN}_{\neg e}$.

$\overset{\neg e}{\to}$ stands for non erasing reduction. It follows that $\pi \in \mathrm{SN} \iff \pi \in \mathrm{WN}_{\neg e}$ (via $\mathrm{SN} \iff \mathrm{SN}_{\neg e}$).

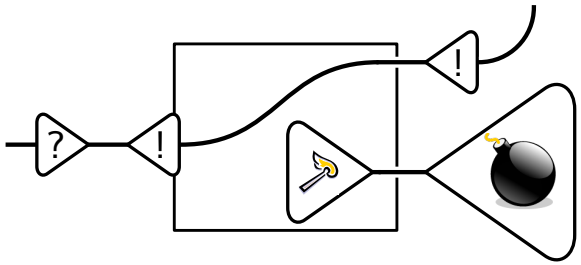
Michele Pagani and Lorenzo Tortora de Falco.
Strong normalization property for second order linear logic.
To appear on *Theor. Comput. Sci.*, 2008.

“Lazarus” clashes



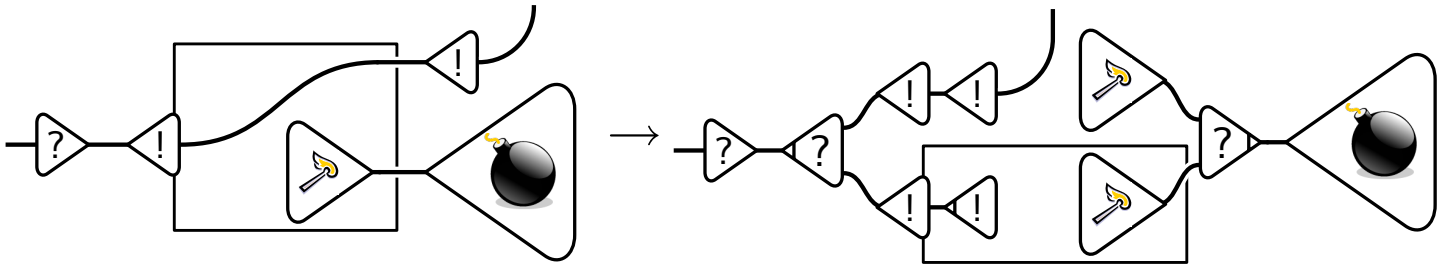
In LL Lazarus clashes do not negate standardization.

“Lazarus” clashes



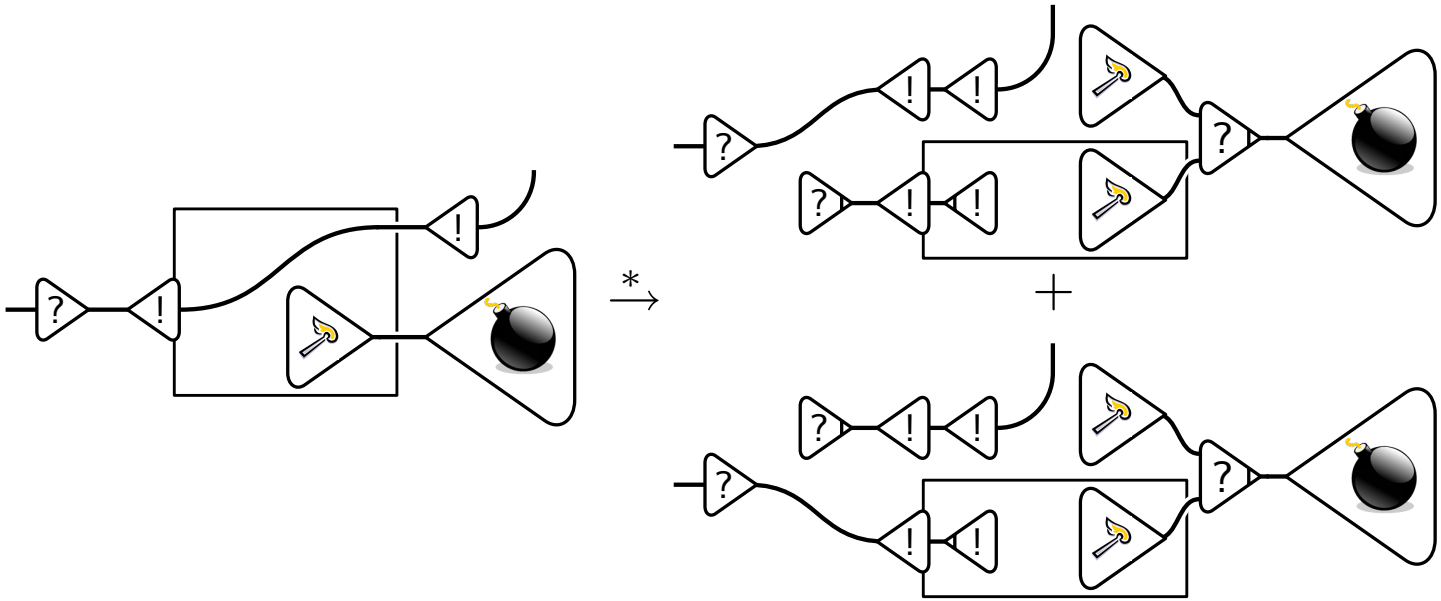
In LL Lazarus clashes do not negate standardization.

“Lazarus” clashes



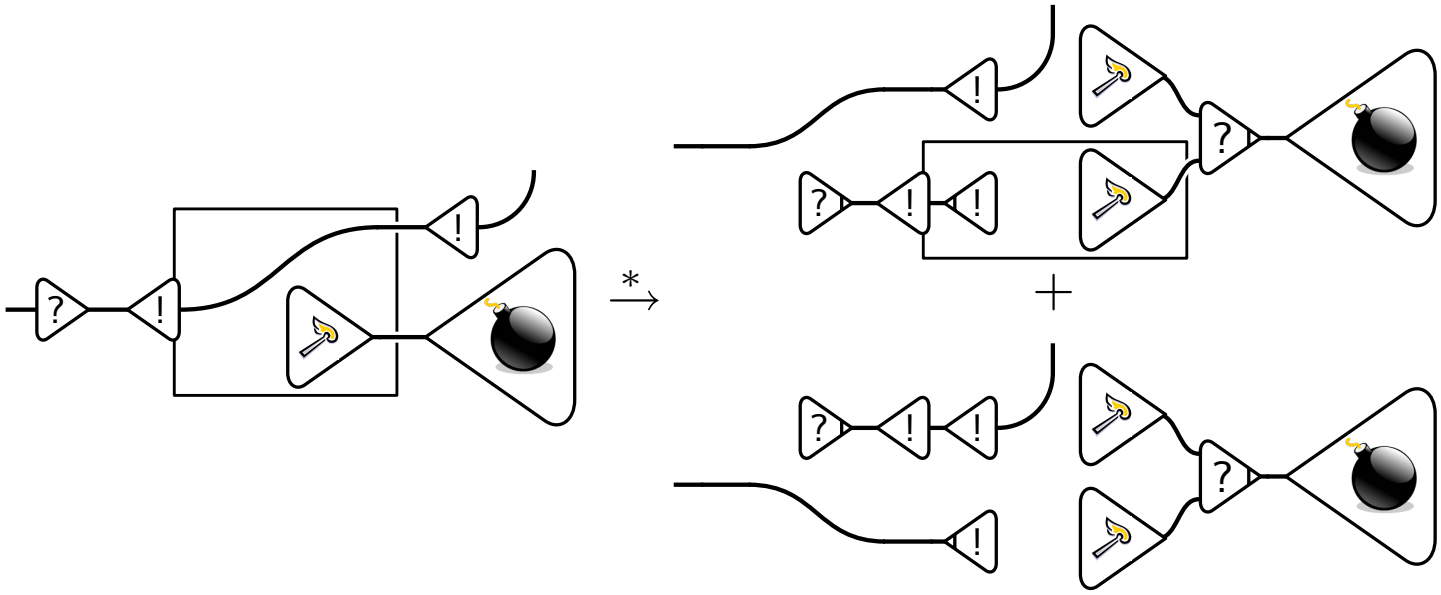
In LL Lazarus clashes do not negate standardization.

“Lazarus” clashes



In LL Lazarus clashes do not negate standardization.

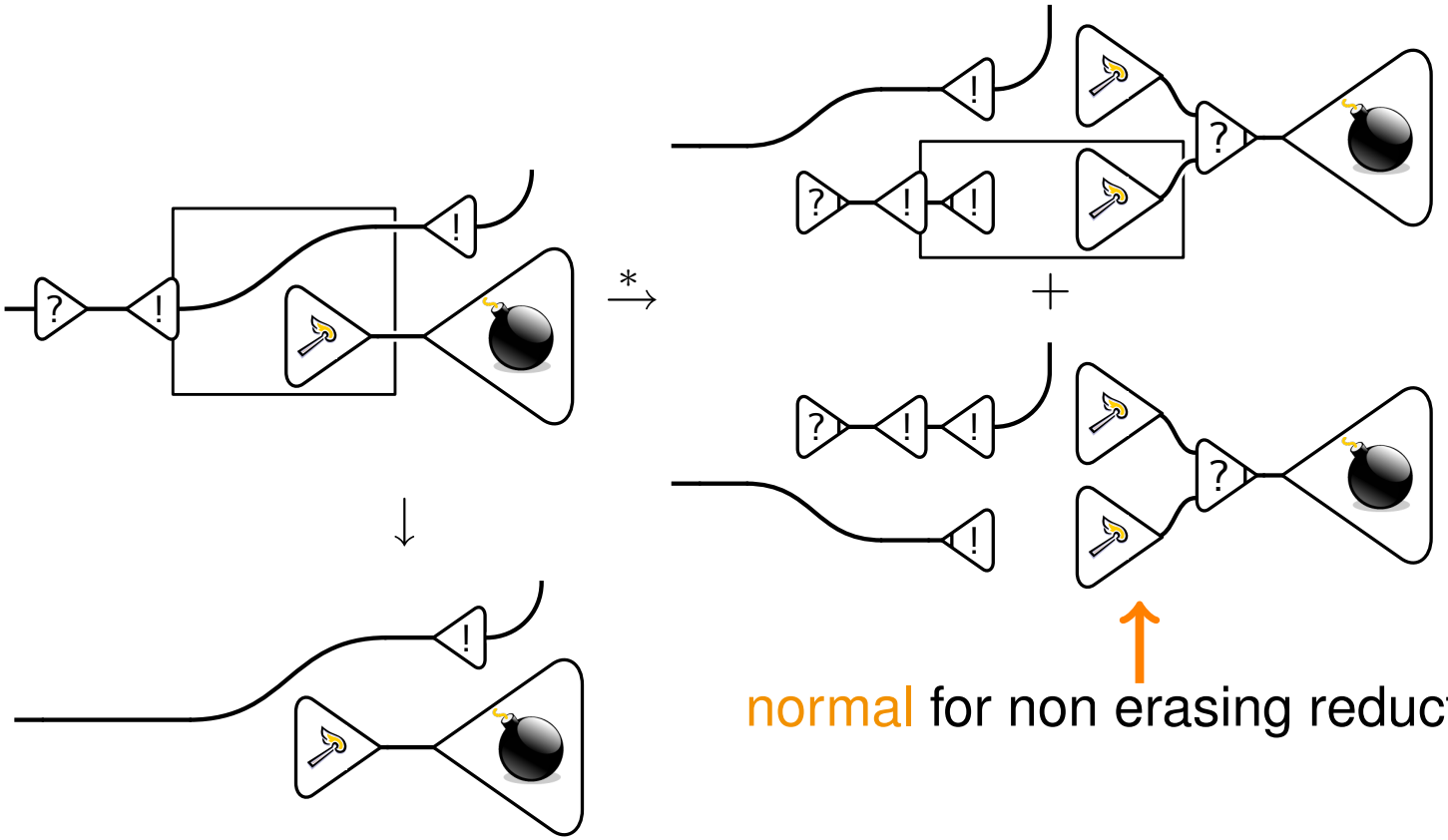
“Lazarus” clashes



normal for non erasing reduction

In LL Lazarus clashes do not negate standardization.

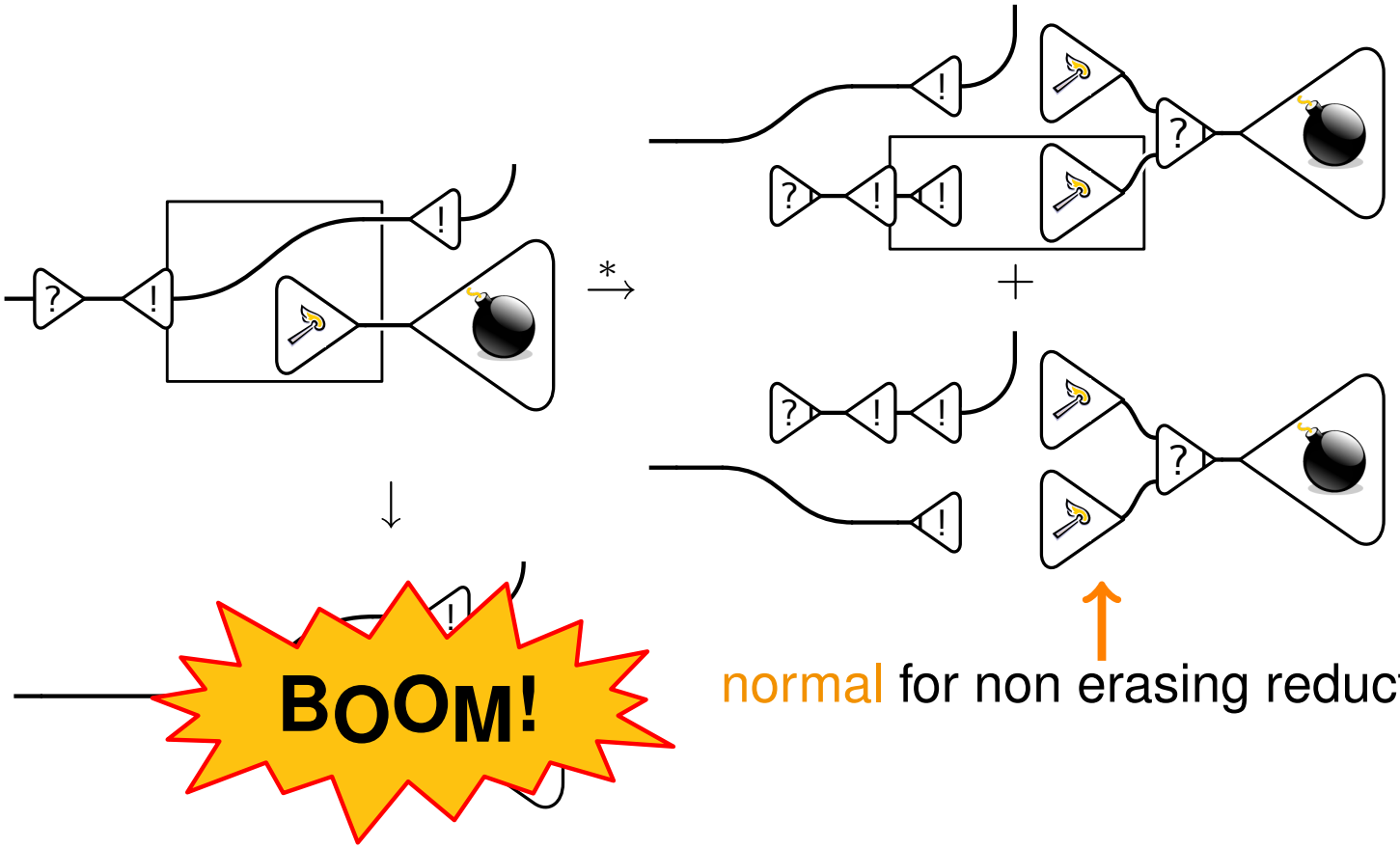
“Lazarus” clashes



normal for non erasing reduction

In LL Lazarus clashes do not negate standardization.

“Lazarus” clashes



normal for non erasing reduction

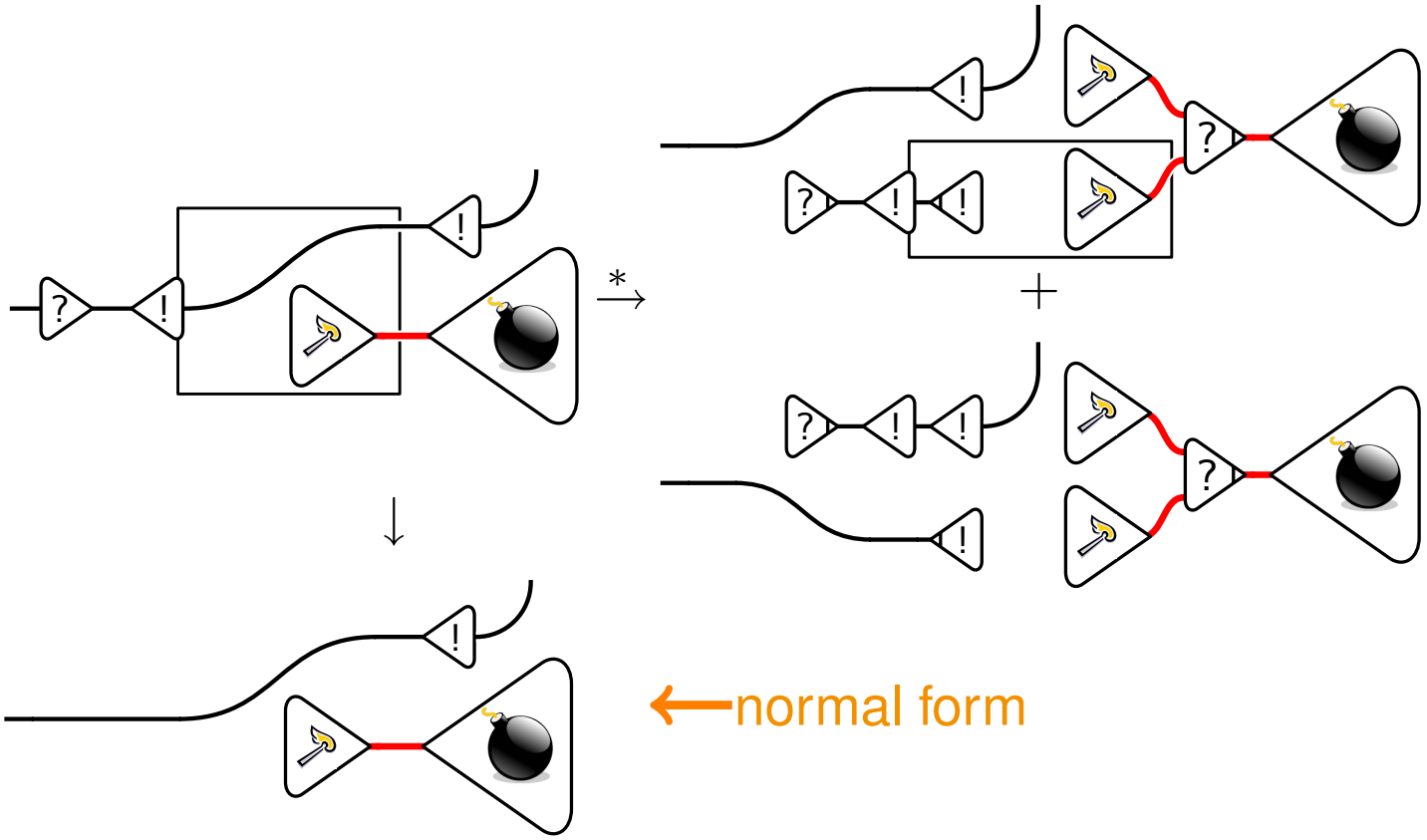
In LL Lazarus clashes do not negate standardization.

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.

Details

“Lazarus” clashes



What do we want?

Theorem

Reduction is strongly normalizing in the 2nd order typed case.

Theorem (standstrictcons?)

For π untyped switching acyclic LL net,
if $\pi \overset{\neg e}{\to} \pi'$, then $\pi' \in \mathrm{SN}_{\neg e} \implies \pi \in \mathrm{SN}_{\neg e}$.

$\overset{\neg e}{\to}$ stands for non erasing reduction. It follows that $\pi \in \mathrm{SN} \iff \pi \in \mathrm{WN}_{\neg e}$ (via $\mathrm{SN} \iff \mathrm{SN}_{\neg e}$).

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 got it!

Theorem

Reduction is strongly normalizing in the simply typed case.

Theorem (standstrictcons? with Pagani)

For π laxly typed switching acyclic DiLL net, if $\pi \xrightarrow{\neg e} \pi'$, then $\pi' \in \mathrm{SN}_{\neg e} \implies \pi \in \mathrm{SN}_{\neg e}$.

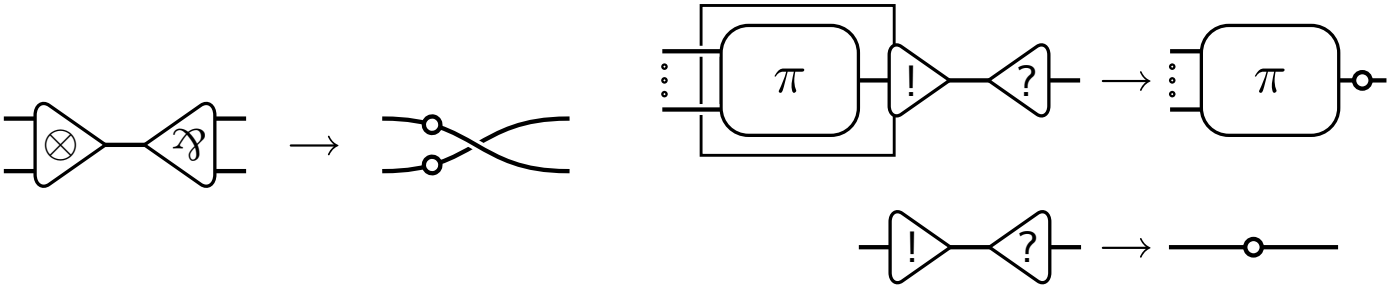
$\xrightarrow{\neg e}$ stands for non erasing reduction. It follows that $\pi \in \mathrm{SN} \iff \pi \in \mathrm{WN}_{\neg e}$ (via $\mathrm{SN} \iff \mathrm{SN}_{\neg e}$).

Normalization for simply typed nets recovered by WN (Pagani09)

Outline

Sketch of the proof of finite developments

Let DiLL$^\circ$ be the system with “new” cuts blocked.

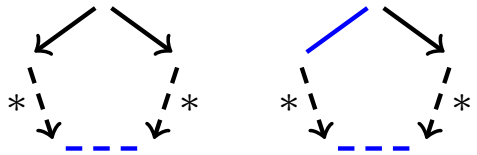


1. We define a measure:
   - Natural numbers on exponential cuts;
   - Multiset of natural numbers on polynets.
2. decreasing on exponential reduction of DiLL$^\circ$,
3. invariant under equivalences!

$\Longrightarrow$ finite developments modulo.

Church rosser modulo

1. We check for DiLL$^\circ$ local confluence and coherence modulo:



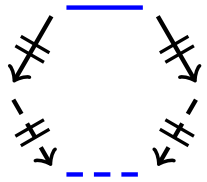
$$\Longrightarrow \text{ DiLL}^\circ \text{ is CR}\sim$$

(by a lemma by Huet generalizing Newman's one)

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

$$\rightarrow \subseteq \twoheadrightarrow \subseteq \xrightarrow{*} \implies \twoheadrightarrow^* = \xrightarrow{*} \quad \text{and} \quad (\twoheadleftarrow\!\!\twoheadrightarrow \cup \sim)^* = (\leftrightarrow \cup \sim)^*$$

3. DiLL$^\circ$ CR$\sim$ $\Longrightarrow$ $\twoheadrightarrow$ strongly confluent modulo

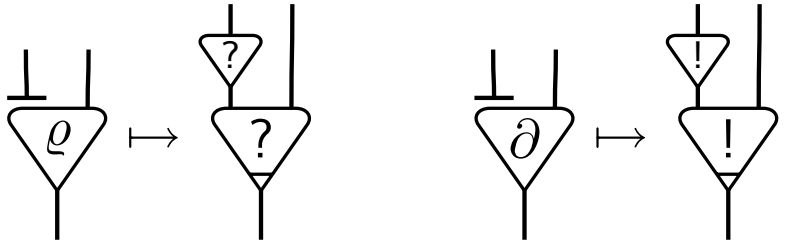


$$\Longrightarrow \text{ DiLL is CR}\sim$$

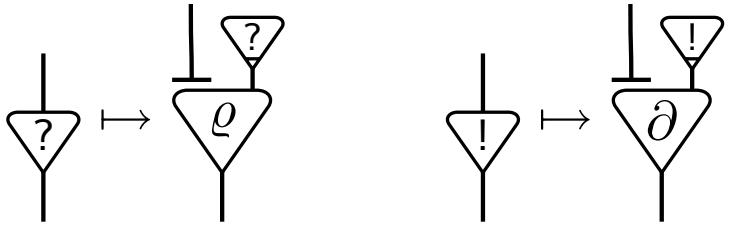
Standardization: $\mathrm{DiLL}_{\partial\varrho}$

As seen, non erasing reduction is not confluent in DiLL. We pass to another system, $\mathrm{DiLL}_{\partial\varrho}$.

From $\mathrm{DiLL}_{\partial\varrho}$ to DiLL:



From DiLL to $\mathrm{DiLL}_{\partial\varrho}$:

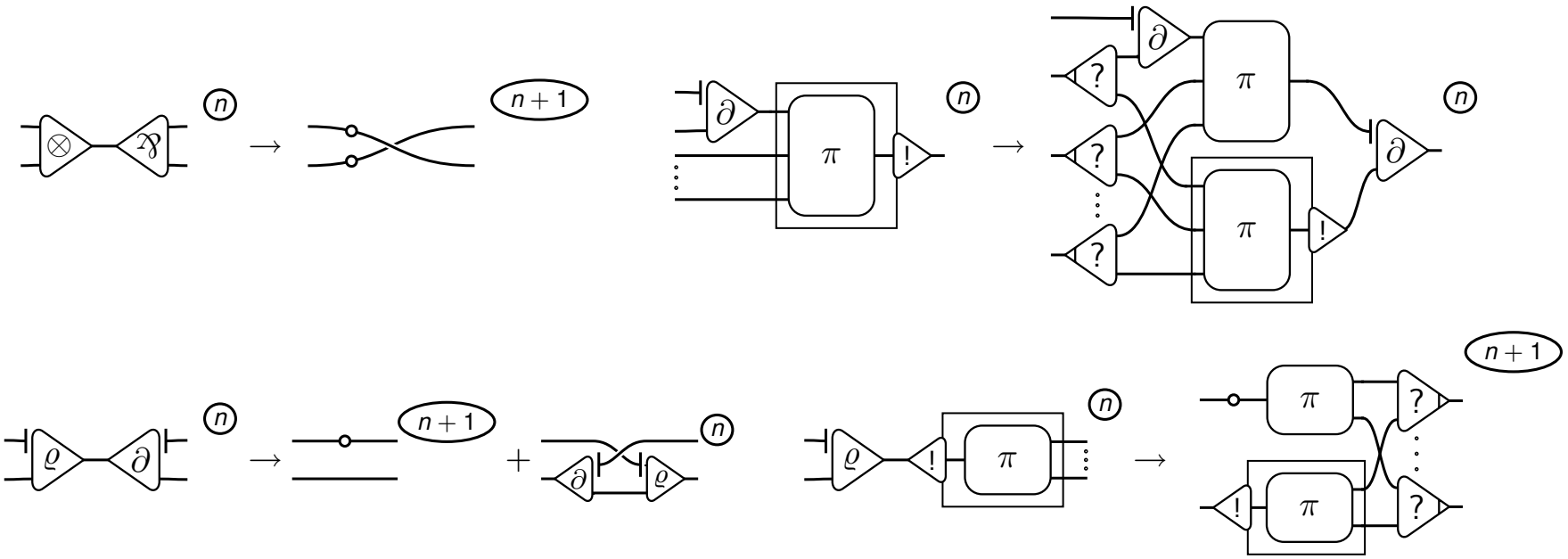


Finite developments $+$ new confluence and coherence diagrams

$\Longrightarrow$ non erasing reduction in $\mathrm{DiLL}_{\partial\varrho}$ is CR$\sim$.

Labelled DiLL$_{\partial\varrho}$

We further enrich the system with labels in $\mathbb{N}$



- The CR$\sim$ result is still valid for labelled reduciton, we are ready for Gandy's method!
- Define (using the labels) $\|\pi\|$ increasing on non erasing steps.
- If $\pi \in \mathrm{WN}_{\neg\mathrm{e}}$ then $\|\mathrm{NF}_{\neg\mathrm{e}}(\pi)\|$ bounds non erasing steps.

Standardization theorem

...then in $\mathsf{DiLL}_{\partial\varrho}$ $\mathsf{WN}_{\neg\mathsf{e}} \iff \mathsf{SN}\sim$ (as $\mathsf{SN}_{\neg\mathsf{e}} \iff \mathsf{SN}$).

Lemma

For π in DiLL*, if $\pi \in \mathsf{WN}_{\neg\mathsf{e}}$ then its translation $\pi^\diamond$ in $\mathsf{DiLL}_{\partial\varrho}$ is also.*

Not easy!

Then

$$\pi \in \mathsf{WN}_{\neg\mathsf{e}} \implies \pi^\diamond \in \mathsf{WN}_{\neg\mathsf{e}} \implies \pi^\diamond \in \mathsf{SN}\sim \implies \pi \in \mathsf{SN}\sim.$$

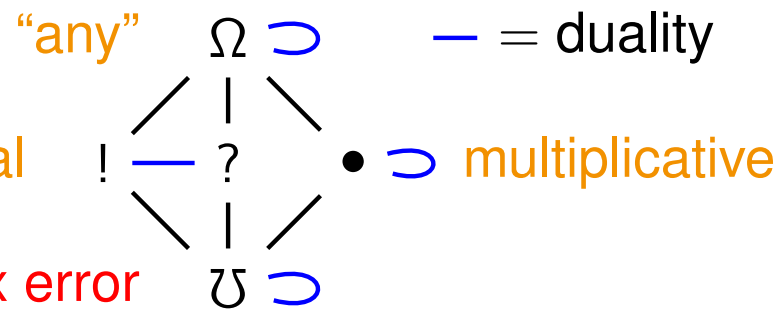
$$\implies \text{in DiLL } \pi \in \mathsf{WN}_{\neg\mathsf{e}} \iff \pi \in \mathsf{SN}\sim.$$

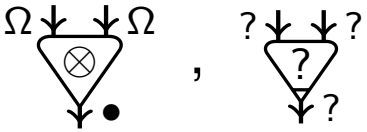
Thanks

Questions?

Lax typing

- 5 ordered types: exponential



- 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.

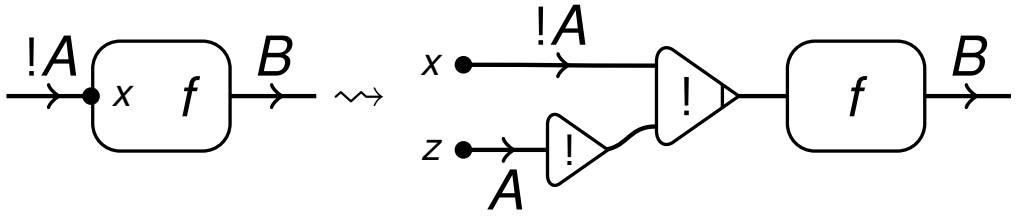
Back

General partial derivation

Analysis:

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

Nets:



Programs: linear substitution.

Back