Differential Nets

Paolo Tranquilli

paolo.tranquilli@ens-lyon.fr

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon

Workshop on Geometric and Logic Approaches to Computation, Nancy, 23-24/02/2010

The origins of Linear Logic

Top used reference:

Jean-Yves Girard.
Linear Logic.
Th. Comp. Sc., 50:1–102, 1987.

Proofs	Curry Howard ⟷	Programs
structural rules	⟷	duplication/erasing
no structural rules	⟷	linear programs

However the intuitions come from before...

Jean-Yves Girard.
Normal functors, power series, and λ-calculus.
Annals of Pure and Applied Logic, 2(37), 1986.

The origins of Linear Logic

Top used reference:

Jean-Yves Girard.
Linear Logic.
Th. Comp. Sc., 50:1–102, 1987.

Proofs	Curry Howard ⟷	Programs
structural rules	⟷	duplication/erasing
no structural rules	⟷	linear programs

However the intuitions come from before...

Jean-Yves Girard.
Normal functors, power series, and λ-calculus.
Annals of Pure and Applied Logic, 2(37), 1986.

The origins of Linear Logic

Top used reference:

Jean-Yves Girard.
Linear Logic.
Th. Comp. Sc., 50:1–102, 1987.

Proofs	Curry Howard ⟷	Programs
structural rules	⟷	duplication/erasing
no structural rules	⟷	linear programs

However the intuitions come from before…

Jean-Yves Girard.
Normal functors, power series, and λ-calculus.
Annals of Pure and Applied Logic, 2(37), 1986.

The origins of Linear Logic

Top used reference:

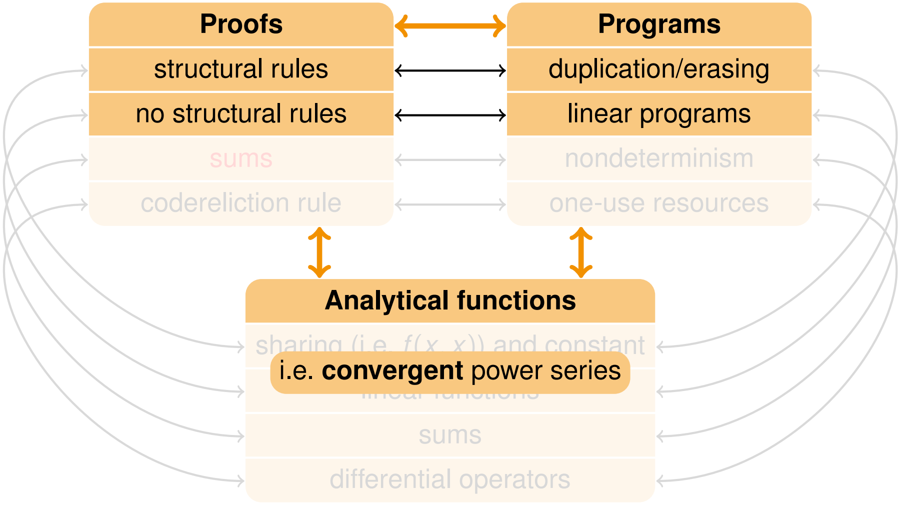
Jean-Yves Girard.
Linear Logic.
Th. Comp. Sc., 50:1–102, 1987.

Proofs	Curry Howard ⟷	Programs
structural rules	⟷	duplication/erasing
no structural rules	⟷	linear programs

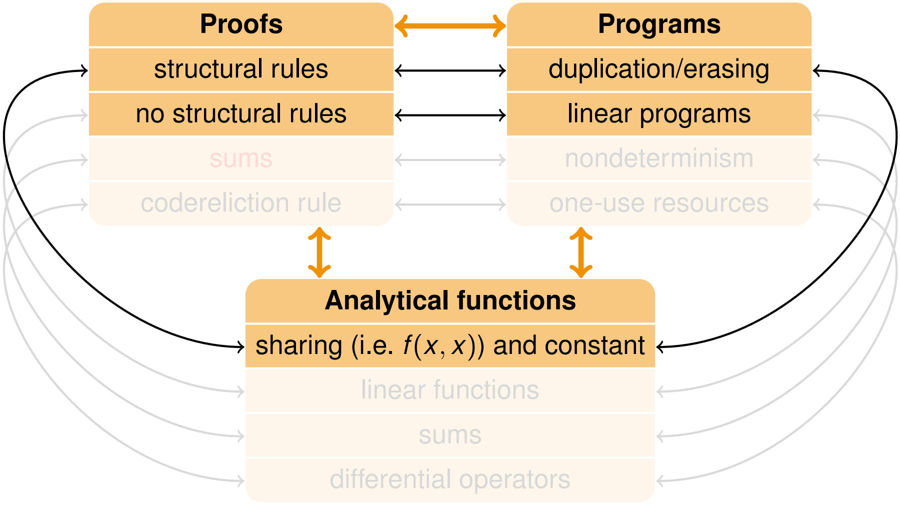
However the intuitions come from before…

Jean-Yves Girard.
Normal functors, power series, and λ-calculus.
Annals of Pure and Applied Logic, 2(37), 1986.

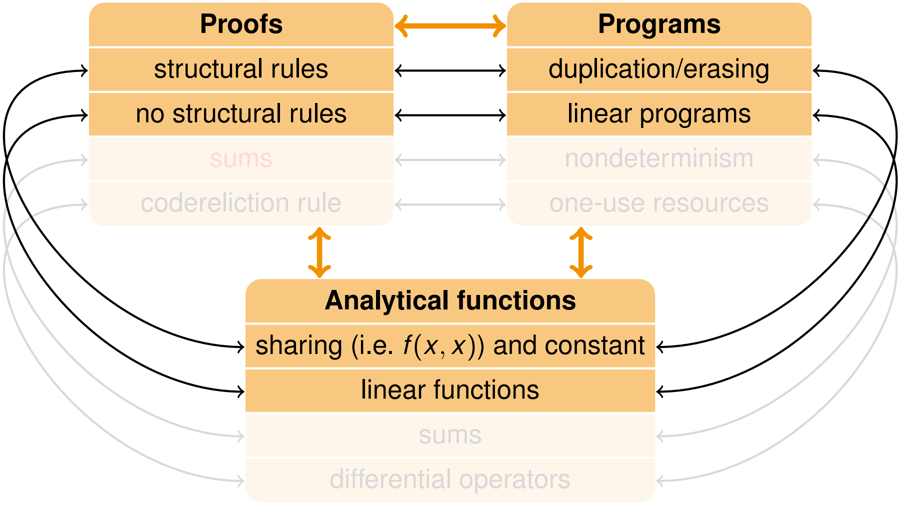
Proofs/programs as analytical functions



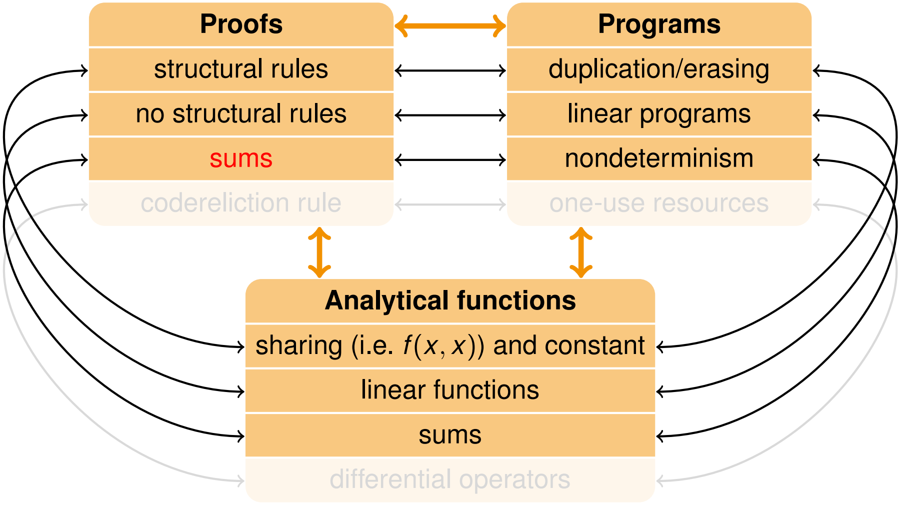
Proofs/programs as analytical functions



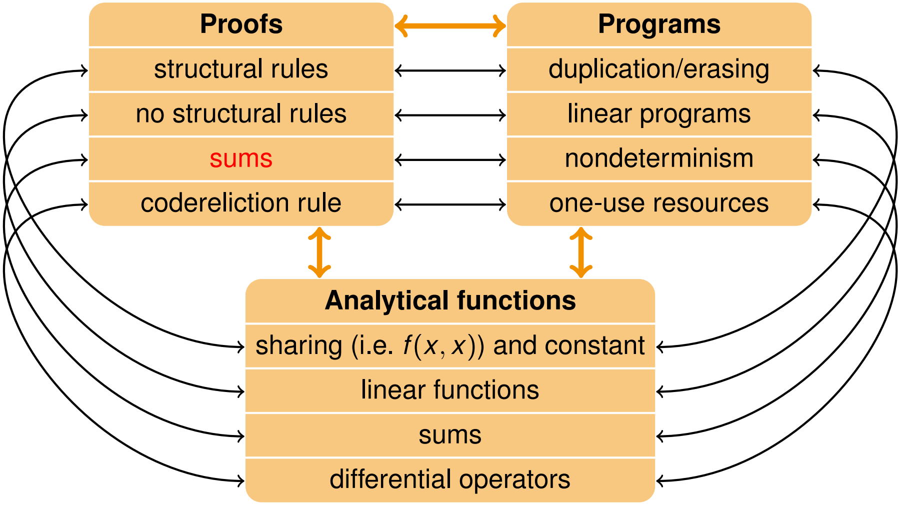
Proofs/programs as analytical functions



Proofs/programs as analytical functions



Proofs/programs as analytical functions



Rebirth of the quantitative approach

- Sums stray from logic: they are autodual, and 0 is a proof of anything.
- This led Girard to use quantitative semantics just as a blueprint $\rightsquigarrow$ coherent spaces and ultimately LL.
- Ehrhard unearthed the project, devising LL models with topological vector spaces.

Thomas Ehrhard. 2002
On Köthe sequence spaces and linear logic.

Thomas Ehrhard. 2005.
Finiteness spaces.

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

Thomas Ehrhard and Laurent Regnier. 2003.
The differential lambda-calculus.

Thomas Ehrhard and Laurent Regnier. 2006.
Differential interaction nets.

Rebirth of the quantitative approach

- Sums stray from logic: they are autodual, and 0 is a proof of anything.
- This led Girard to use quantitative semantics just as a blueprint $\rightsquigarrow$ coherent spaces and ultimately LL.
- Ehrhard unearthed the project, devising LL models with topological vector spaces.

Thomas Ehrhard. 2002
On Köthe sequence spaces and linear logic.

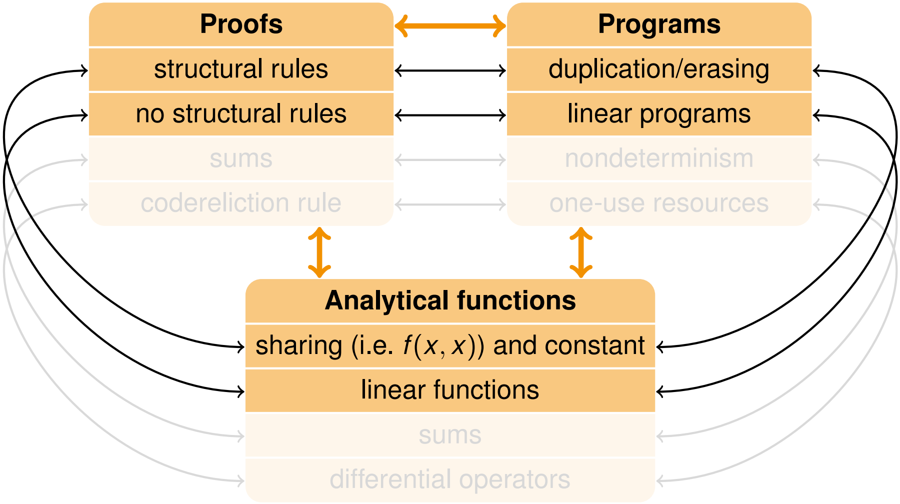
Thomas Ehrhard. 2005.
Finiteness spaces.

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

Thomas Ehrhard and Laurent Regnier. 2003.
The differential lambda-calculus.

Thomas Ehrhard and Laurent Regnier. 2006.
Differential interaction nets.

Ordinary proof nets



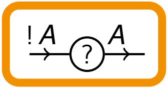
Proof nets as string diagrams

- Proof nets are a parallel syntax that abstracts away topologically commutation of rules, giving a neat account of cut-elimination.
- We can see them as string diagrams for monoidal categories with equality of morphisms $\leftrightsquigarrow$ topological equivalence.
- Linear category $\rightsquigarrow$
  monoidal comonad ! + isomorphism $!(A \,\&\, B) \cong\, !A \otimes !B + \ldots$
  $\rightsquigarrow$ further equalities of morphisms.
- Following will be a brief presentation of MELL proof nets with categorical and analysis intuitions.
- $A \multimap B \rightsquigarrow$ linear functions, $!A \multimap B \rightsquigarrow$ analytical functions.

Dereliction

!'s counit:

$\epsilon : !A \rightarrow A$

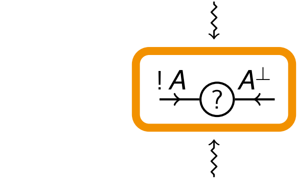


the identity x seen as analytical

Dereliction

!'s counit:

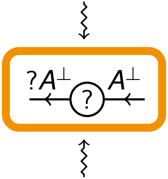
$$\epsilon : !A \to A$$



the identity x seen as analytical

!'s counit:

$$\epsilon : !A \rightarrow A$$

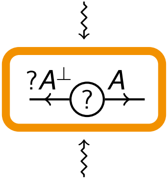


the identity x seen as analytical

Dereliction

!'s counit:

$$\epsilon : !A \to A$$

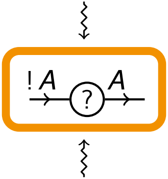


the identity x seen as analytical

Dereliction

!'s counit:

$$\epsilon : !A \to A$$



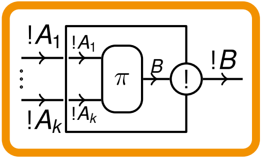
the identity x seen as analytical

Exponential box

!'s functoriality, multiplication (digging) and monoidalness:

$$\delta : !A \to !!A \qquad m : !A \otimes !B \to !(A \otimes B) \qquad \frac{\sigma : A \to B}{!\sigma : !A \to !B}$$

$$\pi^! : \bigotimes_i !A_i \xrightarrow{\otimes\delta} \bigotimes_i !!A_i \xrightarrow{m} !\left(\bigotimes_i !A_i\right) \xrightarrow{!\pi} !B$$



composition of analytical f's is analytical
box packages function to be ready for plugging

Structural rules

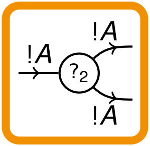
!'s functoriality, diagonal, terminal object, exponential isomorphism:

$$\Delta : A \to A \,\&\, A$$

$$\rightsquigarrow$$

$$d : !A \xrightarrow{!\Delta} !(A \,\&\, A) \cong !A \otimes !A$$

$$\rightsquigarrow$$



$$\leftrightsquigarrow$$

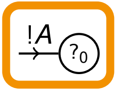
$f(x, y)$ analyt. $\implies$ $f(x, x)$ analyt.

$$T : A \to \top$$

$$\rightsquigarrow$$

$$e : !A \xrightarrow{!T} !\top \cong 1$$

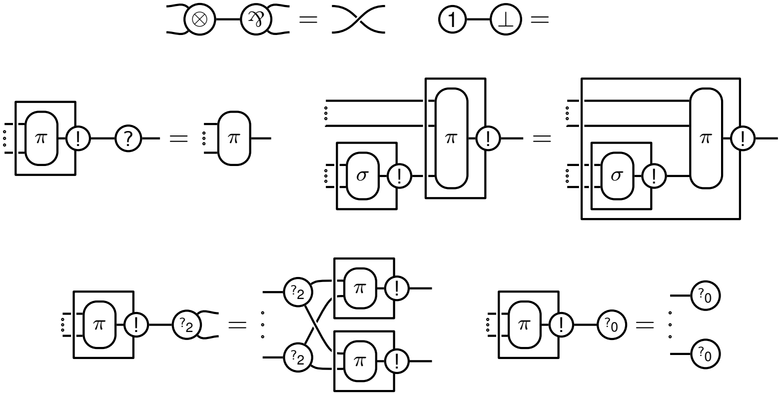
$$\rightsquigarrow$$



$$\leftrightsquigarrow$$

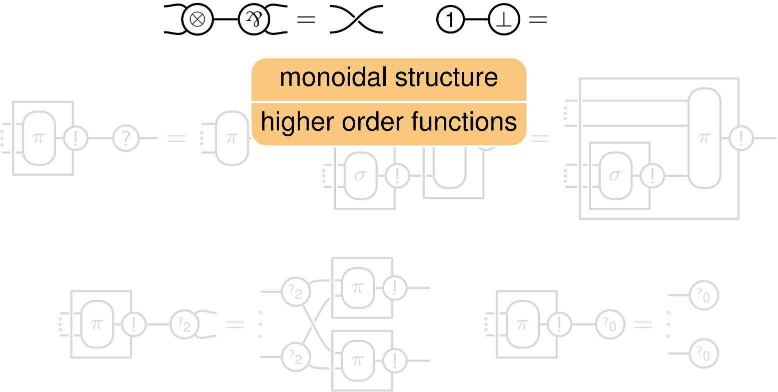
constant is analyt.

Contraction and weakening make $!A$ a comonoid.



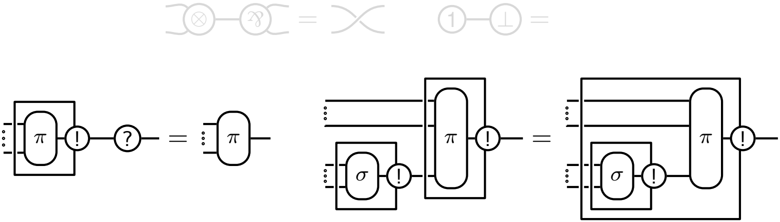
from string diagrams. . . to interaction nets!

From equalities. . . to rewriting



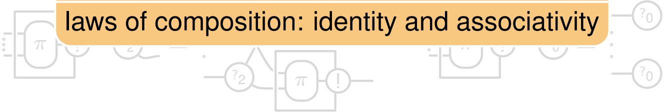
from string diagrams. . . to interaction nets!

From equalities. . . to rewriting



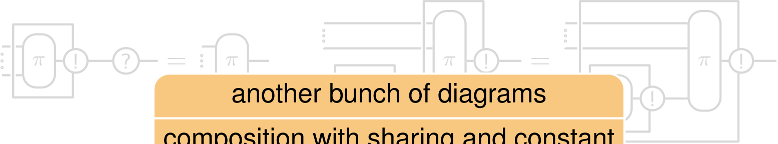
lots of diagrams

laws of composition: identity and associativity



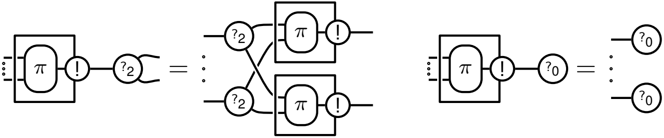
from string diagrams. . . to interaction nets!

From equalities. . . to rewriting



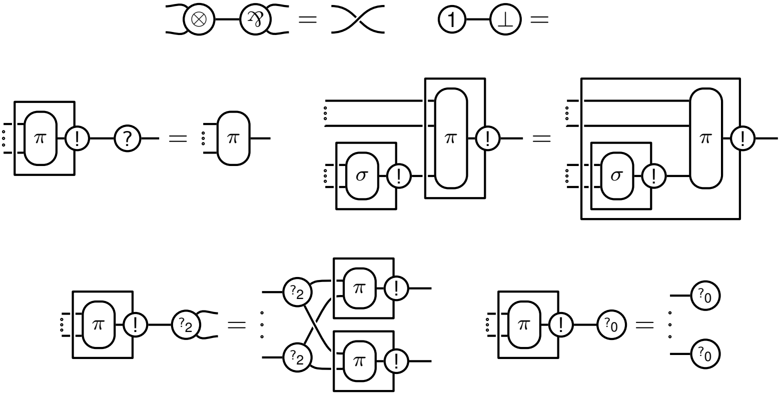
another bunch of diagrams

composition with sharing and constant



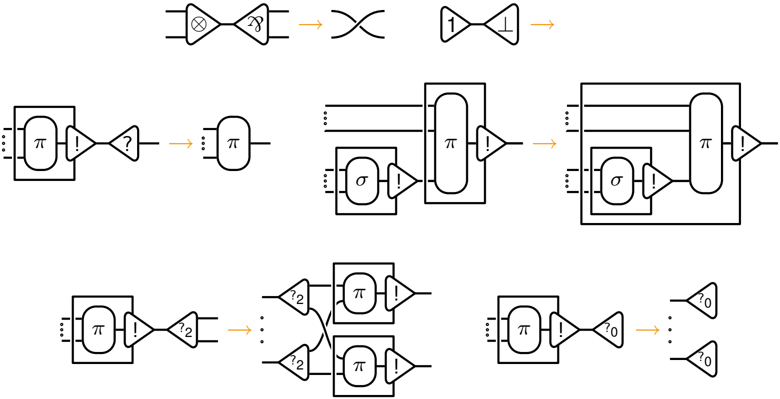
from string diagrams. . . to interaction nets!

From equalities. . . to rewriting



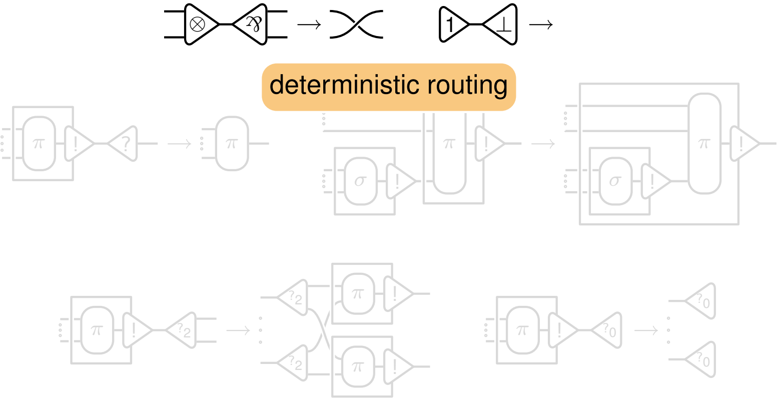
from string diagrams. . . to interaction nets!

From equalities. . . to rewriting



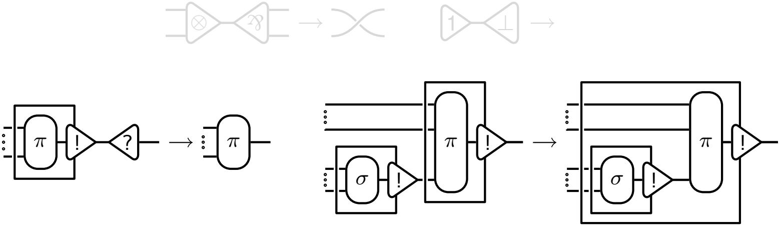
from string diagrams. . . to interaction nets!

From equalities. . . to rewriting



from string diagrams. . . to interaction nets!

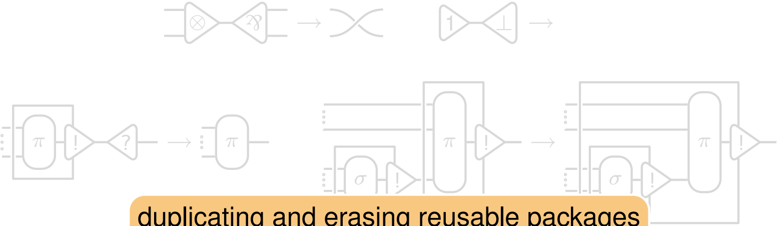
From equalities. . . to rewriting



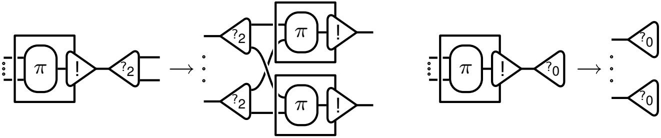
opening and fusing reusable packages

from string diagrams. . . to interaction nets!

From equalities. . . to rewriting

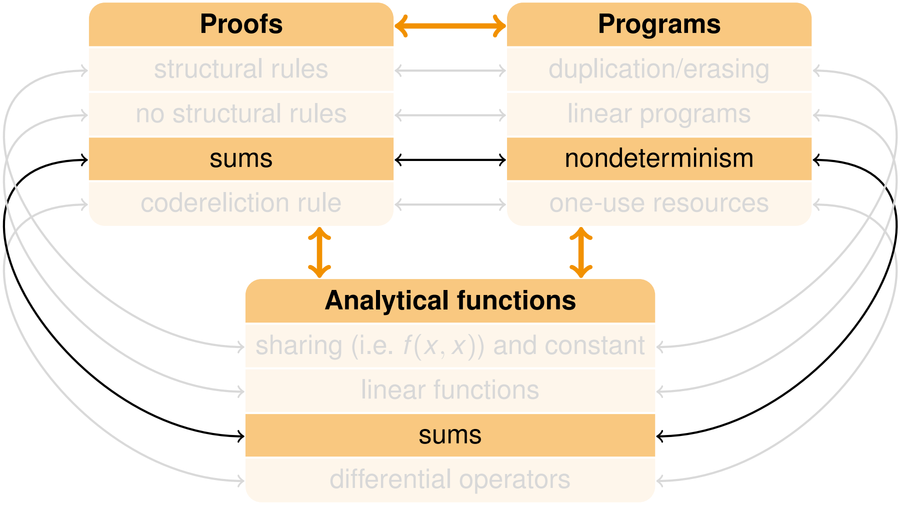


duplicating and erasing reusable packages



from string diagrams. . . to interaction nets!

Adding nondeterminism



Sums from biproducts

- What happens if product & and coproduct $\oplus$ are equal?
- We have the biproduct $\ast$ (and $0 = \top$).
- Then morphisms $A \to B$ get a commutative monoid structure: in short, can be summed.

$$\pi + \sigma : A \xrightarrow{\Delta} A \ast A \xrightarrow{\pi \ast \sigma} B \ast B \xrightarrow{\nabla} B, \qquad 0 : A \to \top \to B.$$

- Sum distributes on composition:

$$(\textstyle\sum_i \pi_i); (\sum_j \sigma_j) = \sum_{ij} (\pi_i; \sigma_j),$$

as morphisms are linear!

Sums from biproducts

- What happens if product & and coproduct $\oplus$ are equal?
- We have the biproduct $\ast$ (and $0 = \top$).
- Then morphisms $A \to B$ get a commutative monoid structure: in short, can be summed.

$$\pi + \sigma : A \xrightarrow{\Delta} A \ast A \xrightarrow{\pi \ast \sigma} B \ast B \xrightarrow{\nabla} B, \qquad 0 : A \to \top \to B.$$

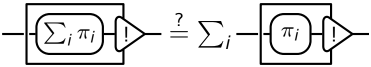
- Sum distributes on composition:

$$-\boxed{\sum_i \pi_i}-\boxed{\sum_j \sigma_j}- \; = \sum_{ij} -\boxed{\pi_i}-\boxed{\sigma_j}-$$

as morphisms are linear!

Sums, nondeterminism, boxes

- Computationally, $\pi + \sigma$ can be viewed as internal choice between the two, or as independent parallel computations.
- What about boxes?

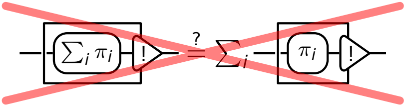


- Application of analytical functions/ordinary programs is linear in the function, but not in the argument:

$$(\lambda x.M)(N_1 + N_2) \neq (\lambda x.M)N_1 + (\lambda x.M)N_2.$$

Sums, nondeterminism, boxes

- Computationally, $\pi + \sigma$ can be viewed as internal choice between the two, or as independent parallel computations.
- What about boxes?



- Application of analytical functions/ordinary programs is linear in the function, but not in the argument:

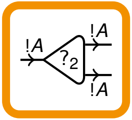
$$(\lambda x.M)(N_1 + N_2) \neq (\lambda x.M)N_1 + (\lambda x.M)N_2.$$

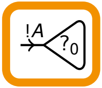
Costructural rules

!'s functoriality, codiagonal, coterminal object, exponential isomorphism:

$$\Delta : A \to A \ast A \quad\quad A \to \top$$

$$d : !A \xrightarrow{!\Delta} !(A \ast A) \cong !A \otimes !A \quad\quad e : !A \to !\top \cong 1$$





$f(x, y)$ analyt. $\Longrightarrow$ $f(x, x)$ analyt. constant is analyt.

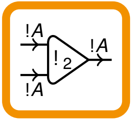
Cocontraction and coweakening make $!A$ a comonoid.

Together they make $!A$ a bialgebra.

!'s functoriality, codiagonal, coterminal object, exponential isomorphism:

$$\nabla : A \ast A \to A$$

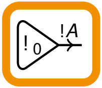
$$m : !A \otimes !A \cong !(A \ast A) \xrightarrow{!\nabla} !A$$



$f(x)$ analyt. $\Longrightarrow$ $f(x+y)$ analyt.

$$\top \to A$$

$$u : 1 \cong !\top \to !A$$



$f(x)$ analyt. $\Longrightarrow$ $f(0)$ analyt.

Cocontraction and coweakening make $!A$ a monoid.

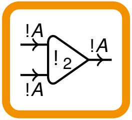
Together they make $!A$ a bialgebra.

Costructural rules

!'s functoriality, codiagonal, coterminal object, exponential isomorphism:

$$\nabla : A \ast A \to A$$

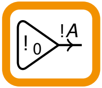
$$m : !A \otimes !A \cong !(A \ast A) \xrightarrow{!\nabla} !A$$



$f(x)$ analyt. $\implies$ $f(x+y)$ analyt.

$$\top \to A$$

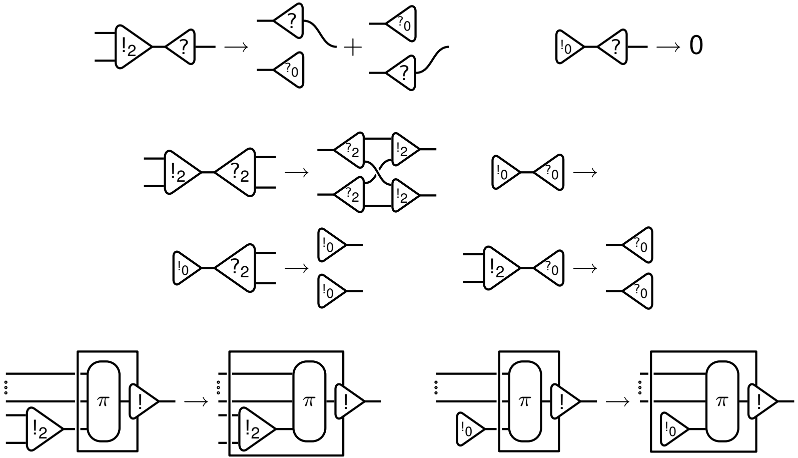
$$u : 1 \cong !\top \to !A$$



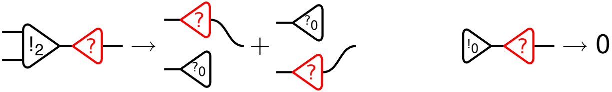
$f(x)$ analyt. $\implies$ $f(0)$ analyt.

Cocontraction and coweakening make $!A$ a monoid.
Together they make $!A$ a bialgebra.

New equalities/reductions



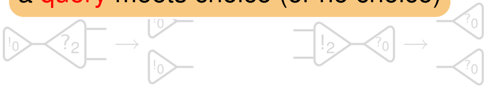
New equalities/reductions



by definition (+ diagrams. . .)

$x + y$ and 0 as analytic functions

a query meets choice (or no choice)

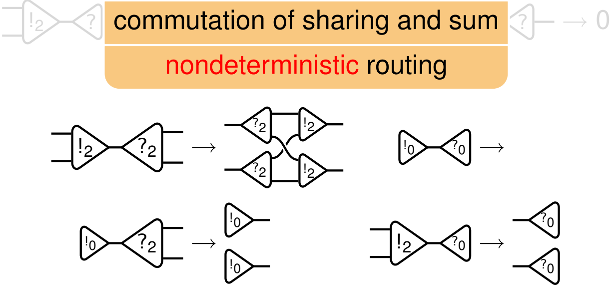


New equalities/reductions

bialgebraic structure of $!A$

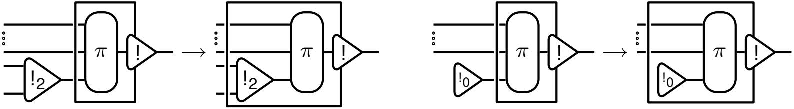
commutation of sharing and sum

nondeterministic routing

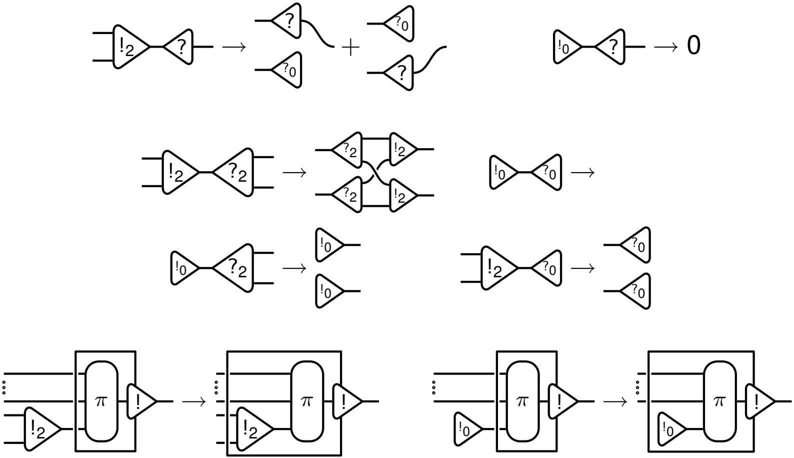


New equalities/reductions

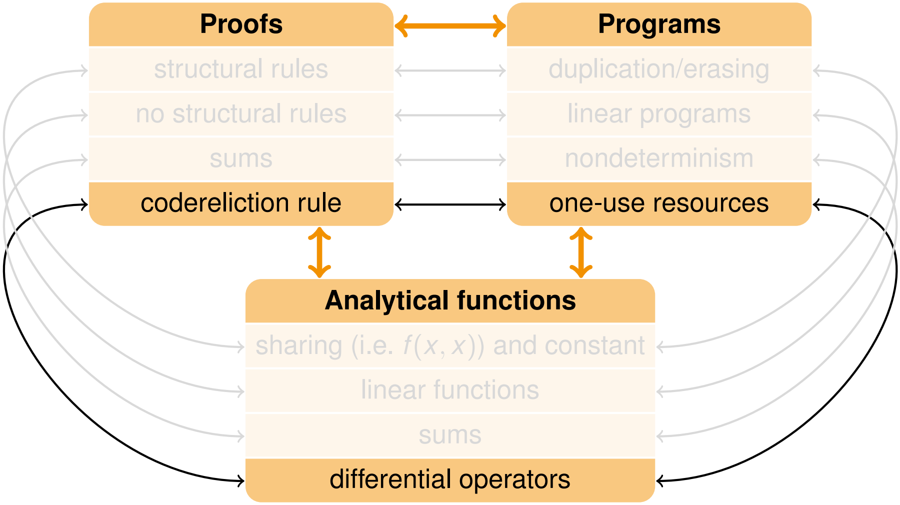
cocontraction and coweakening are as boxes



New equalities/reductions



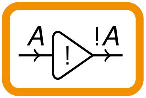
Adding derivation



Codereliction

Symmetric to dereliction:

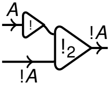
$$\eta : A \to !A$$



turning $f(x)$ into the linear map $\left.\frac{\partial f}{\partial x}\right|_{x=0}$.

Derivation in 0 is all that's needed:

$$\frac{\partial f}{\partial x} = \left.\frac{\partial f(y+x)}{\partial y}\right|_{y=0}, \text{ i.e. }$$



.

Derivation in computation

Q: What is the derivative of a function?

A: The best linear aproximation.

Q: What is the derivative of a program?

A: The best linear aproximation!

i.e. the (nondeterministic) aproximation using its input exactly once

One-use resources!

can go where

can, but won't be duplicated.

(or erased)

Derivation in computation

Q: What is the derivative of a function?

A: The best linear aproximation.

Q: What is the derivative of a program?

A: The best linear aproximation!

i.e. the (nondeterministic) aproximation using its input exactly once

One-use resources!

can go where

can, but won't be duplicated.

(or erased)

Derivation in computation

Q: What is the derivative of a function?

A: The best linear aproximation.

Q: What is the derivative of a program?

A: The best linear aproximation!

i.e. the (nondeterministic) aproximation using its input exactly once

⇝

One-use resources!

can go where

can, but won't be duplicated.

(or erased)

Derivation in computation

Q: What is the derivative of a function?

A: The best linear aproximation.

Q: What is the derivative of a program?

A: The best linear aproximation!

i.e. the (nondeterministic) aproximation using its input exactly once

⇝

One-use resources!

can go where

can, but won't be duplicated.

(or erased)

Derivation in computation

Q: What is the derivative of a function?

A: The best linear aproximation.

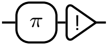
Q: What is the derivative of a program?

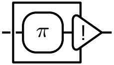
A: The best linear aproximation!

i.e. the (nondeterministic) aproximation using its input exactly once

$\rightsquigarrow$

One-use resources!

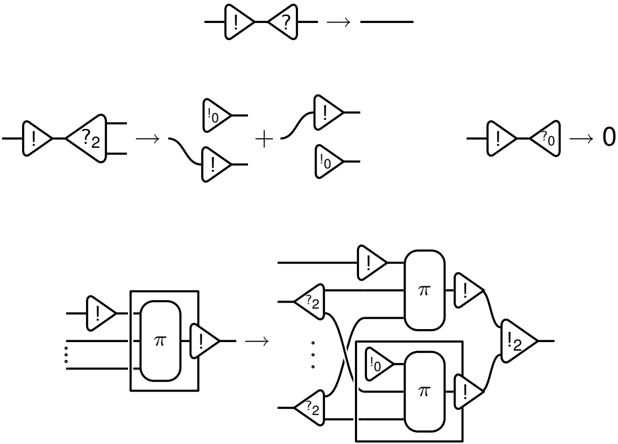


can go where


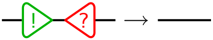
can, but won't be duplicated.

(or erased)

The new reductions



The new reductions

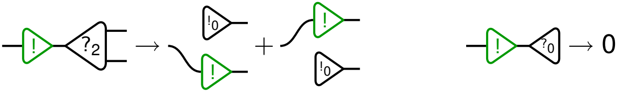


codereliction is right inverse of dereliction

$$\frac{\partial x}{\partial x}\Big|_{x=0} = \mathrm{id}$$

a query meets a one-use resource and is answered

The new reductions



...

$$\frac{\partial f(x,x)}{\partial x}\Big|_{x=0} = \frac{\partial f(x,0)}{\partial x}\Big|_{x=0} + \frac{\partial f(0,x)}{\partial x}\Big|_{x=0}, \qquad \frac{\partial C}{\partial x} = 0$$

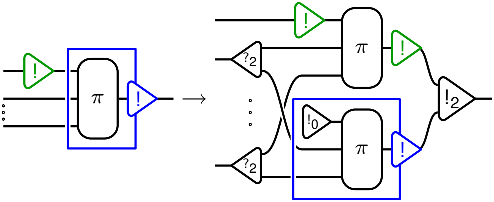
a one-use resource is contended by multiple (or no) queries

The new reductions

. . .

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

a one-use resource is asked by a reusable one,
of which exactly one copy gets it

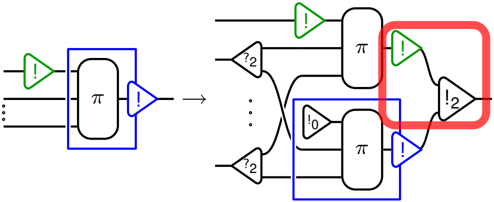


The new reductions

...

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

a one-use resource is asked by a reusable one, of which exactly one copy gets it

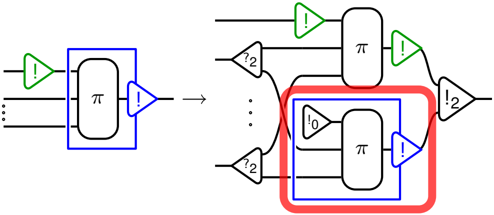


The new reductions

…

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

a one-use resource is asked by a reusable one,
of which exactly one copy gets it

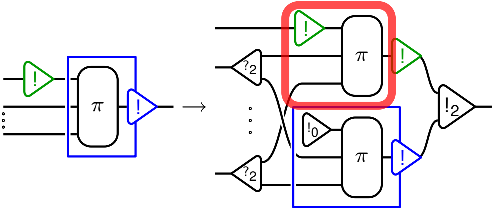


The new reductions

...

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

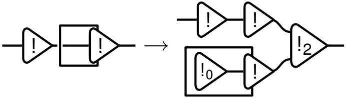
a one-use resource is asked by a reusable one,
of which exactly one copy gets it

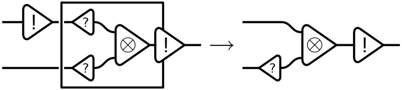


Codereliction and categorical models

- The diagrams for codereliction are a little messy.
- For example with a bit of “reverse engineering”:

Interaction with digging: Interaction with !’s monoidalness:





- Also symmetry is not complete: in concrete and categorical models so far there is no codigging, i.e. ! is not also a monad.

Richard Blute, J. Robin B. Cockett, and R. A. G. Seely.
Differential categories.
Mathematical Structures in Computer Science, 16(6):1049–1083, 2006.

Marcelo Fiore.
Differential structure in models of multiplicative biadditive intuitionistic linear logic.
In *TLCA*, volume 4583 of *LNCS*, pages 163–177. Springer, 2007.

Are differential nets a good rewriting system?

Termination?

Do all nondeterministic branches terminate?

Confluence?

Is nondeterminism truly internal?
Do the possible outcomes depend on the reduction strategy?

Yes!

	FD	CR	Cons.	Stand.	WN	SN
Untyped DiLL						
Second order DiLL						
Propositional DiLL						

FD : finite developments
CR : Church Rosser
Cons. : conservation (non erasing reductions preserve inifinite ones)
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.

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

[PaTr09] Michele Pagani, P. T.
The conservation theorem for differential nets with promotion.
Submitted for publication.

skipping a fair bit about:

- correctness
- equivalences

Yes!

	FD	CR	Cons.	Stand.	WN	SN
Untyped DiLL	[Tr09]	[Tr09]	[PaTr09]	[PaTr09]	No	No
Second order DiLL	⇓ Yes	⇓ Yes	⇓ Yes	⇓ Yes	?	?
Propositional DiLL	⇓ Yes	⇓ Yes	⇓ Yes	⇓ Yes	[Pa09]	[PaTr09]

FD : finite developments
CR : Church Rosser
Cons. : conservation (non erasing reductions preserve inifinite ones)
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.
Submitted for publication.

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

skipping a fair bit about:

- correctness
- equivalences

Yes!

	FD	CR	Cons.	Stand.	WN	SN
Untyped DiLL	[Tr09]	[Tr09]	[PaTr09]	[PaTr09]	No	No
Second order DiLL	⇓ Yes	⇓ Yes	⇓ Yes	⇓ Yes	?	?
Propositional DiLL	⇓ Yes	⇓ Yes	⇓ Yes	⇓ Yes	[Pa09]	[PaTr09]

FD : finite developments
CR : Church Rosser
Cons. : conservation (non erasing reductions preserve inifinite ones)
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.

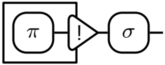
[PaTr09] Michele Pagani, P. T.
The conservation theorem for differential nets with promotion.
Submitted for publication.

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

skipping a fair bit about:

- correctness
- equivalences

Iterated derivation

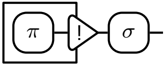
- 

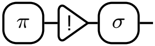
represents $\sigma(\pi)$;
-

represents $\left.\frac{\partial\sigma}{\partial x}\right|_{x=0} \cdot \pi$;
-

represents $\left.\frac{\partial^n\sigma}{\partial x^n}\right|_{x=0} \cdot (\pi, \ldots, \pi)$;
- … so we can Taylor-expand!

Iterated derivation

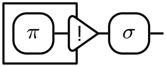
- 

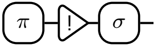
represents $\sigma(\pi)$;
- 

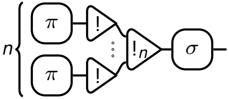
represents $\left.\frac{\partial \sigma}{\partial x}\right|_{x=0} \cdot \pi$;
-

represents $\left.\frac{\partial^n \sigma}{\partial x^n}\right|_{x=0} \cdot (\pi, \ldots, \pi)$;
- … so we can Taylor-expand!

Iterated derivation

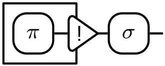
- 

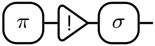
represents $\sigma(\pi)$;
- 

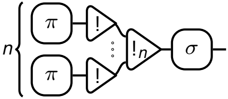
represents $\left.\frac{\partial \sigma}{\partial x}\right|_{x=0} \cdot \pi$;
- 

represents $\left.\frac{\partial^n \sigma}{\partial x^n}\right|_{x=0} \cdot (\pi, \dots, \pi)$;
- … so we can Taylor-expand!

Iterated derivation

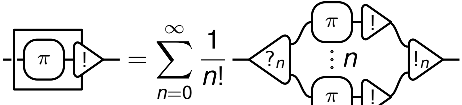
- 

represents $\sigma(\pi)$;
- 

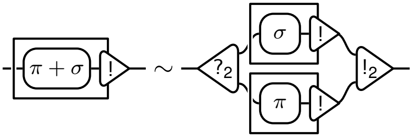
represents $\left.\frac{\partial\sigma}{\partial x}\right|_{x=0}\cdot\pi$;
- 

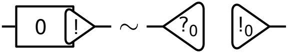
represents $\left.\frac{\partial^n\sigma}{\partial x^n}\right|_{x=0}\cdot(\pi,\ldots,\pi)$;
- … so we can Taylor-expand!

The Taylor expansion



- Taylor expansion converts any net (even pure, non terminating ones) into an infinite sum of finite objects.
- Halfway between syntax and semantics.
- The formula also tells that the exponential box is an exponential. In particular (compare with $\mathrm{e}^{x+y} = \mathrm{e}^x \cdot \mathrm{e}^y$):





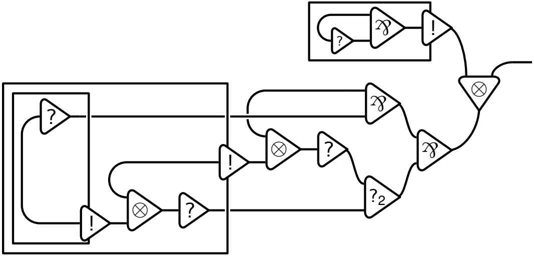
Proof nets and λ-calculus

The translation of intuitionistic logic based on $A \Rightarrow B \cong\ !A \multimap B$ extends to λ-calculus (coKliesli of linear category $\rightsquigarrow$ CCC).

For example:

$$(\lambda f, x.(f(fx)))\lambda z.z$$

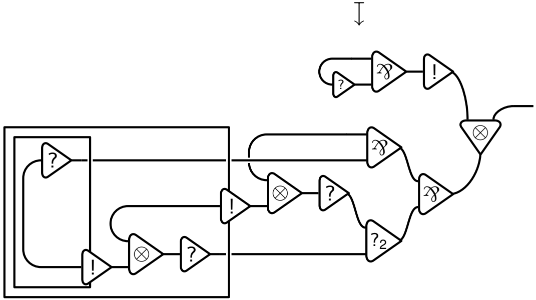
$$\downarrow$$



And if we switch to differential nets?

Resource calculus (more on it in Simona's talk):

$$(\lambda f, x.(f(fx)))[\lambda z.z] \longrightarrow 0$$

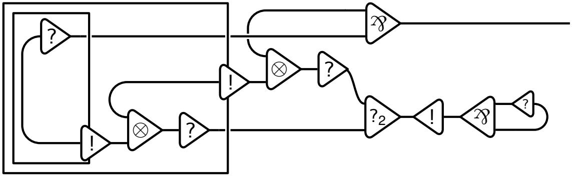


And if we switch to differential nets?

Resource calculus (more on it in Simona's talk):

$$(\lambda f, x.(f(fx)))[\lambda z.z] \to 0$$

$$\downarrow$$

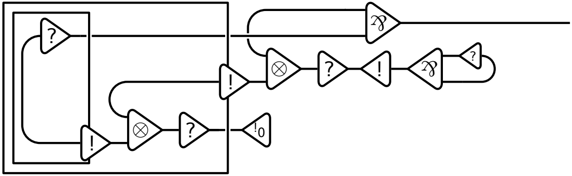


And if we switch to differential nets?

Resource calculus (more on it in Simona's talk):

$$(\lambda f, x.(f(fx)))[\lambda z.z] \rightarrow 0$$

$$\downarrow$$

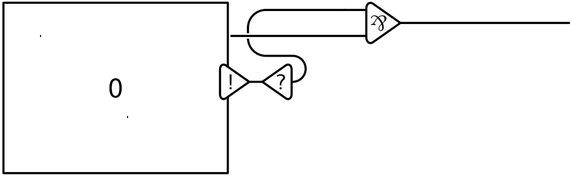


And if we switch to differential nets?

Resource calculus (more on it in Simona's talk):

$$(\lambda f, x.(f(fx)))[\lambda z.z] \to 0$$

$\updownarrow$



And if we switch to differential nets?

Resource calculus (more on it in Simona's talk):

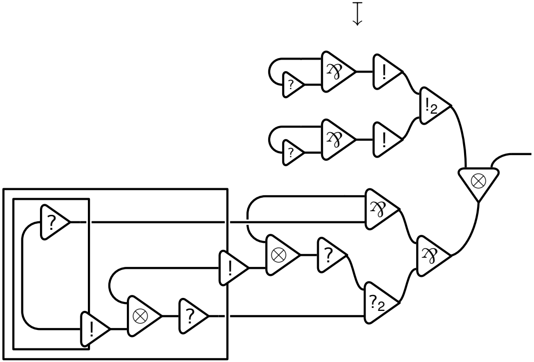
$$(\lambda f, x.(f(fx)))[\lambda z.z] \to 0$$
$$\downarrow$$

0

And if we switch to differential nets?

Resource calculus (more on it in Simona's talk):

$$(\lambda f, x.(f(fx)))[\lambda z.z, \lambda z.z] \rightarrow 2 \cdot \lambda z.z$$

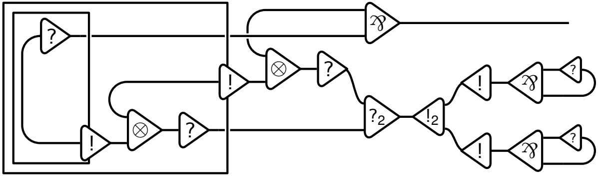


And if we switch to differential nets?

Resource calculus (more on it in Simona's talk):

$$(\lambda f, x.(f(fx)))[\lambda z.z, \lambda z.z] \to 2 \cdot \lambda z.z$$

$$\downarrow$$



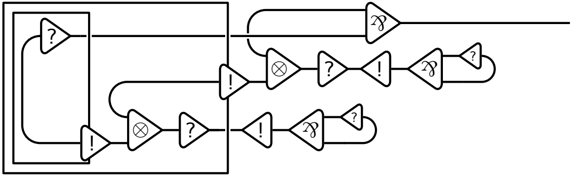
And if we switch to differential nets?

Resource calculus (more on it in Simona's talk):

$$(\lambda f, x.(f(fx)))[\lambda z.z, \lambda z.z] \to 2 \cdot \lambda z.z$$

$$\downarrow$$

2 ·



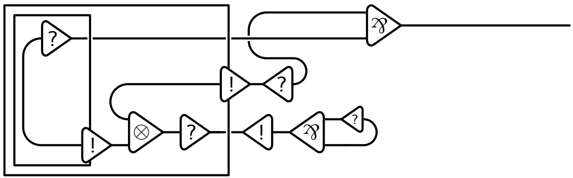
And if we switch to differential nets?

Resource calculus (more on it in Simona's talk):

$$(\lambda f, x.(f(fx)))[\lambda z.z, \lambda z.z] \rightarrow 2 \cdot \lambda z.z$$

$\updownarrow$

2 ·



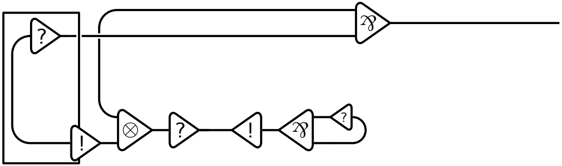
And if we switch to differential nets?

Resource calculus (more on it in Simona's talk):

$$(\lambda f, x.(f(fx)))[\lambda z.z, \lambda z.z] \to 2 \cdot \lambda z.z$$

$$\downarrow$$

2 ·



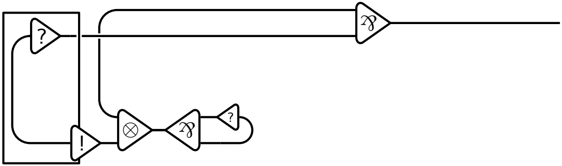
And if we switch to differential nets?

Resource calculus (more on it in Simona's talk):

$$(\lambda f, x.(f(fx)))[\lambda z.z, \lambda z.z] \to 2 \cdot \lambda z.z$$

$$\downarrow$$

2 ·

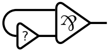


And if we switch to differential nets?

Resource calculus (more on it in Simona's talk):

$$(\lambda f, x.(f(fx)))[\lambda z.z, \lambda z.z] \to 2 \cdot \lambda z.z$$

$$\updownarrow$$

$2 \cdot$ 

And now, maybe something on the whiteboard!

And questions!