An exegesis of transcendental syntax

PhD defense

LIPN – Université Sorbonne Paris Nord

Boris Eng

Logic with proof systems

Notion of proof. I want to prove C. I assume $A, B, \dots$. I conclude C.

Logic with proof systems

Notion of proof. I want to prove C. I assume $A, B, \ldots$. I conclude C.

How are proofs formalised ?

Logic with proof systems

Notion of proof. I want to prove C. I assume $A, B, \dots$. I conclude C.

How are proofs formalised ?

Proof systems with inference rules and sequents

$$\text{hypotheses} \vdash \text{conclusion} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

Logic with proof systems

Notion of proof. I want to prove C. I assume $A, B, \ldots$. I conclude C.

How are proofs formalised ?

Proof systems with inference rules and sequents

$$\text{hypotheses} \vdash \text{conclusion} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

Proof trees

$$\cfrac{\cfrac{\cfrac{}{\Gamma, A, B \vdash A}\,\text{ax}}{\Gamma, A \vdash B \Rightarrow A}\Rightarrow i}{\Gamma \vdash A \Rightarrow (B \Rightarrow A)}\Rightarrow i$$

Logic with proof systems

Notion of proof. I want to prove C. I assume A, B, ... I conclude C.

How are proofs formalised ?

Proof systems with inference rules and sequents

$$\text{hypotheses} \vdash \text{conclusion} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

Proof trees

$$\dfrac{\dfrac{\dfrac{}{\Gamma, A, B \vdash A}\,\text{ax}}{\Gamma, A \vdash B \Rightarrow A} \Rightarrow i}{\Gamma \vdash A \Rightarrow (B \Rightarrow A)} \Rightarrow i$$

$\square$

Computation with functional programs

Functions/programs : $(f : x \mapsto a)$, Computation : $t_1 \rightsquigarrow t_2 \rightsquigarrow \ldots \rightsquigarrow t_n$

Computation with functional programs

Functions/programs : $(f : x \mapsto a)$, Computation : $t_1 \rightsquigarrow t_2 \rightsquigarrow \dots \rightsquigarrow t_n$

How are type systems for functional programs formalised ?

Computation with functional programs

Functions/programs : $(f : x \mapsto a)$, Computation : $t_1 \rightsquigarrow t_2 \rightsquigarrow \ldots \rightsquigarrow t_n$

How are type systems for functional programs formalised ?

Typing systems with typing rules :

context $\vdash$ program : type

$$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash (f : x \mapsto b) : A \to B} \qquad \frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B}$$

Computation with functional programs

Functions/programs : $(f : x \mapsto a)$, Computation : $t_1 \leadsto t_2 \leadsto \ldots \leadsto t_n$

How are type systems for functional programs formalised ?

Typing systems with typing rules :

$$\text{context} \vdash \text{program} : \text{type} \qquad \frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash (f : x \mapsto b) : A \to B} \qquad \frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B}$$

Typing trees

$$\dfrac{\dfrac{\dfrac{}{\Gamma, x : A, y : B \vdash A}\,\mathrm{ax}}{\Gamma, x : A \vdash (y \mapsto x) : B \to A}\to i}{\Gamma \vdash (x \mapsto (y \mapsto x)) : A \to (B \to A)}\to i$$

Computation with functional programs

Functions/programs : $(f : x \mapsto a)$, Computation : $t_1 \rightsquigarrow t_2 \rightsquigarrow \dots \rightsquigarrow t_n$

How are type systems for functional programs formalised ?

Typing systems with typing rules :

context $\vdash$ program : type

$$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash (f : x \mapsto b) : A \to B} \qquad \frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B}$$

Typing trees

$$\dfrac{\dfrac{\dfrac{}{\Gamma, x : A, y : B \vdash A}\,\mathrm{ax}}{\Gamma, x : A \vdash (y \mapsto x) : B \to A}\,\to i}{\Gamma \vdash (x \mapsto (y \mapsto x)) : A \to (B \to A)}\,\to i$$

Curry-Howard-Lambek correspondence (CHL)

Formal correspondence between logic and computation.

$$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash (f : x \mapsto b) : A \to B} \to i \qquad \frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B} \to e$$

Curry-Howard-Lambek correspondence (CHL)

Formal correspondence between logic and computation.

$$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash (f : x \mapsto b) : A \to B} \to i \qquad \frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B} \to e$$

Logic	**Computation**
Inference rules	Typing rules
Formula	Type
Proof	(Functional) Program
Implication $\Rightarrow$	Function type $\to$
Cut-elimination	Execution/evaluation

Curry-Howard-Lambek correspondence (CHL)

Formal correspondence between logic and computation.

$$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash (f : x \mapsto b) : A \to B} \to i \qquad \frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B} \to e$$

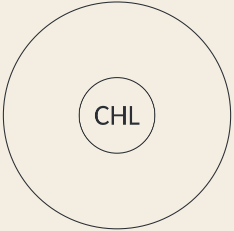
Logic	**Computation**
Inference rules	Typing rules
Formula	Type
Proof	(Functional) Program
Implication $\Rightarrow$	Function type $\to$
Cut-elimination	Execution/evaluation

Leads to : cultural mix in proof/type theory, proof assistants, ...

Curry-Howard-Lambek correspondence (CHL)

The unclear status of logic and computation

Programming = Proving. We only discovered a small part.

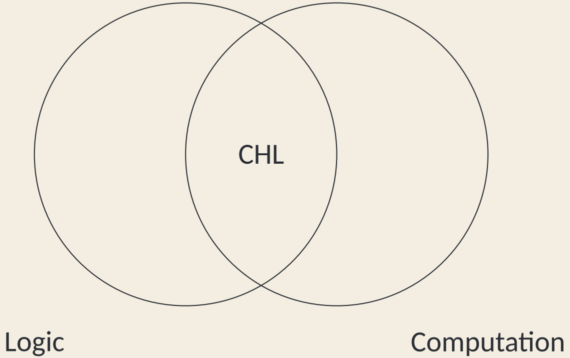


A logico-computational world

Curry-Howard-Lambek correspondence (CHL)

The unclear status of logic and computation

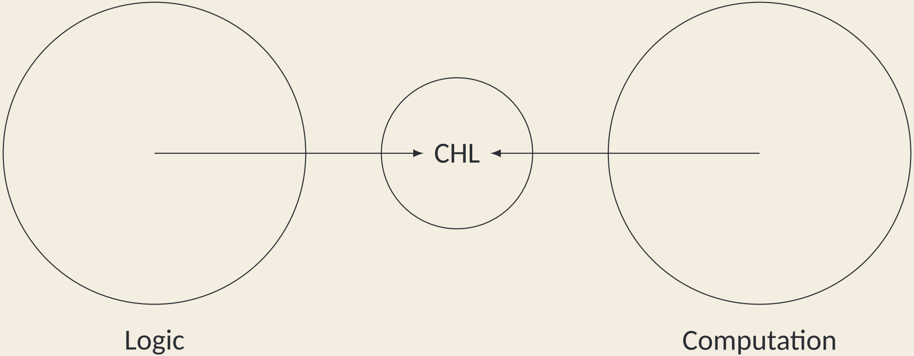
CHL is an intersection. Only some models of computation are logical.



Curry-Howard-Lambek correspondence (CHL)

The unclear status of logic and computation

Disjointness. CHL is an identity.



The approach of Girard's transcendental syntax (according to me) ☐

A long trip

Logic

Computation

Curry-Howard-Lambek (CHL)

Proof-structures / Proof-nets

Jean-Yves Girard's works

Transcendental Syntax

A long trip

Logic

Computation

Curry-Howard-Lambek (CHL)

Proof-structures / Proof-nets

Jean-Yves Girard's works

Transcendental Syntax

The subject of my thesis

- Formalisation of transcendental syntax

A long trip

Logic

Computation

Curry-Howard-Lambek (CHL)

Proof-structures / Proof-nets

Jean-Yves Girard's works

Transcendental Syntax

The subject of my thesis

- Formalisation of transcendental syntax

  ↳ 4 cryptic papers

A long trip

Logic

Computation

Curry-Howard-Lambek (CHL)

Proof-structures / Proof-nets

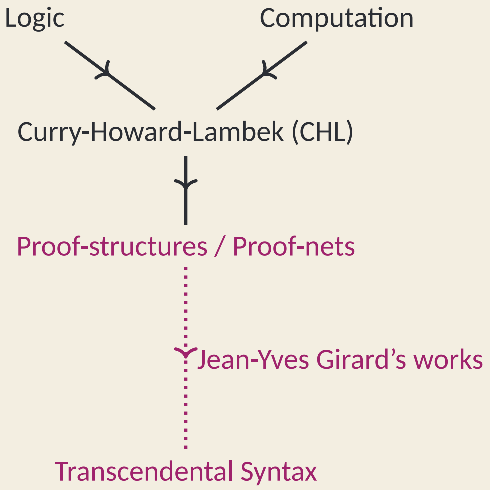
Jean-Yves Girard's works

Transcendental Syntax

The subject of my thesis

- Formalisation of transcendental syntax
  - ↳ 4 cryptic papers
  - ↳ no formal definition / proof

A long trip



The subject of my thesis

- Formalisation of transcendental syntax
  - ↳ 4 cryptic papers
  - ↳ no formal definition / proof
  - ↳ almost no references

A long trip

Logic

Computation

Curry-Howard-Lambek (CHL)

Proof-structures / Proof-nets

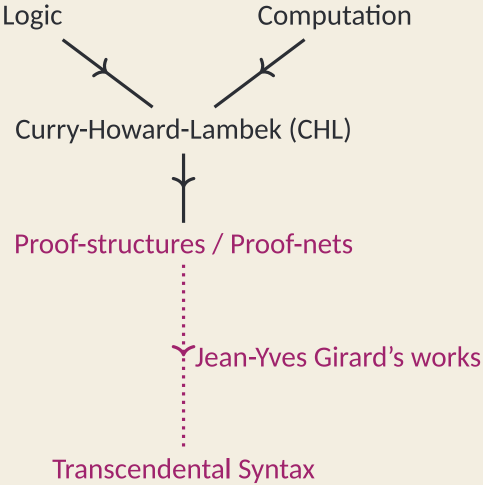
Jean-Yves Girard's works

Transcendental Syntax

The subject of my thesis

- Formalisation of transcendental syntax
  - ↳ 4 cryptic papers
  - ↳ no formal definition / proof
  - ↳ almost no references
  - ↳ no people working on it

A long trip



The subject of my thesis

- Formalisation of transcendental syntax
  - ↳ 4 cryptic papers
  - ↳ no formal definition / proof
  - ↳ almost no references
  - ↳ no people working on it
- Bridge between CHL and TS

A long trip

Logic
Computation

Curry-Howard-Lambek (CHL)

Proof-structures / Proof-nets

Jean-Yves Girard's works

Transcendental Syntax

The subject of my thesis

- Formalisation of transcendental syntax
  ↳ 4 cryptic papers
  ↳ no formal definition / proof
  ↳ almost no references
  ↳ no people working on it
- Bridge between CHL and TS
  ↳ reconstruction of a context

A long trip

Logic

Computation

Curry-Howard-Lambek (CHL)

Proof-structures / Proof-nets

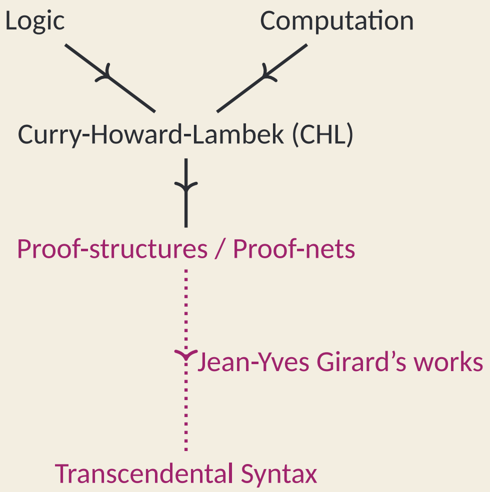
Jean-Yves Girard's works

Transcendental Syntax

The subject of my thesis

- Formalisation of transcendental syntax
  - ↳ 4 cryptic papers
  - ↳ no formal definition / proof
  - ↳ almost no references
  - ↳ no people working on it
- Bridge between CHL and TS
  - ↳ reconstruction of a context
  - ↳ connexions with other subjects

A long trip



The subject of my thesis

- Formalisation of transcendental syntax
  - ↳ 4 cryptic papers
  - ↳ no formal definition / proof
  - ↳ almost no references
  - ↳ no people working on it
- Bridge between CHL and TS
  - ↳ reconstruction of a context
  - ↳ connexions with other subjects
- A new perspective on logic

A long trip

Logic

Computation

Curry-Howard-Lambek (CHL)

Proof-structures / Proof-nets

Jean-Yves Girard's works

Transcendental Syntax

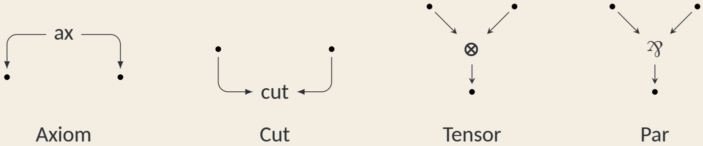
The subject of my thesis

- Formalisation of transcendental syntax
  - ↳ 4 cryptic papers
  - ↳ no formal definition / proof
  - ↳ almost no references
  - ↳ no people working on it
- Bridge between CHL and TS
  - ↳ reconstruction of a context
  - ↳ connexions with other subjects
- A new perspective on logic □

A modern presentation of proofs in multiplicative linear logic

Proof-structures as “aspiring proofs”. A “parallel” presentation of proofs.

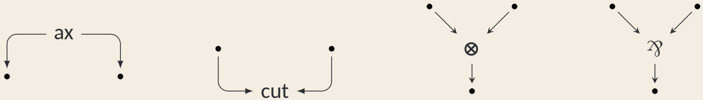
↳ case of multiplicative linear logic (MLL)



A modern presentation of proofs in multiplicative linear logic

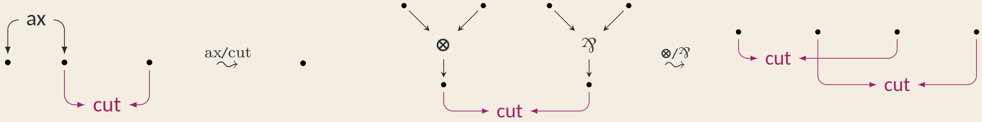
Proof-structures as “aspiring proofs”. A “parallel” presentation of proofs.

↳ case of multiplicative linear logic (MLL)



Axiom Cut Tensor Par

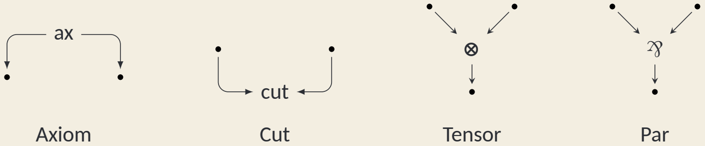
They are reducible networks. Cut-elimination ≃ program evaluation/execution :



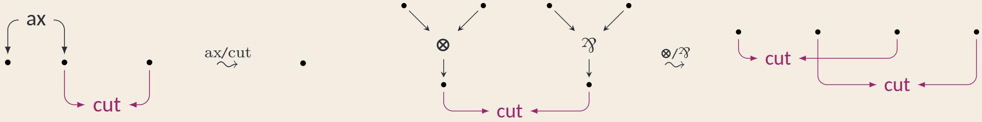
A modern presentation of proofs in multiplicative linear logic

Proof-structures as “aspiring proofs”. A “parallel” presentation of proofs.

↳ case of multiplicative linear logic (MLL)



They are reducible networks. Cut-elimination ≃ program evaluation/execution :

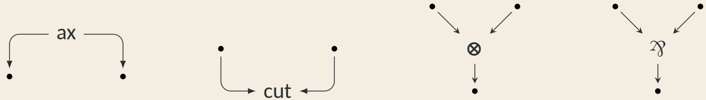


↳ computation with “linear” functional programs (using argument exactly once)

A modern presentation of proofs in multiplicative linear logic

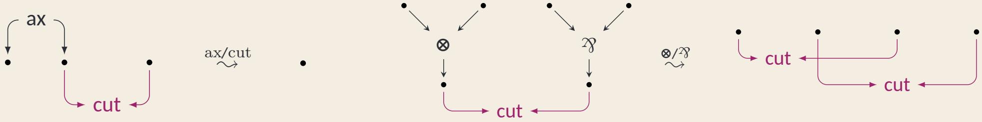
Proof-structures as “aspiring proofs”. A “parallel” presentation of proofs.

↳ case of multiplicative linear logic (MLL)



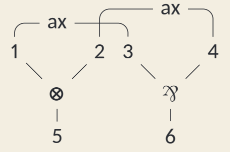
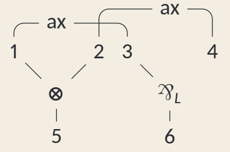
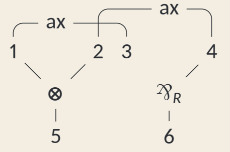
Axiom Cut Tensor Par

They are reducible networks. Cut-elimination $\simeq$ program evaluation/execution :



↳ computation with “linear” functional programs (using argument exactly once) □

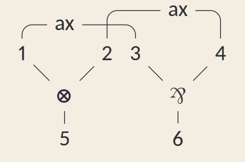
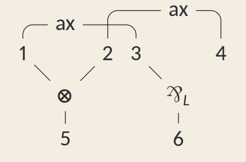
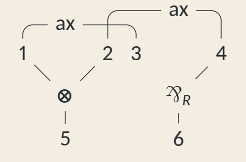
Danos-Regnier correctness criterion by testing

Proof-structure	Switching 1	Switching 2
		

Given a proof-structure :

- it passes all the tests $\longrightarrow$ it is logical (a correct proof = a proof-net)
- it does not $\longrightarrow$ it is not logical

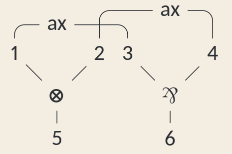
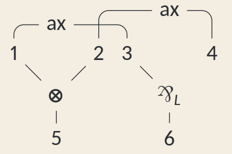
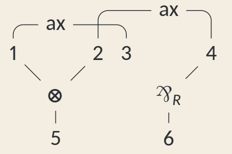
Danos-Regnier correctness criterion by testing

Proof-structure	Switching 1	Switching 2
		

Given a proof-structure :

- it passes all the tests $\longrightarrow$ it is logical (a correct proof = a proof-net)
- it does not $\longrightarrow$ it is not logical
- reminiscent of program testing

Danos-Regnier correctness criterion by testing

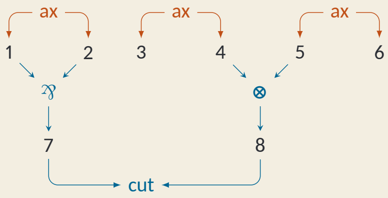
<table>
<tr><th>Proof-structure</th><th>Switching 1</th><th>Switching 2</th></tr>
<tr>
<td>
</td>
<td>
</td>
<td>
</td>
</tr>
</table>

Given a proof-structure :

- it passes all the tests $\longrightarrow$ it is logical (a correct proof = a proof-net)
- it does not $\longrightarrow$ it is not logical
- reminiscent of program testing

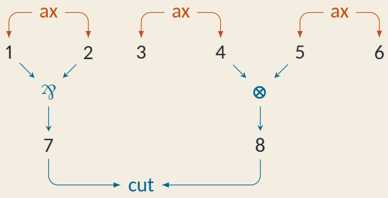
$\square$

Geometry of Interaction (GoI) : an abstraction of proofs



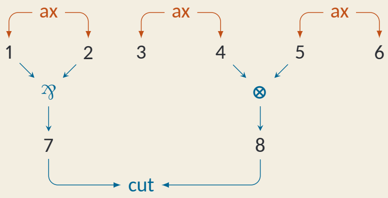
- Logical rules define constraints on possible paths

Geometry of Interaction (GoI) : an abstraction of proofs



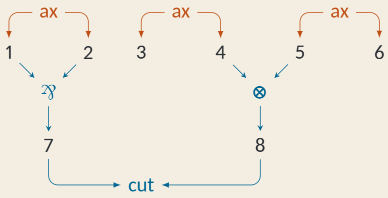
- Logical rules define constraints on possible paths
  - cut-elimination : maximal constrained paths

Geometry of Interaction (GoI) : an abstraction of proofs



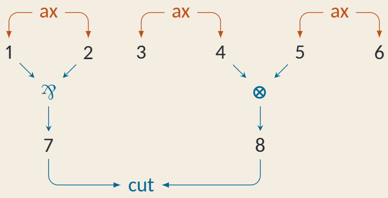
- Logical rules define constraints on possible paths
  - cut-elimination : maximal constrained paths
  - correctness criterion : criterion over constrained paths

Geometry of Interaction (GoI) : an abstraction of proofs



- Logical rules define constraints on possible paths
  - cut-elimination : maximal constrained paths
  - correctness criterion : criterion over constrained paths
- Alternative paths models : permutations, operator algebras, graphs (Seiller) etc

Geometry of Interaction (GoI) : an abstraction of proofs



- Logical rules define constraints on possible paths
  - cut-elimination : maximal constrained paths
  - correctness criterion : criterion over constrained paths
- Alternative paths models : permutations, operator algebras, graphs (Seiller) etc

□

Transcendental Syntax (TS) : a synthesis

Logic reconstructed from computation

Transcendental Syntax (TS) : a synthesis

Logic reconstructed from computation

$$\begin{tikzcd}
\textbf{Computation} & & \textbf{Logic} \\
\text{Reducible object} \arrow[d, "\text{Procedure}"'] & & \text{Typing by interaction (Usage)} \\
& {} \arrow[ur] \arrow[dr] & \\
\text{Result} & & \text{Typing by finite testing (Usine)}
\end{tikzcd}$$

Transcendental Syntax (TS) : a synthesis

Logic reconstructed from computation

Computation	**Logic**
Reducible object	Typing by interaction (Usage)
Procedure	
Result	Typing by finite testing (Usine)

We have to select a model of computation as elementary material.

Transcendental Syntax (TS) : a synthesis

Logic reconstructed from computation

Computation

Reducible object

Procedure

Result

Logic

Typing by interaction (Usage)

Typing by finite testing (Usine)

We have to select a model of computation as elementary material.

- Computation : model of computation called “stellar resolution”

Transcendental Syntax (TS) : a synthesis

Logic reconstructed from computation

Computation	**Logic**
Reducible object	→ Typing by interaction (Usage)
Procedure ↓	
Result	→ Typing by finite testing (Usine)

We have to select a model of computation as elementary material.

- Computation : model of computation called "stellar resolution"
  ↳ because it generalises the notion of contrained path

Transcendental Syntax (TS) : a synthesis

Logic reconstructed from computation

Computation

Reducible object

Procedure

Result

Logic

Typing by interaction (Usage)

Typing by finite testing (Usine)

We have to select a model of computation as elementary material.

- Computation : model of computation called “stellar resolution”
  - ↳ because it generalises the notion of contrained path
  - ↳ only an informal sketch in Girard’s paper

Transcendental Syntax (TS) : a synthesis

Logic reconstructed from computation

Computation	**Logic**
Reducible object	Typing by interaction (Usage)
Procedure	
Result	Typing by finite testing (Usine)

We have to select a model of computation as elementary material.

- Computation : model of computation called “stellar resolution”
  - ↳ because it generalises the notion of contrained path
  - ↳ only an informal sketch in Girard’s paper

To reconstruct (linear) logic through proof-nets.

Transcendental Syntax (TS) : a synthesis

Logic reconstructed from computation

Computation

Reducible object

$\downarrow$ Procedure

Result

Logic

Typing by interaction (Usage)

Typing by finite testing (Usine)

We have to select a model of computation as elementary material.

- Computation : model of computation called “stellar resolution”
  - ↳ because it generalises the notion of contrained path
  - ↳ only an informal sketch in Girard’s paper

To reconstruct (linear) logic through proof-nets. □

Stellar resolution

- Rays : $r ::= X \mid f(r_1, \ldots, r_k) \mid +f(r_1, \ldots, r_k) \mid -f(r_1, \ldots, r_k)$ e.g. $+f(X)$

Stellar resolution

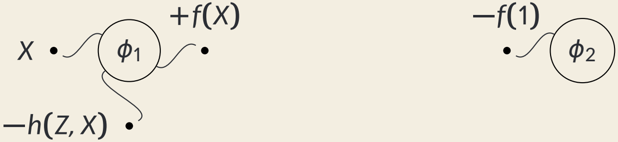
- Rays : $r ::= X \mid f(r_1, \ldots, r_k) \mid +f(r_1, \ldots, r_k) \mid -f(r_1, \ldots, r_k)$ e.g. $+f(X)$
- Stars : finite indexed family of rays $\phi := [r_1, \ldots, r_k]$ e.g. $[X, +f(X), -h(Z, X)]$

Stellar resolution

- Rays : $r ::= X \mid f(r_1, \ldots, r_k) \mid +f(r_1, \ldots, r_k) \mid -f(r_1, \ldots, r_k)$ e.g. $+f(X)$
- Stars : finite indexed family of rays $\phi := [r_1, \ldots, r_k]$ e.g. $[X, +f(X), -h(Z, X)]$
- Constellation : indexed family of stars $\Phi := \phi_1 + \ldots + \phi_n$
  ↳ e.g. $[X, +f(X), -h(Z, X)] + [-f(1)]$

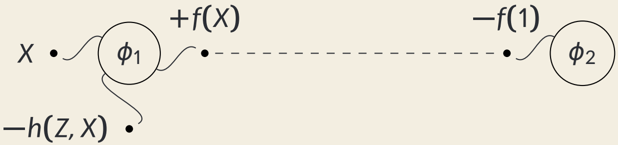
Stellar resolution

- Rays : $r ::= X \mid f(r_1, \dots, r_k) \mid +f(r_1, \dots, r_k) \mid -f(r_1, \dots, r_k)$ e.g. $+f(X)$
- Stars : finite indexed family of rays $\phi := [r_1, \dots, r_k]$ e.g. $[X, +f(X), -h(Z, X)]$
- Constellation : indexed family of stars $\Phi := \phi_1 + \dots + \phi_n$
  ↳ e.g. $[X, +f(X), -h(Z, X)] + [-f(1)]$



Stellar resolution

- Rays : $r ::= X \mid f(r_1, \dots, r_k) \mid +f(r_1, \dots, r_k) \mid -f(r_1, \dots, r_k)$ e.g. $+f(X)$
- Stars : finite indexed family of rays $\phi := [r_1, \dots, r_k]$ e.g. $[X, +f(X), -h(Z, X)]$
- Constellation : indexed family of stars $\Phi := \phi_1 + \dots + \phi_n$
  ↳ e.g. $[X, +f(X), -h(Z, X)] + [-f(1)]$

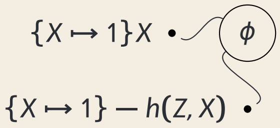


Local interaction by fusion :

$[X, +f(X), -h(Z, X)] \triangledown [-f(1)]$

Stellar resolution

- Rays : $r ::= X \mid f(r_1, \dots, r_k) \mid +f(r_1, \dots, r_k) \mid -f(r_1, \dots, r_k)$ e.g. $+f(X)$
- Stars : finite indexed family of rays $\phi := [r_1, \dots, r_k]$ e.g. $[X, +f(X), -h(Z, X)]$
- Constellation : indexed family of stars $\Phi := \phi_1 + \dots + \phi_n$
  ↳ e.g. $[X, +f(X), -h(Z, X)] + [-f(1)]$

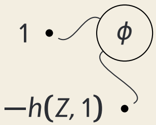


Local interaction by fusion :

$[X, +f(X), -h(Z, X)] \triangledown [-f(1)] = \{X \mapsto 1\}[X, -h(Z, X)]$

Stellar resolution

- Rays : $r ::= X \mid f(r_1, \ldots, r_k) \mid +f(r_1, \ldots, r_k) \mid -f(r_1, \ldots, r_k)$ e.g. $+f(X)$
- Stars : finite indexed family of rays $\phi := [r_1, \ldots, r_k]$ e.g. $[X, +f(X), -h(Z, X)]$
- Constellation : indexed family of stars $\Phi := \phi_1 + \ldots + \phi_n$
  ↳ e.g. $[X, +f(X), -h(Z, X)] + [-f(1)]$

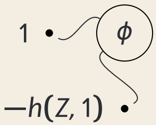


Local interaction by fusion :

$$[X, +f(X), -h(Z, X)] \triangledown [-f(1)] = \{X \mapsto 1\}[X, -h(Z, X)] = [1, -h(Z, 1)]$$

Stellar resolution

- Rays : $r ::= X \mid f(r_1, \ldots, r_k) \mid +f(r_1, \ldots, r_k) \mid -f(r_1, \ldots, r_k)$ e.g. $+f(X)$
- Stars : finite indexed family of rays $\phi := [r_1, \ldots, r_k]$ e.g. $[X, +f(X), -h(Z, X)]$
- Constellation : indexed family of stars $\Phi := \phi_1 + \ldots + \phi_n$
  ↳ e.g. $[X, +f(X), -h(Z, X)] + [-f(1)]$



Local interaction by fusion :

$[X, +f(X), -h(Z, X)] \triangledown [-f(1)] = \{X \mapsto 1\}[X, -h(Z, X)] = [1, -h(Z, 1)]$

↳ Variant of Robinson's resolution used in logic programming

Programming with stellar resolution

Execution $\mathrm{Ex}(\Phi)$: construct all "connexion graphs", contracts everything by fusion

↳ obtain a new constellation

Programming with stellar resolution

Execution $\text{Ex}(\Phi)$: construct all “connexion graphs”, contracts everything by fusion

↳ obtain a new constellation

↳ programming with structural constraints and information flows.

Programming with stellar resolution

Execution $\mathrm{Ex}(\Phi)$: construct all “connexion graphs”, contracts everything by fusion

↳ obtain a new constellation

↳ programming with structural constraints and information flows.

↳ 3 methods of execution developed in the thesis.

Programming with stellar resolution

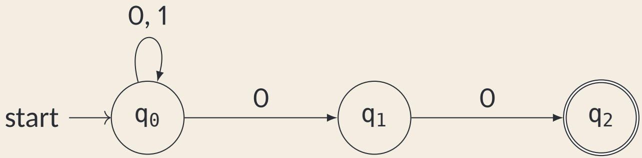
Execution $\mathrm{Ex}(\Phi)$: construct all “connexion graphs”, contracts everything by fusion

↳ obtain a new constellation

↳ programming with structural constraints and information flows.

↳ 3 methods of execution developed in the thesis.

$A =$



Programming with stellar resolution

Execution $\mathrm{Ex}(\Phi)$: construct all "connexion graphs", contracts everything by fusion

↳ obtain a new constellation

↳ programming with structural constraints and information flows.

↳ 3 methods of execution developed in the thesis.

$$A^\star := [-i(W), +a(W, \mathtt{q_0})] + [-a(\epsilon, \mathtt{q_2}), \mathtt{accept}] + [-a(0 \cdot W, \mathtt{q_0}), +a(W, \mathtt{q_0})] +$$

$$[-a(1 \cdot W, \mathtt{q_0}), +a(W, \mathtt{q_0})] + [-a(0 \cdot W, \mathtt{q_0}), +a(W, \mathtt{q_1})] + [-a(0 \cdot W, \mathtt{q_1}), +a(W, \mathtt{q_2})]$$

Programming with stellar resolution

Execution $\mathrm{Ex}(\Phi)$: construct all “connexion graphs”, contracts everything by fusion

↳ obtain a new constellation

↳ programming with structural constraints and information flows.

↳ 3 methods of execution developed in the thesis.

$$A^\star := [-i(W), +a(W, \mathtt{q}_0)] + [-a(\epsilon, \mathtt{q}_2), \mathtt{accept}] + [-a(0 \cdot W, \mathtt{q}_0), +a(W, \mathtt{q}_0)] +$$

$$[-a(1 \cdot W, \mathtt{q}_0), +a(W, \mathtt{q}_0)] + [-a(0 \cdot W, \mathtt{q}_0), +a(W, \mathtt{q}_1)] + [-a(0 \cdot W, \mathtt{q}_1), +a(W, \mathtt{q}_2)]$$

The problem $[\mathtt{accept}] \overset{?}{\in} \mathrm{Ex}(A^\star)$ simulates word acceptance.

Results on stellar resolution

Several encoding of models of computation :

Results on stellar resolution

Several encoding of models of computation :

- logic programs (with Horn clauses)

Results on stellar resolution

Several encoding of models of computation :

- logic programs (with Horn clauses)
- generalised circuits (subsuming boolean/arithmetic circuits)

Results on stellar resolution

Several encoding of models of computation :

- logic programs (with Horn clauses)
- generalised circuits (subsuming boolean/arithmetic circuits)
- state machines (automata, Turing machines, ...)

Results on stellar resolution

Several encoding of models of computation :

- logic programs (with Horn clauses)
- generalised circuits (subsuming boolean/arithmetic circuits)
- state machines (automata, Turing machines, ...)
- self-assembling tile systems (used in DNA computing)

Results on stellar resolution

Several encoding of models of computation :

- logic programs (with Horn clauses)
- generalised circuits (subsuming boolean/arithmetic circuits)
- state machines (automata, Turing machines, ...)
- self-assembling tile systems (used in DNA computing)
  ↳ encoding of abstract tile assembly model (aTAM)

Results on stellar resolution

Several encoding of models of computation :

- logic programs (with Horn clauses)
- generalised circuits (subsuming boolean/arithmetic circuits)
- state machines (automata, Turing machines, ...)
- self-assembling tile systems (used in DNA computing)
  ↳ encoding of abstract tile assembly model (aTAM)
  ↳ stellar resolution very close to Jonoska's flexible tiles

Results on stellar resolution

Several encoding of models of computation :

- logic programs (with Horn clauses)
- generalised circuits (subsuming boolean/arithmetic circuits)
- state machines (automata, Turing machines, ...)
- self-assembling tile systems (used in DNA computing)
  ↳ encoding of abstract tile assembly model (aTAM)
  ↳ stellar resolution very close to Jonoska's flexible tiles

Associativity of fusion. when fusion succeeds, $\phi_1 \overset{i,j}{\triangledown} (\phi_2 \overset{i',j'}{\triangledown} \phi_3) \approx_\alpha (\phi_1 \overset{i,j}{\triangledown} \phi_2) \overset{i',j'}{\triangledown} \phi_3$

Results on stellar resolution

Several encoding of models of computation :

- logic programs (with Horn clauses)
- generalised circuits (subsuming boolean/arithmetic circuits)
- state machines (automata, Turing machines, ...)
- self-assembling tile systems (used in DNA computing)
  ↳ encoding of abstract tile assembly model (aTAM)
  ↳ stellar resolution very close to Jonoska's flexible tiles

Associativity of fusion. when fusion succeeds, $\phi_1 \overset{i,j}{\triangledown} (\phi_2 \overset{i',j'}{\triangledown} \phi_3) \approx_\alpha (\phi_1 \overset{i,j}{\triangledown} \phi_2) \overset{i',j'}{\triangledown} \phi_3$

Partial pre-execution (under condition). $\mathrm{Ex}(\Phi \uplus \Phi') \simeq \mathrm{Ex}(\mathrm{Ex}(\Phi) \uplus \Phi')$

Results on stellar resolution

Several encoding of models of computation :

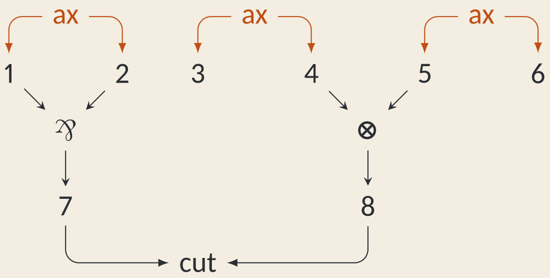
- logic programs (with Horn clauses)
- generalised circuits (subsuming boolean/arithmetic circuits)
- state machines (automata, Turing machines, ...)
- self-assembling tile systems (used in DNA computing)
  ↳ encoding of abstract tile assembly model (aTAM)
  ↳ stellar resolution very close to Jonoska's flexible tiles

Associativity of fusion. when fusion succeeds, $\phi_1 \overset{i,j}{\triangledown} (\phi_2 \overset{i',j'}{\triangledown} \phi_3) \approx_\alpha (\phi_1 \overset{i,j}{\triangledown} \phi_2) \overset{i',j'}{\triangledown} \phi_3$

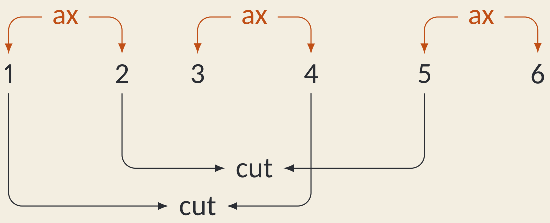
Partial pre-execution (under condition). $\mathrm{Ex}(\Phi \uplus \Phi') \simeq \mathrm{Ex}(\mathrm{Ex}(\Phi) \uplus \Phi')$

Associativity of execution. $\mathrm{Ex}(\Phi_1 \uplus \mathrm{Ex}(\Phi_2 \uplus \Phi_3)) \simeq \mathrm{Ex}(\mathrm{Ex}(\Phi_1 \uplus \Phi_2) \uplus \Phi_3)$

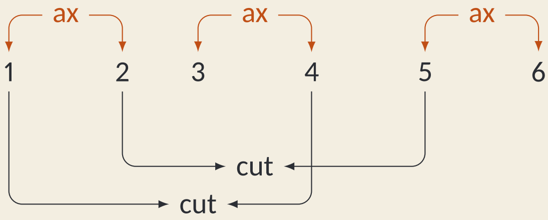
Interpretation of multiplicative linear logic



Interpretation of multiplicative linear logic

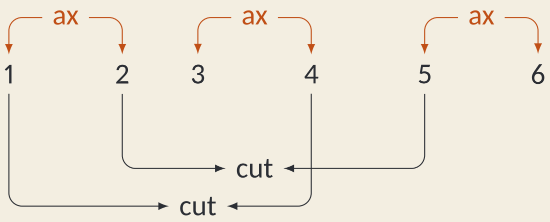


Interpretation of multiplicative linear logic



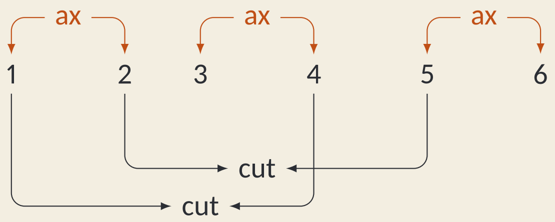
$$[+7(\mathtt{l}\cdot X), +7(\mathtt{r}\cdot X)] + [3(X), +8(\mathtt{l}\cdot X)] + [+8(\mathtt{r}\cdot X), 6(X)]$$

Interpretation of multiplicative linear logic



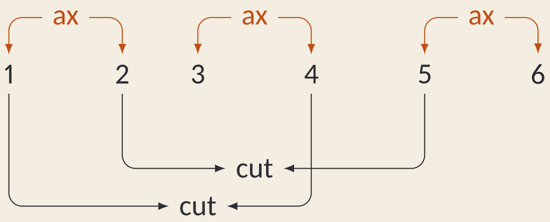
$$[\,+7(\overset{1}{\mathtt{l}}\cdot X), +7(\overset{2}{\mathtt{r}}\cdot X)\,] + [\,3(\overset{3}{X}), +8(\overset{4}{\mathtt{l}}\cdot X)\,] + [\,+8(\overset{5}{\mathtt{r}}\cdot X), 6(\overset{6}{X})\,]$$

Interpretation of multiplicative linear logic



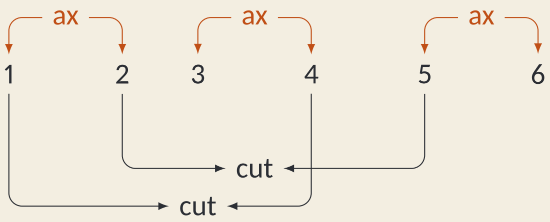
$$[+7(\overset{1}{\mathtt{l}}\cdot X), +7(\overset{2}{\mathtt{r}}\cdot X)] + [3(\overset{3}{X}), +8(\overset{4}{\mathtt{l}}\cdot X)] + [+8(\overset{5}{\mathtt{r}}\cdot X), 6(\overset{6}{X})]$$

Interpretation of multiplicative linear logic



$$[+7(\overset{1}{\mathtt{l}}\cdot X), +7(\overset{2}{\mathtt{r}}\cdot X)] + [3(\overset{3}{X}), +8(\overset{4}{\mathtt{l}}\cdot X)] + [+8(\overset{5}{\mathtt{r}}\cdot X), 6(\overset{6}{X})] + [-7(X), -8(X)].$$

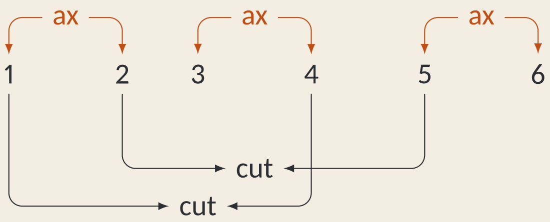
Interpretation of multiplicative linear logic



$$[+7(\overset{1}{\mathtt{l}}\cdot X), +7(\overset{2}{\mathtt{r}}\cdot X)] + [3(\overset{3}{X}), +8(\overset{4}{\mathtt{l}}\cdot X)] + [+8(\overset{5}{\mathtt{r}}\cdot X), 6(\overset{6}{X})] + [-7(X), -8(X)].$$

More general framework : non-proof-structures can enjoy a logical Interpretation

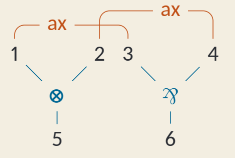
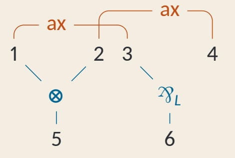
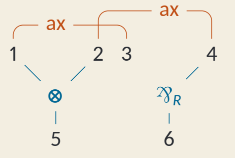
Interpretation of multiplicative linear logic



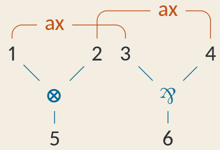
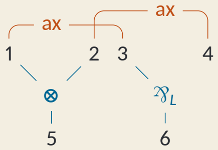
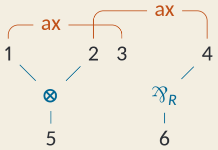
$$[\overset{1}{+7(\mathtt{l}\cdot X)}, \overset{2}{+7(\mathtt{r}\cdot X)}] + [\overset{3}{3(X)}, \overset{4}{+8(\mathtt{l}\cdot X)}] + [\overset{5}{+8(\mathtt{r}\cdot X)}, \overset{6}{6(X)}] + [-7(X), -8(X)].$$

More general framework : non-proof-structures can enjoy a logical Interpretation □

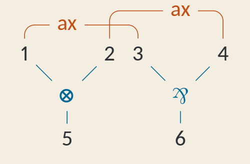
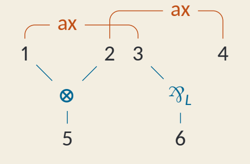
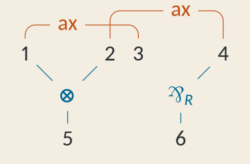
Logical interpretation : Girard's Usine

Proof-structure	Switching 1	Switching 2
		

Logical interpretation : Girard's Usine

<table>
<tr><th>Proof-structure</th><th>Switching 1</th><th>Switching 2</th></tr>
<tr>
<td>
</td>
<td>
</td>
<td>
</td>
</tr>
</table>

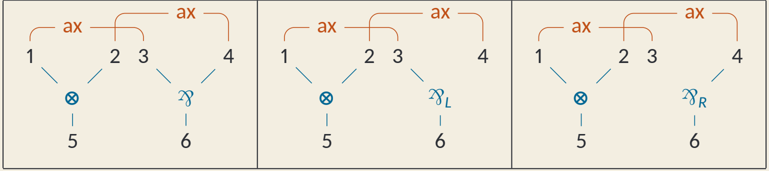
Logical interpretation : Girard's Usine

Proof-structure	Switching 1	Switching 2
		

- Top and bottom become constellations structurally imitating proof-structures

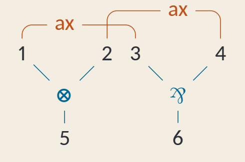
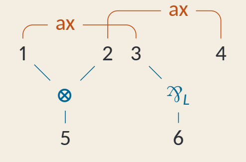
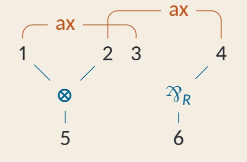
Logical interpretation : Girard’s Usine

Proof-structure	Switching 1	Switching 2



- Top and bottom become constellations structurally imitating proof-structures
  ↳ tensor ⊗ is $[-1(X), -2(X), 5(X)]$

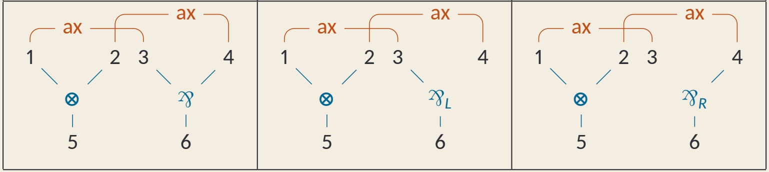
Logical interpretation : Girard's Usine

Proof-structure	Switching 1	Switching 2
		

- Top and bottom become constellations structurally imitating proof-structures
  - ↳ tensor $\otimes$ is $[-1(X), -2(X), 5(X)]$
  - ↳ left par $\parr_L$ is $[-3(X), 6(X)] + [-4(X)]$

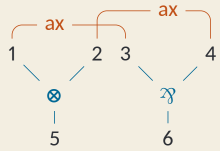
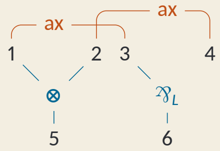
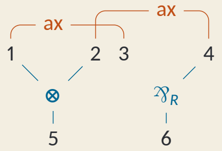
Logical interpretation : Girard's Usine

Proof-structure	Switching 1	Switching 2



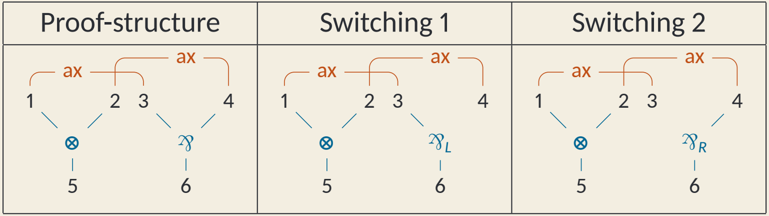
- Top and bottom become constellations structurally imitating proof-structures
  - ↳ tensor ⊗ is $[-1(X), -2(X), 5(X)]$
  - ↳ left par $⅋_L$ is $[-3(X), 6(X)] + [-4(X)]$
  - ↳ Φ interacting with $\Phi_{\mathtt{switch}_1}, \ldots, \Phi_{\mathtt{switch}_n}$

Logical interpretation : Girard's Usine

Proof-structure	Switching 1	Switching 2
		

- Top and bottom become constellations structurally imitating proof-structures
  - ↳ tensor $\otimes$ is $[-1(X), -2(X), 5(X)]$
  - ↳ left par $\parr_L$ is $[-3(X), 6(X)] + [-4(X)]$
  - ↳ Φ interacting with $\Phi_{\mathtt{switch}_1}, \ldots, \Phi_{\mathtt{switch}_n}$
- Usine : judges from structure/shape/appearance, generalises type systems

Logical interpretation : Girard's Usine



- Top and bottom become constellations structurally imitating proof-structures
  - ↳ tensor $\otimes$ is $[-1(X), -2(X), 5(X)]$
  - ↳ left par $⅋_L$ is $[-3(X), 6(X)] + [-4(X)]$
  - ↳ Φ interacting with $\Phi_{\mathtt{switch}_1}, \ldots, \Phi_{\mathtt{switch}_n}$
- Usine : judges from structure/shape/appearance, generalises type systems

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

Logical interpretation : Girard’s Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine’s classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$

  ↳ Φ passes the test Φ' (or the converse)

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'
  - ↳ 3 orthogonality relations studied in the thesis

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'
  - ↳ 3 orthogonality relations studied in the thesis
- Constellations organised in "social groups" (sets) $\mathbf{A} := \{\Phi_1, \ldots, \Phi_n\}$

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'
  - ↳ 3 orthogonality relations studied in the thesis
- Constellations organised in "social groups" (sets) $\mathbf{A} := \{\Phi_1, \ldots, \Phi_n\}$
- Orthogonal (sort of negation) $\mathbf{A}^{\perp} := \{\Phi \mid \forall \Phi' \in \mathbf{A}, \Phi \perp \Phi'\}$

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'
  - ↳ 3 orthogonality relations studied in the thesis
- Constellations organised in "social groups" (sets) $\mathbf{A} := \{\Phi_1, \ldots, \Phi_n\}$
- Orthogonal (sort of negation) $\mathbf{A}^\perp := \{\Phi \mid \forall \Phi' \in \mathbf{A}, \Phi \perp \Phi'\}$
- Type/Behaviours $\mathbf{A}$ when $\exists \mathbf{B}.\ \mathbf{A} = \mathbf{B}^\perp$, definition by interaction

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'
  - ↳ 3 orthogonality relations studied in the thesis
- Constellations organised in "social groups" (sets) $\mathbf{A} := \{\Phi_1, \dots, \Phi_n\}$
- Orthogonal (sort of negation) $\mathbf{A}^\perp := \{\Phi \mid \forall \Phi' \in \mathbf{A}, \Phi \perp \Phi'\}$
- Type/Behaviours $\mathbf{A}$ when $\exists \mathbf{B}.\ \mathbf{A} = \mathbf{B}^\perp$, definition by interaction
- Types as combination of behaviours : $\mathbf{A} \otimes \mathbf{B}$, $\mathbf{A} \parr \mathbf{B}$, $\mathbf{A} \multimap \mathbf{B}$

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'
  - ↳ 3 orthogonality relations studied in the thesis
- Constellations organised in "social groups" (sets) $\mathbf{A} := \{\Phi_1, \dots, \Phi_n\}$
- Orthogonal (sort of negation) $\mathbf{A}^\perp := \{\Phi \mid \forall \Phi' \in \mathbf{A}, \Phi \perp \Phi'\}$
- Type/Behaviours $\mathbf{A}$ when $\exists \mathbf{B}.\ \mathbf{A} = \mathbf{B}^\perp$, definition by interaction
- Types as combination of behaviours : $\mathbf{A} \otimes \mathbf{B}$, $\mathbf{A} \parr \mathbf{B}$, $\mathbf{A} \multimap \mathbf{B}$ $\square$

Classical results : soundness and completeness

Classical theorems of logic but new in this context (MLL and MLL+MIX).

Classical results : soundness and completeness

Classical theorems of logic but new in this context (MLL and MLL+MIX).

Induced behaviour. Type/formula label $\vdash \Gamma$ turned into Usage's behaviour $[\![\Gamma]\!]$.

Classical results : soundness and completeness

Classical theorems of logic but new in this context (MLL and MLL+MIX).

Induced behaviour. Type/formula label $\vdash \Gamma$ turned into Usage's behaviour $[\![\Gamma]\!]$.

Full soundness. Given a proof-net $\mathscr{R}$ of conclusions $\vdash \Gamma$, $\mathrm{Ex}(\mathscr{R}^\star) \in [\![\vdash \Gamma]\!]$.

Classical results : soundness and completeness

Classical theorems of logic but new in this context (MLL and MLL+MIX).

Induced behaviour. Type/formula label $\vdash \Gamma$ turned into Usage's behaviour $[\![\Gamma]\!]$.

Full soundness. Given a proof-net $\mathscr{R}$ of conclusions $\vdash \Gamma$, $\mathrm{Ex}(\mathscr{R}^{\bigstar}) \in [\![\vdash \Gamma]\!]$.

Completeness. If a constellation $\Phi \in [\![\vdash \Gamma]\!]$ is "proof-like" w.r.t. $\vdash \Gamma$, then there is a corresponding proof-net of conclusions $\vdash \Gamma$.

Classical results : soundness and completeness

Classical theorems of logic but new in this context (MLL and MLL+MIX).

Induced behaviour. Type/formula label $\vdash \Gamma$ turned into Usage's behaviour $[\![\Gamma]\!]$.

Full soundness. Given a proof-net $\mathscr{R}$ of conclusions $\vdash \Gamma$, $\mathrm{Ex}(\mathscr{R}^{\star}) \in [\![\vdash \Gamma]\!]$.

Completeness. If a constellation $\Phi \in [\![\vdash \Gamma]\!]$ is "proof-like" w.r.t. $\vdash \Gamma$, then there is a corresponding proof-net of conclusions $\vdash \Gamma$.

$\square$

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$

  ↳ Φ passes the test Φ' (or the converse)

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'
- Constellations organised in "social groups" (sets) $\mathbf{A} := \{\Phi_1, \ldots, \Phi_n\}$

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'
- Constellations organised in "social groups" (sets) $\mathbf{A} := \{\Phi_1, \ldots, \Phi_n\}$
- Orthogonal (sort of negation) $\mathbf{A}^{\perp} := \{\Phi \mid \forall \Phi' \in \mathbf{A}, \Phi \perp \Phi'\}$

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'
- Constellations organised in "social groups" (sets) $\mathbf{A} := \{\Phi_1, \ldots, \Phi_n\}$
- Orthogonal (sort of negation) $\mathbf{A}^\perp := \{\Phi \mid \forall \Phi' \in \mathbf{A}, \Phi \perp \Phi'\}$
- Type/Behaviours $\mathbf{A}$ when $\exists \mathbf{B}.\ \mathbf{A} = \mathbf{B}^\perp$, definition by interaction

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'
- Constellations organised in "social groups" (sets) $\mathbf{A} := \{\Phi_1, \dots, \Phi_n\}$
- Orthogonal (sort of negation) $\mathbf{A}^{\perp} := \{\Phi \mid \forall \Phi' \in \mathbf{A}, \Phi \perp \Phi'\}$
- Type/Behaviours $\mathbf{A}$ when $\exists \mathbf{B}.\ \mathbf{A} = \mathbf{B}^{\perp}$, definition by interaction
- Types as combination of behaviours : $\mathbf{A} \otimes \mathbf{B}$, $\mathbf{A} \parr \mathbf{B}$, $\mathbf{A} \multimap \mathbf{B}$

Logical interpretation : Girard's Usage

Usage : judges from actions/interactions

↳ uses realisability techniques (e.g. Krivine's classical realisability)

↳ alternative approaches : Riba (with untyped λ-calculus), Beffara (with process calculi)

- Choice of a symmetric binary "orthogonality" relation over constellations $\Phi \perp \Phi'$
  - ↳ Φ passes the test Φ' (or the converse)
  - ↳ Φ interacts correctly with Φ'
- Constellations organised in "social groups" (sets) $\mathbf{A} := \{\Phi_1, \ldots, \Phi_n\}$
- Orthogonal (sort of negation) $\mathbf{A}^\perp := \{\Phi \mid \forall \Phi' \in \mathbf{A}, \Phi \perp \Phi'\}$
- Type/Behaviours $\mathbf{A}$ when $\exists \mathbf{B}.\ \mathbf{A} = \mathbf{B}^\perp$, definition by interaction
- Types as combination of behaviours : $\mathbf{A} \otimes \mathbf{B}$, $\mathbf{A} \parr \mathbf{B}$, $\mathbf{A} \multimap \mathbf{B}$

□

Conclusion

Future works and some possible extensions

Conclusion

Future works and some possible extensions

- Extensions beyond multiplicative linear logic (MLL)

Conclusion

Future works and some possible extensions

- Extensions beyond multiplicative linear logic (MLL)

  ↳ non-determinism (MALL) and duplication/erasure (MELL)

Conclusion

Future works and some possible extensions

- Extensions beyond multiplicative linear logic (MLL)
  - ↳ non-determinism (MALL) and duplication/erasure (MELL)
  - ↳ Girard's apodictic/epidictic (local and global mechanisms)

Conclusion

Future works and some possible extensions

- Extensions beyond multiplicative linear logic (MLL)
  - ↳ non-determinism (MALL) and duplication/erasure (MELL)
  - ↳ Girard's apodictic/epidictic (local and global mechanisms)
  - ↳ update of proof-nets

Conclusion

Future works and some possible extensions

- Extensions beyond multiplicative linear logic (MLL)
  - ↳ non-determinism (MALL) and duplication/erasure (MELL)
  - ↳ Girard's apodictic/epidictic (local and global mechanisms)
  - ↳ update of proof-nets
- Interpretation of predicate calculus

Conclusion

Future works and some possible extensions

- Extensions beyond multiplicative linear logic (MLL)
  ↳ non-determinism (MALL) and duplication/erasure (MELL)
  ↳ Girard's apodictic/epidictic (local and global mechanisms)
  ↳ update of proof-nets
- Interpretation of predicate calculus
- Possible generalisation of Krivine and Beffara works on realisability

Conclusion

Future works and some possible extensions

- Extensions beyond multiplicative linear logic (MLL)
  - ↳ non-determinism (MALL) and duplication/erasure (MELL)
  - ↳ Girard's apodictic/epidictic (local and global mechanisms)
  - ↳ update of proof-nets
- Interpretation of predicate calculus
- Possible generalisation of Krivine and Beffara works on realisability
- Open problems in computational complexity (e.g. $\mathsf{P} \overset{?}{=} \mathsf{NP}$).

Conclusion

Future works and some possible extensions

- Extensions beyond multiplicative linear logic (MLL)
  ↳ non-determinism (MALL) and duplication/erasure (MELL)
  ↳ Girard's apodictic/epidictic (local and global mechanisms)
  ↳ update of proof-nets
- Interpretation of predicate calculus
- Possible generalisation of Krivine and Beffara works on realisability
- Open problems in computational complexity (e.g. $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$).
  ↳ Implicit complexity : proof systems capturing complexity classes

Conclusion

Future works and some possible extensions

- Extensions beyond multiplicative linear logic (MLL)
  ↳ non-determinism (MALL) and duplication/erasure (MELL)
  ↳ Girard's apodictic/epidictic (local and global mechanisms)
  ↳ update of proof-nets
- Interpretation of predicate calculus
- Possible generalisation of Krivine and Beffara works on realisability
- Open problems in computational complexity (e.g. $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$).
  ↳ Implicit complexity : proof systems capturing complexity classes
  ↳ Descriptive complexity : logical languages capturing complexity classes

Conclusion

Future works and some possible extensions

- Extensions beyond multiplicative linear logic (MLL)
  - ↳ non-determinism (MALL) and duplication/erasure (MELL)
  - ↳ Girard's apodictic/epidictic (local and global mechanisms)
  - ↳ update of proof-nets
- Interpretation of predicate calculus
- Possible generalisation of Krivine and Beffara works on realisability
- Open problems in computational complexity (e.g. $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$).
  - ↳ Implicit complexity : proof systems capturing complexity classes
  - ↳ Descriptive complexity : logical languages capturing complexity classes

Appendix

Stellar resolution : execution

Abstract execution

Actual connexion $\longmapsto$ Dependency multigraph (showing compatible rays)

$$\phi_1 — \phi_2 — \phi_2' — \phi_2'' — \phi_3 \qquad\qquad \phi_1 — \phi_2 — \phi_3$$

Diagram. Multigraph homomorphism $\delta : G_\delta \to \mathfrak{D}[\Phi]$

↳ with functions δ_v for each vertex v associating rays to incident edges

↳ G_δ required to be non-empty, finite and connected

Diagram evaluation. Edge contraction by fusion (correct diagram if no failure)

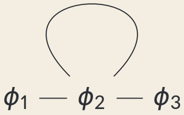
Execution. $\mathrm{Ex}(\Phi) =$ evaluation of all saturated and correct (no failure) diagrams.

Variants. Effective versions with concrete and interactive execution.

Stellar resolution : execution (2/2)

Concrete and interactive execution

Concrete execution. Iterative construction of diagrams / tilings.



$\phi_1 - \phi_2 - \phi_3$ evaluated into ψ_1

$\phi_1 - \phi_2 - \phi_2' - \phi_3$ evaluated into ψ_2

$\phi_1 - \phi_2 - \phi_2' - \phi_2''$

...

Duplicates removed by checking multigraph isomorphism...

Interactive execution. Fusion of stars “on the fly” (without postponing evaluation)