Fachhochschule Nordwestschweiz
Hochschule für Informatik

Linear Logic: Its Syntax (and Semantics)

Jean-Yves Girard, 1995

PL Circle
28.11.2025

Übersicht

- Jean-Yves Girard
- Andere Logiken: Klassisch und Intuitionismus
- Linear Logic
  - Basics
  - Syntax
  - Linear Sequent Calculus
  - Proof Nets
- Anwendungen in der Computer Science
  - Linear/Affine Types
  - Session Types
  - Proofs as Concurrent Programs

> Scientific standards are terrible, one should never joke[79] : to be taken seriously, put people to sleep ! In a paper, no funny drawings, it would be a waste of paper. However, full pages of repetitive definitions, of Prussian formalism, are not considered as a waste.

Jean-Yves Girard

- 1947 in Lyon
- Logiker und Mathematiker, Beweistheorie
- Emeritierter directeur de recherche am CNRS
- Médaille d'argent du CNRS, 1983
- Getypte Polymorphe Form des λ-Kalküls, Système F (unabhängig zu John C. Reynolds)
  - "Girard's idea": Candidats de réductibilité
  - Girard–Reynolds isomorphism
- **Lineare Logik: 1987**

Yann-Joachim Ringard

- Pseudonym von J.-Y. Girard
- “Les montres à moutarde, une approche intégrée au temps et à la nourriture”
- Satirische Kritik an der Semantik von Kripke (Modallogik)
- Mehr als 5000 Film-Reviews seit 2019

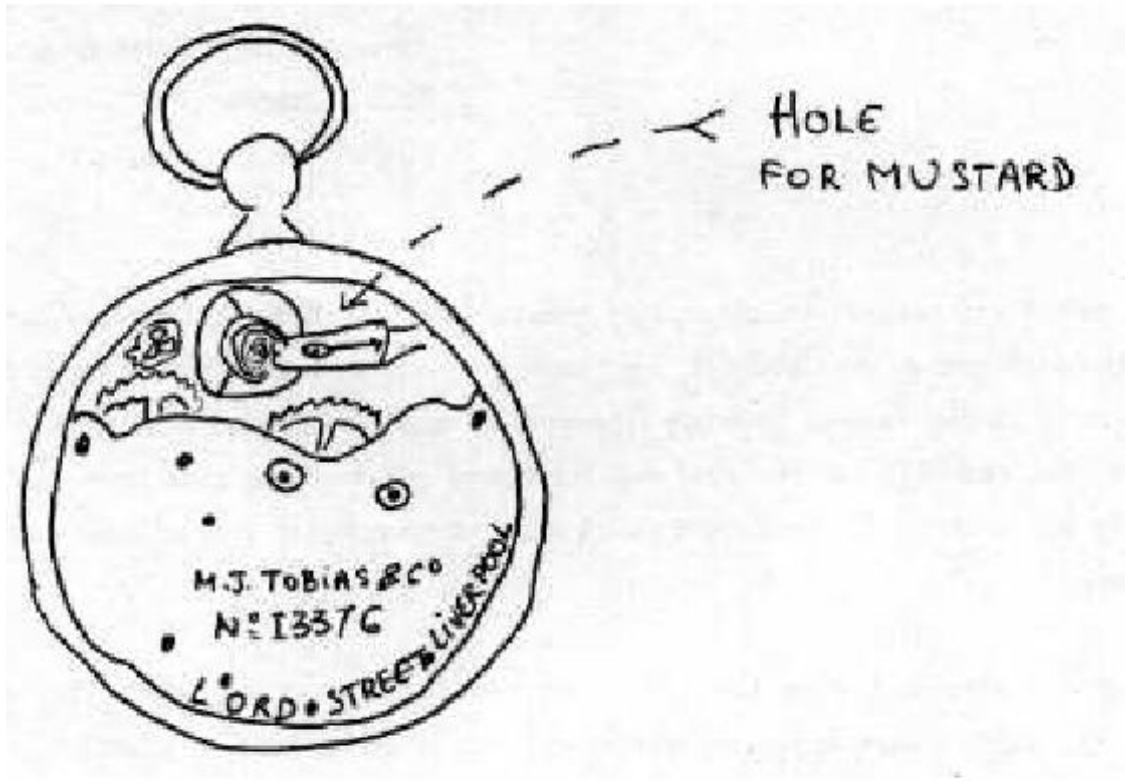


FIG. 1 – Montre à moutarde avant recharge

1 2

à moutarde est dite propre *quand la quantité de moutarde qu'ell*
on nulle ; elle est dite impropre ou dégénérée *autrement.*

Verschiedene Logiken

Klassische Logik

$P \vee \neg P$ ist wahr

$A \rightarrow \neg\neg A$ UND $\neg\neg A \rightarrow A$

Intuitionistische Logik

Stärker an Beweistheorie angelehnt

- Klassische Logik ohne excluded middle und double negation eliminitation
- $P \vee \neg P$ als Existenz oder Widerlegung einer Aussage, aber kann auch keines von beidem geben
- Grundlage für Curry-Howard correspondence

Inspiration für Coq, Agda, proof assistants

Syntax

−◦

&

A−◦B: Lineare Implikation

- A “entails”/“ermöglicht” B

⊗ (times), => UND

& (with) => “Entweder oder”

A−◦B, A−◦C

~~A −◦ B ⊗ C~~

A ⊗ A −◦ B ⊗ C

⊕ (plus), wie &, aber ohne Kontrolle

⅋ (par) “Parallel”

- $A ⅋ B, A^{\perp} −◦ B, B^{\perp} −◦ A$

Negation:

- $A^{\perp}$
- “Bedarf nach einem A”

Exponentials:

- !A, beliebig viele A
- ?A, unbekannt viele A

Lineare Abbildung: f(a + b) = f(a) + f(b)

Weakening und Contraction

Klassische/Intuitionistische Logik:

$A \wedge B \Rightarrow A$

$A \Rightarrow A \wedge A$

In Linear Logik mit ! möglich, um explizit zwischen Ressourcen und Aussagen zu unterscheiden

Linear Logik

$A \multimap A$

~~$A \multimap A \otimes A$~~

$!A \multimap !A \otimes !A$

?A: “Eventuell” A

$$(!A)^{\perp} := ?A^{\perp}$$

$$(?A)^{\perp} := !A^{\perp}$$

Exponentials

!A indicates the possibility of using A ad libitum ; it only indicates a potentiality, in the same way that a piece of paper
on the slot of a copying machine can be copied . . . but nobody would identify
a copying machine with all the copies it produces

The rules for the dual “?” (why not) are precisely the three basic ways of actualizing this potentiality :
erasing (weakening), making a single copy (dereliction), duplicate . . . the machine (contraction)

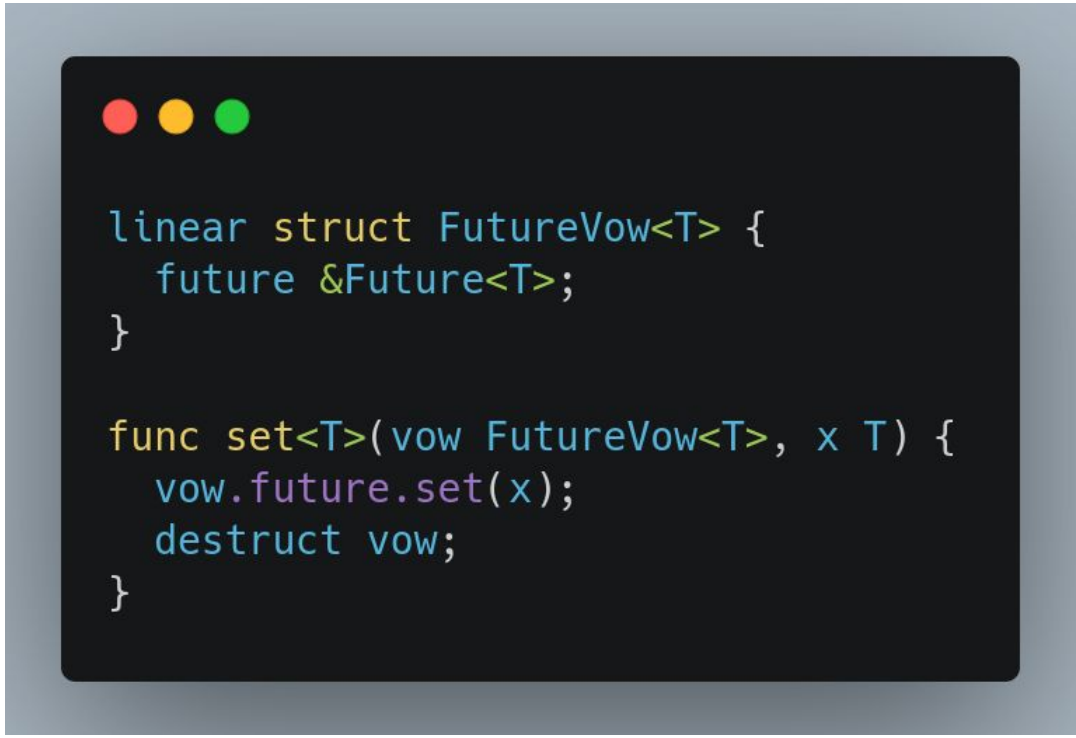
It is no wonder that the first relation of linear logic to computer science was the relation to memory pointed out by Yves Lafont

Linear Types

Linear Types, Higher RAII → Vale, Austral

Haskell Extension:

```
f :: a %1 -> b
data Multiplicity = One | Many
type a %1 -> b = a %One  -> b
type a  -> b = a %Many -> b
f :: (a -> b) %1 -> a -> b
f g x = g x
```



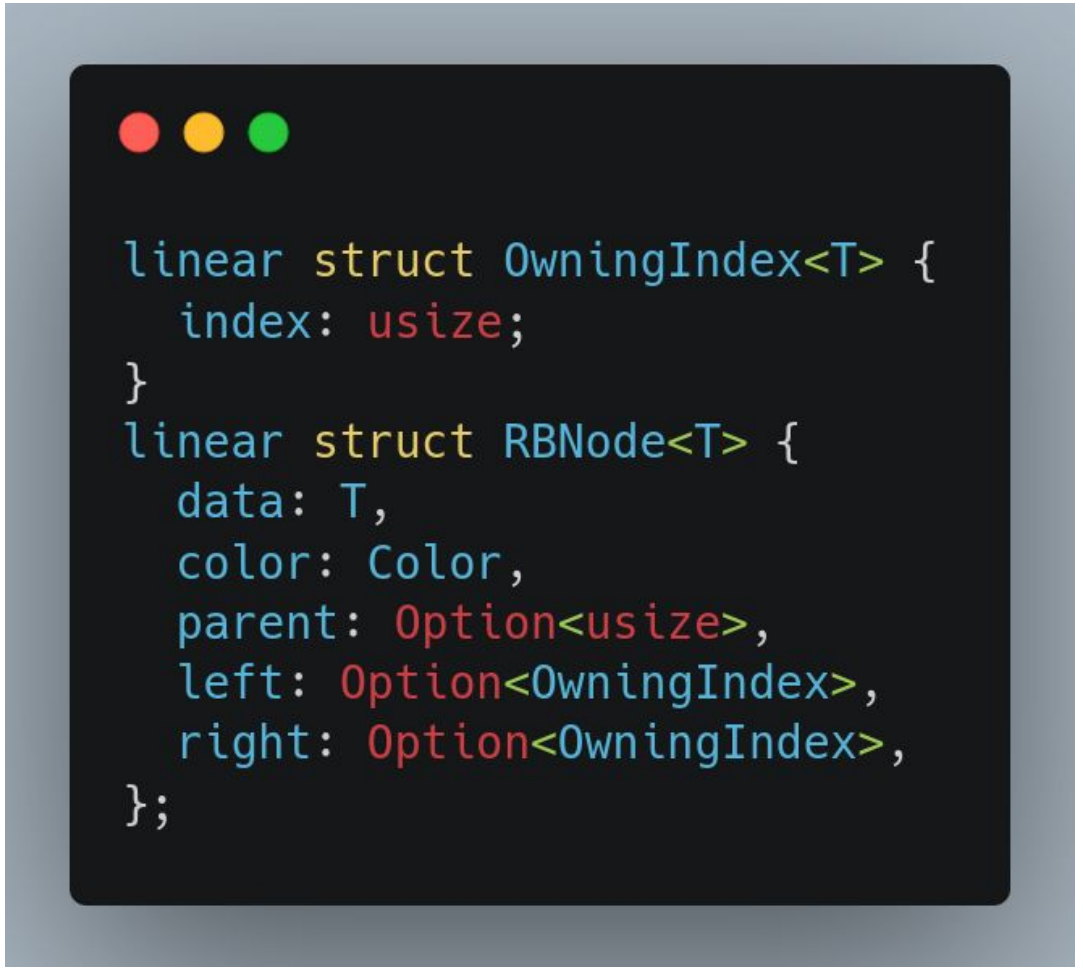
Uses:

- Ressource Interpretation (DB, File handles, …)
- Memory
- Type-State

Linear Rust

```
struct RBNode<T> {
  data: T,
  color: Color,
  parent: Option<u64>,
  left: Option<u64>,
  right: Option<u64>,
};

let tree: HashMap<u64, RBNode<T>> = ...
```



```
linear struct OwningIndex<T> {
  index: usize;
}
linear struct RBNode<T> {
  data: T,
  color: Color,
  parent: Option<usize>,
  left: Option<OwningIndex>,
  right: Option<OwningIndex>,
};
```

Sequent Calculus (klassisch)

$$A_1, \ldots, A_n \vdash B_1, \ldots, B_k,$$

antecedent turnstile sequent formulas

⇒ Wenn alle A_i wahr sind, ist mindestens ein B_i wahr.

Sequent Calculus (klassisch)

Weakening

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}$$

Contraction

$$\frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta} \qquad \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}$$

Linear sequent calculus

$$\frac{}{\vdash A, A^{\perp}} \quad (identity) \qquad \frac{\vdash \Gamma, A \quad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta} \quad (cut)$$

$$\frac{\vdash \Gamma}{\vdash \Gamma'} \quad (exchange : \Gamma' \ is \ a \ permutation \ of \ \Gamma)$$

$$\frac{}{\vdash \mathbf{1}} \quad (one) \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp} \quad (false)$$

$$\frac{\vdash \Gamma, A \quad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta} \; (times)$$

$$\frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \& B} \; (with)$$

$$\frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B} \; (par)$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B} \; (left\ plus)$$

$$\frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B} \; (right\ plus)$$

Logic

$$\frac{}{\vdash A, A^\perp}\ (identity) \qquad \frac{\vdash \Gamma, A \quad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta}\ (cut)$$

$$\frac{\vdash \Gamma}{\vdash \Gamma'}\ (exchange : \Gamma' \text{ is a permutation of } \Gamma)$$

$$\begin{aligned}
\mathbf{1}^\perp &:= \bot & \bot^\perp &:= \mathbf{1} \\
\top^\perp &:= \mathbf{0} & \mathbf{0}^\perp &:= \top \\
(p)^\perp &:= p^\perp & (p^\perp)^\perp &:= p \\
(A \otimes B)^\perp &:= A^\perp \parr B^\perp & (A \parr B)^\perp &:= A^\perp \otimes B^\perp \\
(A \& B)^\perp &:= A^\perp \oplus B^\perp & (A \oplus B)^\perp &:= A^\perp \& B^\perp \\
(!A)^\perp &:= ?A^\perp & (?A)^\perp &:= !A^\perp \\
(\forall x\, A)^\perp &:= \exists x\, A^\perp & (\exists x\, A)^\perp &:= \forall x\, A^\perp
\end{aligned}$$

$$A \multimap B := A^\perp \parr B$$

$$\frac{}{\vdash \mathbf{1}}\ (one) \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \bot}\ (false)$$

$$\frac{\vdash \Gamma, A \quad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}\ (times) \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\ (par)$$

$$\frac{}{\vdash \Gamma, \top}\ (true) \qquad (no\ rule\ for\ zero)$$

$$\frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \& B}\ (with) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}\ (left\ plus) \qquad \frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\ (right\ plus)$$

$$\frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\ (of\ course) \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\ (weakening)$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\ (dereliction) \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\ (contraction)$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \forall x\, A}\ (for\ all : x\ is\ not\ free\ in\ \Gamma) \qquad \frac{\vdash \Gamma, A[t/x]}{\vdash \Gamma, \exists x\, A}\ (there\ is)$$

Linear Logic Propositions as Session Types

In logic, the cut rule allows us to reason using lemmas. A proof of C (the theorem) is well-formed if it is obtained by the composition of a proof of C under the assumption of A (the lemma) and a proof of A. In linear logic, the cut rule is written as:

$$\frac{\Gamma; \Delta \vdash D : A \quad \Gamma; \Delta', x{:}A \vdash E : C}{\Gamma; \Delta, \Delta' \vdash \mathsf{cut}\ D\ (x.\, E) : C}\ \mathsf{cut}$$

In essence, cut allows us to compose two proofs – one providing A and the other one using A to provide C. This principle of composition is captured in the process interpretation as follows:

$$\frac{\Gamma; \Delta \vdash D \rightsquigarrow P :: x : A \quad \Gamma; \Delta', x{:}A \vdash E \rightsquigarrow Q :: z : C}{\Gamma; \Delta, \Delta' \vdash \mathsf{cut}\ D\ (x.\, E) \rightsquigarrow (\nu x)(P \mid Q) :: z : C}\ \mathsf{Tcut}$$

The process P implements session A along channel x, while process Q implements session C along channel z, under the assumption that a session of type A is available on x. Furthermore, since we follow a linear typing discipline, Q requires *all* the behavior supplied by P along x and therefore composing the two processes must necessarily restrict the scope of x to the two processes.

Session Types in Rust

Protocol adherence – Expectations delivered, obligations fulfilled.

Deadlock freedom – Cyclic communication is statically ruled out.

```
type Calculator = Send<i64, Send<Op, Send<i64, Recv<i64>>>>;
enum Op { Plus, Times }
type User = Dual<Calculator>;  // Recv<i64, Recv<Op, Recv<i64, Send<i64>>>>
```

Proof nets

There is an algorithm transforming any proof of a sequent $\vdash \Gamma$ in linear logic into a cut-free proof of the same sequent.

$$\dfrac{\dfrac{\vdash \Gamma, A}{\vdash \Gamma', A}\,(r) \qquad \dfrac{\vdash A^{\perp}, \Delta}{\vdash A^{\perp}, \Delta'}\,(s)}{\vdash \Gamma', \Delta'}\,(cut)$$

$$\dfrac{\dfrac{\dfrac{\vdash \Gamma, A \quad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}\,(cut)}{\vdash \Gamma', \Delta}\,(r)}{\vdash \Gamma', \Delta'}\,(s)$$

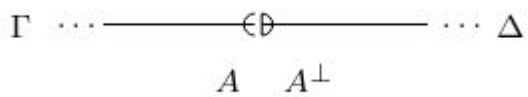
$$\dfrac{\dfrac{\dfrac{\vdash \Gamma, A \quad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}\,(cut)}{\vdash \Gamma, \Delta'}\,(s)}{\vdash \Gamma', \Delta'}\,(r)$$

Limitations of natural deduction

$$\frac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{A \multimap B}\ (\multimap\text{-}intro) \qquad \frac{A \quad A \multimap B}{B}\ (\multimap\text{-}elim)$$

$$\frac{A \qquad B}{A \otimes B}\ (\otimes\text{-}intro) \qquad \frac{A \otimes B \qquad \begin{array}{c}[A][B]\\ \vdots\\ C\end{array}}{C}\ (\otimes\text{-}elim)$$

axiom



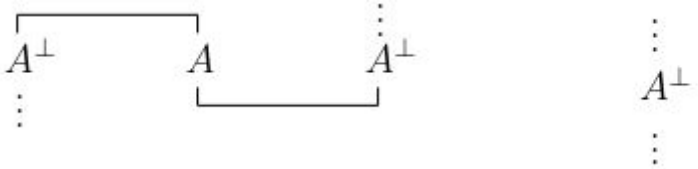
cut

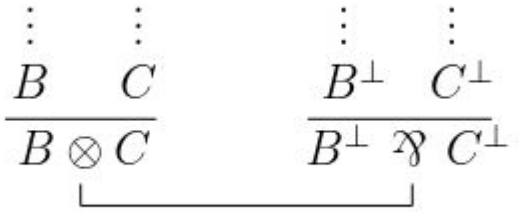
Links

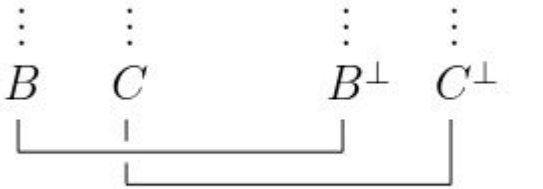
$$\frac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{A \multimap B}\ (\multimap\text{-}intro) \qquad \frac{\begin{array}{cc}A & B\end{array}}{A^\perp \parr B}$$

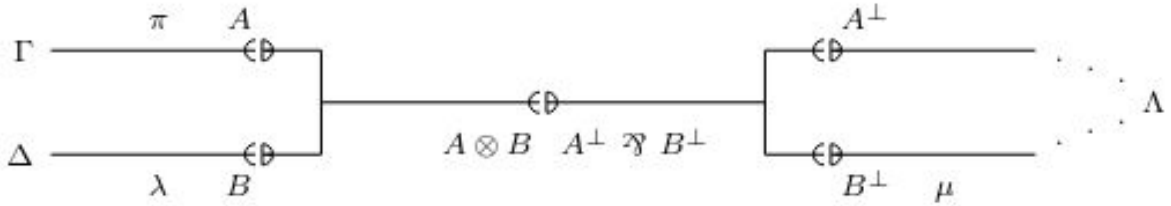
$$\frac{A \quad A \multimap B}{B}\ (\multimap\text{-}elim) \qquad A^\perp \parr B \quad \frac{\begin{array}{cc}A & B^\perp\end{array}}{A \otimes B^\perp} \quad B$$

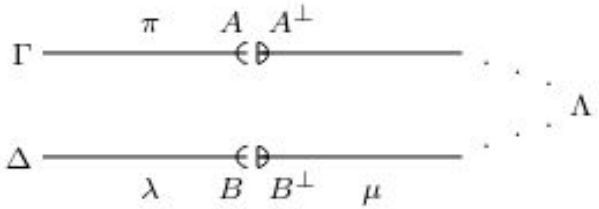
$$\frac{\begin{array}{cc}A \otimes B & \begin{array}{c}[A][B]\\ \vdots\\ C\end{array}\end{array}}{C}\ (\otimes\text{-}elim) \qquad \frac{\begin{array}{cc}A^\perp & B^\perp\end{array}}{A \otimes B \quad A^\perp \parr B^\perp} \quad \begin{array}{cc}A & B\\ \vdots & \vdots\\ C & \end{array}$$


$A^\perp$
A
$A^\perp$
$A^\perp$


B
C
$B \otimes C$
$B^\perp$
$C^\perp$
$B^\perp \parr C^\perp$


B
C
$B^\perp$
$C^\perp$


π
A
Γ
Δ
λ
B
$A \otimes B$
$A^\perp \parr B^\perp$
$A^\perp$
Λ
$B^\perp$
μ


π
A
$A^\perp$
Γ
Λ
Δ
λ
B
$B^\perp$
μ

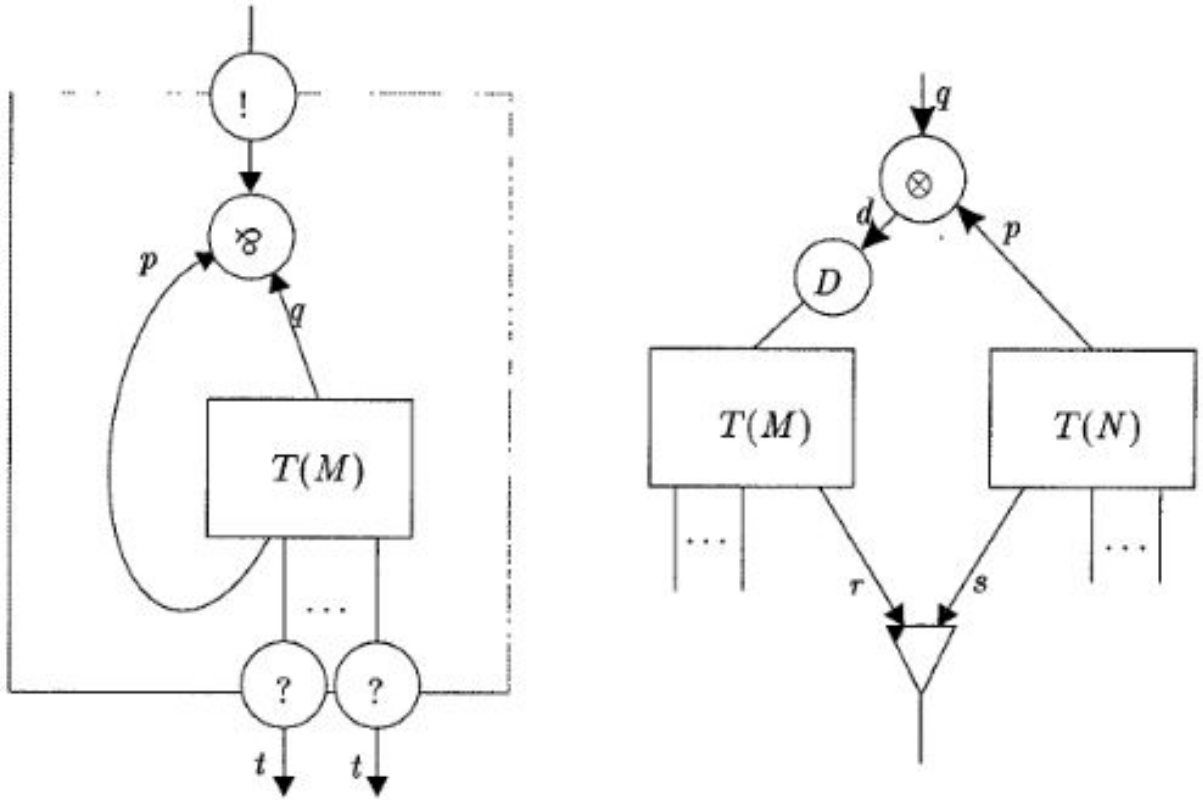


Figure 1: Coding the λ-calculus into Linear Logic proof structures

The Geometry of Interaction Machine

Ian Mackie*
Department of Computing
Imperial College of Science, Technology and Medicine
London SW7 2BZ England
im@doc.ic.ac.uk

⅋ means Parallel, Linear Logic Proofs as Concurrent Functional Programs

$$
\begin{array}{lll}
u, v ::= & \circ & \text{(terminated process)} \\
& \mid \overline{x}.\, u & \text{(output channel } \overline{x} \text{ with continuation)} \\
& \mid \overline{x}\,\overline{y}\; u & \text{(output channels } \overline{x}, \overline{y} \text{ without continuation)} \\
& \mid u \mid v & \text{(parallel composition)} \\
& \mid \mathsf{close}(u) & \text{(process termination)} \\
& \mid x & \text{(variable)} \\
& \mid u\, v & \text{(function application)} \\
& \text{syntactic sugar} & \\
& \mid \lambda x\, u := \overline{x}.\, u & \text{(function definition (provided } x \text{ occurs in } u\text{))} \\
& \mid \overline{x} v.u := (\overline{x}.\, u)v &
\end{array}
$$

$$\frac{\Gamma, x : A \Rightarrow t : B, \Delta}{\Gamma \Rightarrow \lambda x\, t : A \multimap B, \Delta} \multimap \mathrm{I} \text{ (if } x \text{ occurs in } t\text{)}$$

$$\frac{\Gamma, x : A \Rightarrow t : B, \Delta}{\Gamma \Rightarrow \overline{x}.\, t : A \multimap B, \Delta} \multimap \mathrm{I} \text{ (if } x \text{ occurs in } \Delta\text{)}$$

Discussion Points

- Laut Girard und, auch vielen anderen, eignet sich Linear Logic für Concurrency, aber auch Go und Rust kennen Konzepte wie Session Types höchstens am Rande
  - Liegt das daran, dass noch nicht alles geklärt ist?
- Haben Proof Nets noch mehr Potential?

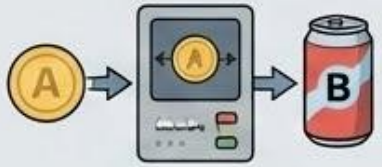
- IMALL with a Mixed-State Modality: A Logical Approach to Quantum Computation
-

LINEAR LOGIC: SYNTAX & INTUITION

IMPLICATION: "ENTAILS" / "ERMÖGLICHT"

Symbol

$$A \multimap B$$



Intuition

Resource Consumption: Consumes resource A to produce resource B. A one-time transformation.

Rules

$$\frac{\Gamma, B \vdash \Delta}{\Gamma, A \multimap B, A \vdash \Delta}$$

If B is in the context, it can be replaced by (A ⊸ B) and A.

Right Rule

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}$$

If A in the context allows proving B, then A ⊸ B is provable from the context.

MULTIPLICATIVE CONJUNCTION (TIMES): "AND" / "SOWOHL ALS AUCH"

Symbol

$$A \otimes B$$

Intuition

Simultaneous Resources: Both resources A and B are available and can be used independently but concurrently.

Rules

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}$$

If A and B are in the context, they can be combined into A ⊗ B.

Right Rule

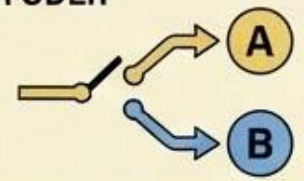
$$\frac{\Gamma \vdash A; \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}$$

If A is provable from Γ and B from Δ, then A ⊗ B is provable from the combined context Γ, Δ.

ADDITIVE CONJUNCTION (WITH): "EITHER OR" / "ENTWEDER ODER"

Symbol

$$A \,\&\, B$$



Intuition

Choice of Resources: One resource is available, and you choose whether to use it as A or as B. Not both.

Rules

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, A \,\&\, B \vdash \Delta} \qquad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \,\&\, B \vdash \Delta}$$

If A is in the context, it can be replaced by A & B (choosing A).

If B is in the context, it can be replaced by A & B (choosing B).

Right Rule

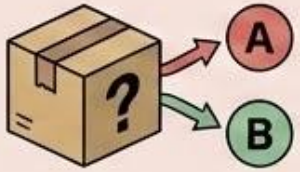
$$\frac{\Gamma \vdash A; \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}$$

If both A and B are provable from the same context Γ, then A & B is provable from Γ.

ADDITIVE DISJUNCTION (PLUS): "PLUS" / "PLUS"

Symbol

$$A \oplus B$$



Intuition

External Choice: You receive either resource A or resource B, but you don't get to choose which one.

Rules

$$\frac{\Gamma, A \vdash \Delta; \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta}$$

If Δ is provable from Γ, A and alse from Γ, B, then it's provable from Γ, A ⊕ B.

Right Rule 1

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}$$

If A is provable from Γ, then A ⊕ B is also provable from Γ.

Right Rule 2

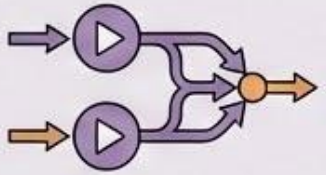
$$\frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}$$

If B is prevable from Γ, then A ⊕ B is also provable from Γ.

MULTIPLICATIVE DISJUNCTION (PAR): "PARALLEL" / "PARALLEL"

Symbol

$$A \parr B$$



Intuition

Parallel Computation: Represents the parallel execution of processes A and B, or the potential to use A to satisfy a demand for B's dual, and vice-versa.

Rules

$$\frac{\Gamma, A \vdash \Delta; \Gamma, B \vdash \Delta}{\Gamma, A \parr B \vdash \Delta}$$

If Δ is provable from Γ, A and from Γ, B, then it's provable from Γ, A ⅋ B.

Rules

$$\frac{\Gamma \vdash A, B}{\Gamma \vdash A \parr B}$$

If both A and B are provable from the context Γ, then A ⅋ B is provable from Γ.

EXPONENTIALS: "OF COURSE" / "WHY NOT"

Symbols

$$!A, \; ?A$$

Intuition

Controlled Resources: !A means 'A' can be used any number of times (duplication/contraction). ?A means 'A' might be needed any number of times.

Dereliction

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta}$$

!A can be used as a single instance of A.

Weakening

$$\frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta}$$

!A can be discarded/ignored.

Contraction

$$\frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta}$$

Two copies of !A can be combined into one.

Promotion

$$\frac{!\Gamma \vdash A}{!\Gamma \vdash !A}$$

If A is provable from a context where all assumptions are exponentials, then !A is provable.

28.11.2025
Linear Logic: Its Syntax (and Semantics)
Jean-Yves Girard, 1995

Locus solum: from the rules of logic to the logic of rules

Ludics

• COMPUTER SCIENCE

Although man-made, computer science is the physics of logicians. In the beginning :

★ Many computer scientists didn't realise the unfeasibility of the halting problem... with as result the building of various paralogics.

★ More educated people were still paying too much attention to formal issues, with an excessive —and surrealistic— emphasis on consistency, and the building of too many Broccoli logics.

In general computers prompted a renewal of positivistic nonsense, artificial intelligence and so on. But how unfair it would be to reduce computer science to these archaic mistakes. It is an immense source of intuitions, let us mention non-determinism, locations, proof-search, streams, process algebras... not to speak of the mere idea of interactivity. Without computer science, would there still be any room left for logic ?

See : **Artifical intelligence, Broccoli logics, Consistency, Explicitation, Halting problem, Interactivity, Locus, Logic, Operational semantics, Paralogics, Process algebra, Proof-search, Stream.**

SKUNK

$\frac{}{\xi \vdash \Lambda}^{(\xi,\varnothing)}$

DAIMON

$\frac{}{\vdash \Lambda}^{\dagger}$