Sequent Calculus:
An Important Connection Between Proofs and Programs

Karim Nour

Université Savoie Mont Blanc - LAMA - LIMD

Séminaires de LIM - Université de la Réunion - Décembre 2024

Outline

Outline

Introduction

- Paradoxes emerged at the beginning of the last century : Russell, Tarski, ...
- Several efforts were made to formalize theories :
  arithmetic (Peano),
  set theory (Zermelo-Fraenkel),
  geometry (Hilbert), ...
- Development of logical systems :
  Hilbert system (Hilbert),
  natural deduction (Gentzen),
  sequent calculus (Gentzen), ...
- We will explore two of these logical systems.

Particular case

- To simplify the presentation, we focus on propositional logic.
- This framework is both less complex and less powerful due to the decidability of propositional logic.
- It cannot express significant mathematical properties or generate complex programs.
- Extending to second-order logic enables more expressive properties and supports richer program generation.

Outline

Propositionnal logic

- Propositional logic is a formal system that focuses on propositions, which are statements that can be evaluated as either true or false.

- We consider a set of atomic propositions $\mathcal{V} = \{a, b, c, \ldots\}$ and a constant $\bot$ for contradiction or false statement.

- A propositional formula is built from atomic propositions and $\bot$ combined with logical connectives : $\neg$, $\rightarrow$, $\wedge$ and $\vee$.

Tautology

- A tautology is a propositional formula that is always true, regardless of the truth values of its components.

A	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$
1	1	1	1	1
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

A	$\neg A$
1	0
0	1

$\bot$
0

Examples : $\neg(a \wedge b) \rightarrow (\neg a \vee \neg b)$

$$((a \rightarrow b) \rightarrow a) \rightarrow a$$

a	b	$\neg(a \wedge b)$	$\neg a \vee \neg b$	$\neg(a \wedge b) \rightarrow (\neg a \vee \neg b)$
1	1	0	0	1
1	0	1	1	1
0	1	1	1	1
0	0	1	1	1

a	b	$a \rightarrow b$	$(a \rightarrow b) \rightarrow a$	$((a \rightarrow b) \rightarrow a) \rightarrow a$
1	1	1	1	1
1	0	0	1	1
0	1	1	0	1
0	0	1	0	1

Questions

- Checking if a formula is a tautology requires exponential time to check all truth assignments to its propositional variables.
- We do not perform real mathematical proofs using truth tables.
- Can tautologies be generated through reasoning using formal systems ?
- Are the systems that allow this truly different from one another ?
- How can they be compared, and why would one be preferred over another ?

The choice of systems

- We choose two logical systems introduced by Gentzen : natural deduction and sequent calculus.
- The rules of natural deduction are intuitive and resemble what we do to produce mathematical proofs.
- However, the dynamics of the calculus for simplifying proofs in natural deduction are quite technical.
- The rules of sequent calculus are less intuitive, but they allow for a more symmetric system.
- However, the dynamics of the calculus for simplifying proofs in sequent calculus are more natural.

Outline

Hilbert system (axioms)

ax1	$A \to (B \to A)$
ax2	$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
ax3	$\neg A \to (A \to \bot)$
ax4	$(A \to \bot) \to \neg A$
ax5	$\neg\neg A \to A$
ax6	$A \wedge B \to A$
ax7	$A \wedge B \to B$
ax8	$A \to (B \to A \wedge B)$
ax9	$A \to A \vee B$
ax10	$A \to B \vee A$
ax11	$A \vee B \to ((A \to C) \to ((B \to C) \to C))$

$$\frac{A \to B \qquad A}{B} \quad \text{modus ponens}$$

Example : $A \to A$

ax1 $(A \to (A \to A)) \to A$ (1)

ax1 $A \to (A \to A)$ (2)

ax2 $(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))$ (3)

MP $(A \to (A \to A)) \to (A \to A)$ (3)+(1) (4)

MP $A \to A$ (4)+(2)

Property

Theorem (Completeness Theorem)

We have $\vdash_{\mathrm{HS}} A$ *if and only if* A *is a tautology*.

The system is not practical for producing mathematical proofs.

Outline

Naturel deduction

- We have two types of inference rules :
  – introduction rules (how to prove a formula)
  – elimination rules (how to use a hypothesis).

- We derive sequents of the form $\Gamma \vdash A$, meaning that A is provable using the hypotheses from the set of formulas Γ.

- We start with the axioms $A \vdash A$ and alternate between introduction and elimination rules to reach the desired conclusion

- This system is very interesting from a pedagogical standpoint, as it helps teach our students how to search for and construct a proof.

Natural deduction rules

$$\frac{}{A \vdash A}\ \text{ax} \qquad\qquad \frac{\Gamma \vdash A}{\Gamma, B \vdash A}\ \text{weak}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \to_i$$

$$\frac{\Gamma \vdash A \to B \qquad \triangle \vdash A}{\Gamma, \triangle \vdash B}\ \to_e$$

Natural deduction rules

$$\frac{\Gamma \vdash A_1 \qquad \triangle \vdash A_2}{\Gamma, \triangle \vdash A_1 \wedge A_2} \wedge_i \qquad \frac{\Gamma \vdash A_1 \wedge A_2}{\Gamma \vdash A_j} \wedge_e$$

$$\frac{\Gamma \vdash A_j}{\Gamma \vdash A_1 \vee A_2} \vee_i$$

$$\frac{\Gamma \vdash A_1 \vee A_2 \qquad \triangle_1, A_1 \vdash C \qquad \triangle_2, A_2 \vdash C}{\Gamma, \triangle_1, \triangle_2 \vdash C} \vee_e$$

Natural deduction rules

$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\ \neg i$$

$$\frac{\Gamma \vdash \neg A \qquad \triangle \vdash A}{\Gamma, \triangle \vdash \bot}\ \neg e$$

$$\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\ \mathrm{PC}$$

Example : $\neg(A \wedge B) \rightarrow (\neg A \vee \neg B)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{B \vdash B}}{A, B \vdash A \wedge B} \quad \dfrac{}{\neg(A \wedge B) \vdash \neg(A \wedge B)}}{A, B, \neg(A \wedge B) \vdash \bot}}{A, \neg(A \wedge B) \vdash \neg B}}{A, \neg(A \wedge B) \vdash \neg A \vee \neg B} \quad \dfrac{}{\neg(\neg A \vee \neg B) \vdash \neg(\neg A \vee \neg B)}}{A, \neg(A \wedge B), \neg(\neg A \vee \neg B) \vdash \bot}}{\neg(A \wedge B), \neg(\neg A \vee \neg B) \vdash \neg A}}{\neg(A \wedge B), \neg(\neg A \vee \neg B) \vdash \neg A \vee \neg B} \quad \dfrac{}{\neg(\neg A \vee \neg B) \vdash \neg(\neg A \vee \neg B)}}{\neg(A \wedge B), \neg(\neg A \vee \neg B) \vdash \bot}}{\neg(A \wedge B) \vdash \neg A \vee \neg B}\ \text{PC}}{\vdash_{\text{ND}} \neg(A \wedge B) \rightarrow (\neg A \vee \neg B)}$$

Example : $((A \to B) \to A) \to A$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{\neg A \vdash \neg A}}{A, \neg A \vdash \bot}}{A, \neg A, \neg B \vdash \bot}}{A, \neg A \vdash B}\,\text{PC}}{\neg A \vdash A \to B} \quad \dfrac{}{(A \to B) \to A \vdash (A \to B) \to A}}{(A \to B) \to A, \neg A \vdash A} \quad \dfrac{}{\neg A \vdash \neg A}}{\dfrac{\dfrac{(A \to B) \to A, \neg A \vdash \bot}{(A \to B) \to A \vdash A}\,\text{PC}}{\vdash_{\text{ND}} ((A \to B) \to A) \to A}}$$

Example : $\neg(P \wedge Q) \iff (P \to \neg Q)$

We prove two implications ($\wedge_i$).

- $\neg(P \wedge Q) \to (P \to \neg Q)$:

  Assume $\neg(P \wedge Q)$, P, and Q, and seek a contradiction ($\to_i$, $\to_i$, $\neg_i$). From P and Q, we obtain $P \wedge Q$ ($\wedge_i$). Since we have $\neg(P \wedge Q)$, we reach a contradiction ($\neg_e$).

- $(P \to \neg Q) \to \neg(P \wedge Q)$:

  Assume $P \to \neg Q$ and $P \wedge Q$, and seek a contradiction ($\to_i$, $\neg_i$). From $P \wedge Q$, we obtain P ($\wedge_e$), and using $P \to \neg Q$, we find $\neg Q$ ($\to_e$). Since we have Q and $\neg Q$, we reach a contradiction ($\neg_e$).

We believe that this type of formalization is more interesting than truth tables and can help students search for and write proofs for other properties.

Properties

Theorem (Completeness Theorem)

We have $\vdash_{\mathrm{ND}} A$ *if and only if* A *is a tautology.*

- How to search for a derivation of a tautology ?
- To search for a derivation of the formula A, can we restrict ourselves solely to the subformulas of A (and their negations) ?

Theorem (Subformula Property)

Natural deduction admits the subformula property.

Subformula Property

- We proceed by eliminating undesirable phenomena known as cuts.

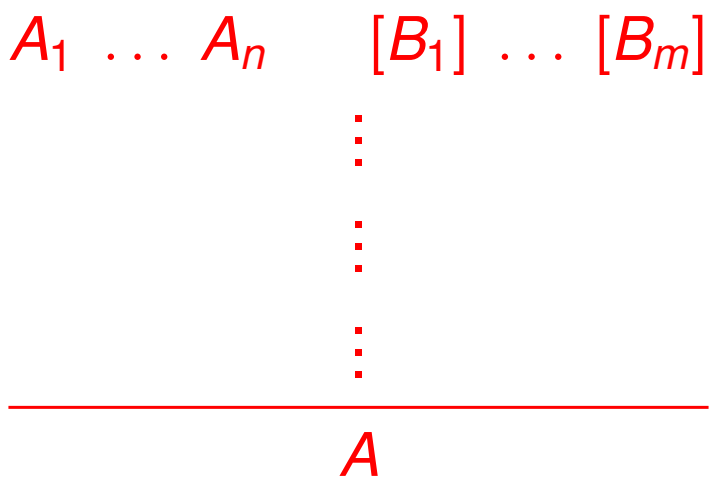
- We have several kinds of cuts :
  - An introduction rule followed by an elimination rule of the same connective.
  - The rule $\vee_e$ followed by an elimination rule.
  - The rule PC followed by an elimination rule.

Subformula Property

- The presentation of the proof rules also needs to be changed to make the proof more comprehensible.
- The proof of this property is very technical, and an improved and stronger version has been discovered recently.
- It is the theorem of cut elimination that establishes a dynamic aspect of the calculus, which can subsequently be used to define a programming language.

New presentation

We represent the proof of the sequent "$A_1, \ldots, A_n \vdash A$" by :



where $[B_1]$ ... $[B_m]$ are auxiliary hypotheses.

New deduction rules

$$\frac{A}{A}\ \text{ax} \qquad \frac{\begin{array}{c}\vdots\\ A\quad B\end{array}}{A}\ \text{weak}$$

$$\frac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{A \to B}\ \to i \qquad \frac{\begin{array}{cc}\vdots & \vdots\\ A \to B & A\end{array}}{B}\ \to e$$

New deduction rules

$$\frac{\begin{array}{cc}\vdots & \vdots \\ A & B\end{array}}{A \wedge B}\ \wedge_i \qquad \frac{\begin{array}{c}\vdots \\ A \wedge B\end{array}}{A}\ \wedge_e \qquad \frac{\begin{array}{c}\vdots \\ A \wedge B\end{array}}{B}\ \wedge_e$$

$$\frac{\begin{array}{c}\vdots \\ A\end{array}}{A \vee B}\ \vee_i \qquad \frac{\begin{array}{c}\vdots \\ B\end{array}}{A \vee B}\ \vee_i \qquad \frac{\begin{array}{ccc} & [A] & [B] \\ \vdots & \vdots & \vdots \\ A \vee B & C & C\end{array}}{C}\ \vee_e$$

New deduction rules

$$\dfrac{\begin{array}{c}[A]\\ \vdots\\ \bot\end{array}}{\neg A}\ \neg i \qquad\qquad \dfrac{\begin{array}{cc}\vdots & \vdots\\ \neg A & A\end{array}}{\bot}\ \neg e$$

$$\dfrac{\begin{array}{c}[\neg A]\\ \vdots\\ \bot\end{array}}{A}\ \mathrm{PC}$$

Example : $((A \to B) \to A) \to A$

$$\dfrac{\dfrac{\dfrac{[A]_{\,2} \quad [\neg A]_{\,3}}{\bot}}{B}_{\,1}}{A \to B}_{\,2} \qquad \dfrac{\dfrac{\dfrac{[(A \to B) \to A]_{\,4}}{A} \qquad [\neg A]_{\,3}}{\bot}}{A}_{\,3}}{((A \to B) \to A) \to A}_{\,4}$$

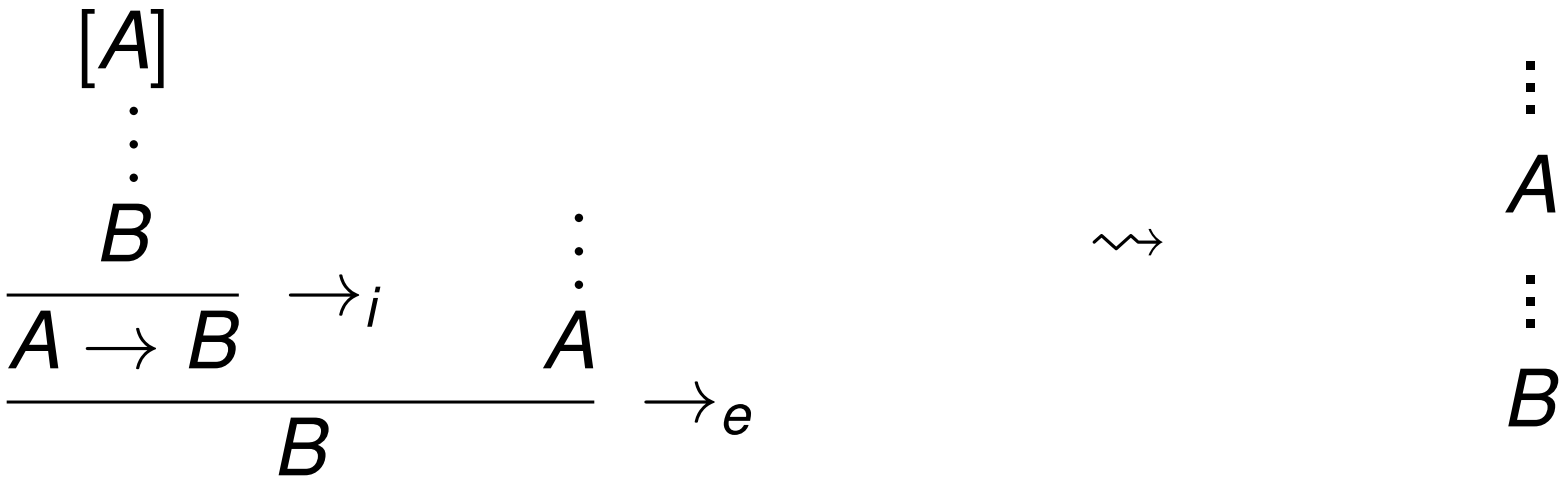
Example : $A \vee \neg A$

$$\dfrac{\dfrac{\dfrac{[A]\,1}{A \vee \neg A} \qquad [\neg(A \vee \neg A)]\,2}{\bot}}{\neg A}\,1$$

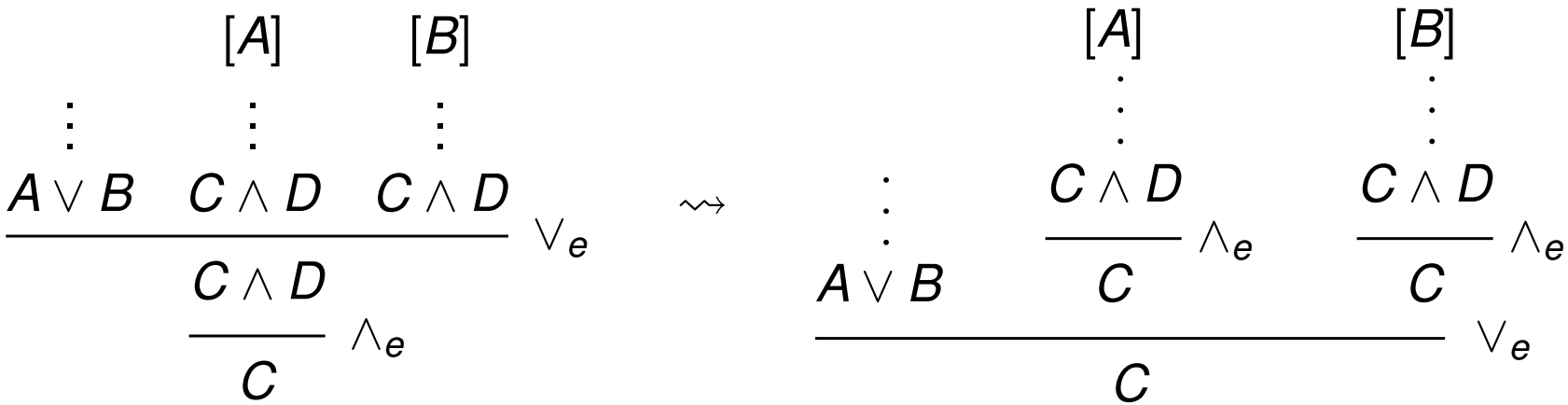
$$\dfrac{\dfrac{\neg A}{A \vee \neg A} \qquad [\neg(A \vee \neg A)]\,2}{\bot}$$

$$\dfrac{\bot}{A \vee \neg A}\,2$$

Example of Cuts



Example of Cuts



Example of Cuts

$$\dfrac{\dfrac{\dfrac{[\neg(A \to B)]}{\vdots}}{\dfrac{\bot}{A \to B}\,\mathrm{PC}} \qquad \dfrac{}{\vdots}}{B}\,{\to_e} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\dfrac{[A \to B] \qquad \dfrac{}{\vdots}}{B}\,{\to_e}}{\bot}\,{\neg_e}}{\neg(A \to B)}\,{\neg_i}}{\dfrac{\dfrac{}{\vdots}}{\dfrac{\bot}{B}\,\mathrm{PC}}}$$

Properties

- We can eliminate all cuts.
- We obtaine a weak cut elimination result.
- We deduce the subformula property.
- The proof is technical and a strong version was discovered recently : de Groote (2001), David & Nour (2003), ...
- The new proofs involve encoding demonstrations as objects that will be considered as programs and translating cut eliminations as rewrite rules.

Outline

Sequent calculus

- We have two types of inference rules :
  – left introduction rules,
  – right introduction rules.

- We derive sequents of the form $\Gamma \vdash \triangle$, meaning that the disjunction of the formulas in the set $\triangle$ is provable using the conjunction of the formulas in the set Γ.

- We start with the axioms $A \vdash A$ and alternate between left introduction and right introduction rules to reach the desired conclusion

- This system has only introduction rules, but it also possesses an explicit cut rule (which is very useful, but we can eliminate it).

- The system is symmetric, but it is not intuitive to use for making proofs.

Sequent calculus rules

$$\frac{}{A \vdash A}\ \text{ax}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ \text{weak}_g \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}\ \text{weak}_d$$

$$\frac{\Gamma_1 \vdash A, \Delta_1 \qquad \Gamma_2, B \vdash \Delta_2}{\Gamma_1, \Gamma_2, A \to B \vdash \Delta_1, \Delta_2}\ \to g$$

$$\frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \to B, \Delta}\ \to d$$

Sequent calculus rules

$$\frac{\Gamma, A_1, A_2 \vdash \Delta}{\Gamma, A_1 \wedge A_2 \vdash \Delta} \wedge_g$$

$$\frac{\Gamma_1 \vdash A_1, \Delta_1 \qquad \Gamma_2 \vdash A_2, \Delta_2}{\Gamma_1, \Gamma_2 \vdash A_1 \wedge A_2, \Delta_1, \Delta_2} \wedge_d$$

$$\frac{\Gamma_1, A_1 \vdash \Delta_1 \qquad \Gamma_2, A_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2, A_1 \vee A_2 \vdash \Delta_1, \Delta_2} \vee_g$$

$$\frac{\Gamma \vdash A_1, A_2, \Delta}{\Gamma \vdash A_1 \vee A_2, \Delta} \vee_d$$

Sequent calculus rules

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}\ \neg g$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}\ \neg d$$

$$\frac{\Gamma_1, A \vdash \Delta_1 \qquad \Gamma_2 \vdash A, \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ \text{cut}$$

Other rules

In case we are working with ordered multisets :

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ \mathrm{contr}_g \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}\ \mathrm{contr}_d$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, B, A \vdash \Delta}\ \mathrm{exch}_g \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash B, A, \Delta}\ \mathrm{exch}_d$$

Example : $\neg(A \wedge B) \rightarrow (\neg A \vee \neg B)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\text{ax} \qquad \dfrac{}{B \vdash B}\,\text{ax}}{A, B \vdash A \wedge B}\,\wedge_d}{A, B, \neg(A \wedge B) \vdash}\,\neg g}{A, \neg(A \wedge B) \vdash \neg B}\,\neg d}{\neg(A \wedge B) \vdash \neg A, \neg B}\,\neg d}{\neg(A \wedge B) \vdash \neg A \vee \neg B}\,\vee_d}{\vdash_{\mathrm{LK}} \neg(A \wedge B) \rightarrow (\neg A \vee \neg B)}\,\rightarrow d$$

Example : $((A \to B) \to A) \to A$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\mathrm{ax}}{A \vdash B, A}\,\mathrm{aff}}{\vdash A \to B, A}\,{\to}_d \qquad \dfrac{}{A \vdash A}\,\mathrm{ax}}{(A \to B) \to A \vdash A}\,{\to}_g}{\vdash_{\mathrm{LK}} ((A \to B) \to A) \to A}\,{\to}_d$$

Example : $A \vee \neg A$

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\mathrm{ax}}{\vdash A, \neg A}\,\neg d}{\vdash_{\mathrm{LK}} A \vee \neg A}\,\vee d$$

Properties

Theorem (Completeness Theorem)

We have $\vdash_{\mathrm{LK}} A$ *if and only if* A *is a tautology*.

Theorem (Cut Elimination Property)

We can eliminate the cut rules from any derivation.

Theorem (Subformula Property)

Sequent calculus admits the subformula property.

Properties

- For the first theorem, we prove that :
  $\vdash_{\mathrm{LK}} A$ if and only if $\vdash_{\mathrm{ND}} A$.

- The proof of the second theorem is achieved through case analysis, identifying the appropriate strategy for eliminating cuts.

- The proof of the third theorem is straightforward.

Cut elimination

- The proof is due to Gentzen and motivates the introduction of the system.

- We begin by generalizing the cut rule to one that allows for the elimination of multiple formulas (possibly zero).

  - Γ_A denotes a multiset obtained by removing certain occurrences of A from Γ.

  - The following rule is called "mix" :

$$\frac{\Gamma \vdash \triangle \quad \Gamma' \vdash \triangle'}{\Gamma_A, \Gamma' \vdash \triangle, \triangle'_A}\ \text{mix}$$

- The cut elimination theorem transforms into a mix elimination theorem.

Mix elimination

- The rank of the mix is the pair (d, h), where :
  – d is the size of the formula,
  – h is its height.

- We prove the following result :

Lemma

If a derivation contains only a single mix as its last rule, then there exists a derivation without any mix that has the same conclusion.

- We proceed by induction on the rank of the mix.

Cut Elimination

- We distinguish several kinds of mixes :
  - The two premises come from a logical rule, and both introduced formulas are active.
  - The two premises come from a logical rule, but one of introduced formulas is inactive.
  - One of the premises comes from a structural rule.
  - One of the premises comes from the axiom rule.

- It is the theorem of cut elimination that establishes a dynamic aspect of the calculus, which can subsequently be used to define a programming language.

Example of mix

$A = B \vee C$

$$\dfrac{\dfrac{\begin{array}{c}\Pi_1\\ \vdots\\ \Gamma, B \vdash \triangle\end{array} \qquad \begin{array}{c}\Pi_2\\ \vdots\\ \Gamma, C \vdash \triangle\end{array}}{\Gamma, B \vee C \vdash \triangle}\vee_g \qquad \dfrac{\begin{array}{c}\Pi_3\\ \vdots\\ \Gamma' \vdash B, C, \triangle'\end{array}}{\Gamma' \vdash B \vee C, \triangle'}\vee_d}{\Gamma_A, \Gamma' \vdash \triangle, \triangle'_A}\,\text{mix}$$

Example of mix

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Pi_1}{\Gamma, B \vdash \Delta} \quad \dfrac{\Pi_2}{\Gamma, C \vdash \Delta}}{\Gamma, B \vee C \vdash \Delta} \vee_g \quad \dfrac{\Pi_3}{\Gamma' \vdash B, C, \Delta'}}{\Gamma_A, \Gamma' \vdash B, C, \Delta, \Delta'_A}\,\mathrm{mix}(1) \quad \dfrac{\dfrac{\dfrac{\Pi_1}{\Gamma, B \vdash \Delta} \quad \dfrac{\Pi_3}{\Gamma' \vdash B, C, \Delta'}}{\Gamma' \vdash B \vee C, \Delta'} \vee_d}{\Gamma_A, \Gamma', B \vdash \Delta, \Delta'_A}\,\mathrm{mix}(2)}{\Gamma_A, \Gamma', \Gamma_A, \Gamma' \vdash C, \Delta, \Delta'_A, \Delta, \Delta'_A}\,\mathrm{mix}(3) \quad \dfrac{\dfrac{\dfrac{\Pi_2}{\Gamma, C \vdash \Delta} \quad \dfrac{\Pi_3}{\Gamma' \vdash B, C, \Delta'}}{\Gamma' \vdash B \vee C, \Delta'} \vee_d}{\Gamma_A, \Gamma', C \vdash \Delta, \Delta'_A}\,\mathrm{mix}(4)}{\Gamma_A, \Gamma', \Gamma_A, \Gamma', \Gamma_A, \Gamma' \vdash \Delta, \Delta'_A, \Delta, \Delta'_A, \Delta, \Delta'_A}\,\mathrm{mix}(5)}{\Gamma_A, \Gamma' \vdash \Delta, \Delta'_A}\,\mathrm{contr}_g, \mathrm{contr}_d$$

Example of mix

The new mixes (1), (2), (3), (4), (5) have respective ranks :

$$(d, h_1), (d, h_2), (d_3, h_3), (d, h_4), (d_5, h_5),$$

where

$$h_1 < h,\ h_2 < h,\ d_3 < d,\ h_4 < h,\ d_5 < d.$$

Therefore, we can apply the induction hypothesis and successively eliminate the mixes (1), (2), (4), (3), (5).

Outline

Combinatory logic

- We consider the implicative part of Hilbert system.
  - $\mathrm{K} : A \to (B \to A)$
  - $\mathrm{S} : (A \to (B \to C)) \to ((A \to B) \to (A \to C))$

$$\frac{U : A \to B \qquad V : A}{(U)V : B}$$

Terms and reduction rules

- **Terms :**

$$V \quad ::= \quad \mathrm{K} \mid \mathrm{S} \mid (V)V$$

- **Reduction rules :**

$$((\mathrm{K})U)V \quad \rightarrow_k \quad U$$

$$(((\mathrm{S})U)V)W \quad \rightarrow_s \quad ((U)W)(V)W$$

Examples

$$E = A \to A$$

$$F = (A \to B) \to ((C \to A) \to (C \to B))$$

$$G = (A \to (B \to C)) \to (B \to (A \to C))$$

$$\mathrm{I} = ((\mathrm{S})\mathrm{K})\mathrm{K}$$

$$\mathrm{P} = ((\mathrm{S})(\mathrm{K})\mathrm{S}))\mathrm{K}$$

$$\mathrm{Q} = ((\mathrm{S})((\mathrm{P})\mathrm{P})\mathrm{S})(\mathrm{K})\mathrm{K}$$

$\mathrm{I} : E$ $\mathrm{P} : F$ $\mathrm{Q} : G$

Remarks

- The calculus is confluent and strongly normalizing.
- The calculus is based on intuitionistic logic.
- The calculus is variable-free but equivalent to λ-calculus.
- The problem for classical logic was resolved in 2006 : the CCL system (Nour).

Outline

Typed system (M. Parigot : 1993)

- We consider the implicative natural deduction.
- **Sequent** : $\Gamma \vdash A$ (Γ is a set of formulas).

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

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\frac{\Gamma, \neg A \vdash A}{\Gamma, \neg A \vdash \bot} \qquad \frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}$$

The typed $\lambda\mu$-calculus

- **Curry-Howard correspondence**

$$\Gamma = x_1 : A_1, \ldots, x_n : A_n, \alpha_1 : \neg B_1, \ldots, \alpha_m : \neg B_m$$

$$\Gamma \vdash t : A$$

- **Typed rules**

$$\frac{}{\Gamma, x : A \vdash x : A}$$

$$\frac{\Gamma, x : A \vdash u : B}{\Gamma \vdash \lambda x.u : A \to B} \qquad \frac{\Gamma \vdash u : A \to B \qquad \Gamma \vdash v : A}{\Gamma \vdash (u)v : B}$$

$$\frac{\Gamma, \alpha : \neg A \vdash u : A}{\Gamma\alpha : \neg A \vdash [\alpha]u : \bot} \qquad \frac{\Gamma, \alpha : \neg A \vdash u : \bot}{\Gamma \vdash \mu\alpha.u : A}$$

Terms and reduction rules

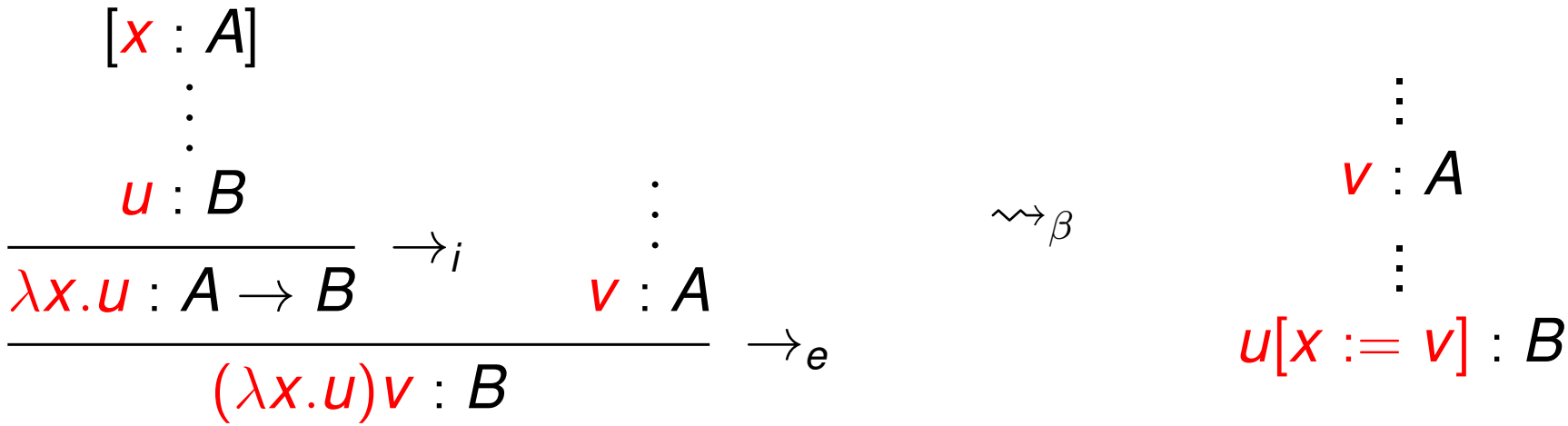
- **Terms :**

$$v \;::=\; x \;\mid\; \lambda x.t \;\mid\; \mu\alpha.t \;\mid\; (t)t \;\mid\; [\alpha]t$$

- **Reduction rules :**

$$\begin{array}{lll}
(\lambda x.u)v & \to_{\beta} & u[x := v] \\
(\mu\alpha.u)v & \to_{\mu} & \mu\alpha.u[[\alpha]w := [\alpha](w)v] \\
(v)\mu\alpha.u & \to_{\mu'} & \mu\alpha.u[[\alpha]w := [\alpha](u)w] \\
[\alpha]\mu\beta.t & \to_{\rho} & t[\beta := \alpha] \\
\mu\alpha.\mu\beta.t & \to_{\varepsilon} & \mu\alpha.t_{\beta} \\
\mu\alpha.[\alpha]t & \to_{\theta} & t \qquad \alpha \notin FV(t)
\end{array}$$

The β-reduction



The μ-reduction

$$\dfrac{\dfrac{\dfrac{[\alpha : \neg(A \to B)] \quad \begin{array}{c}\vdots\\ w : A \to B\end{array}}{[\alpha]w : \bot}\,\neg_e \\ \vdots \\ u[\ldots[\alpha]w\ldots] : \bot}{\mu\alpha.u[\ldots[\alpha]w\ldots] : A \to B}\,\mathrm{PC} \qquad \begin{array}{c}\vdots\\ v : A\end{array}}{(\mu\alpha.u[\ldots[\alpha]w\ldots])v : B}\,\to_e$$

$$\rightsquigarrow_\mu$$

$$\dfrac{\dfrac{[\alpha : \neg B] \qquad \dfrac{\begin{array}{c}\vdots\\ w : A \to B\end{array} \qquad \begin{array}{c}\vdots\\ v : A\end{array}}{(w)v : B}\,\to_e}{[\alpha](w)v : \bot}\,\neg_e \\ \vdots \\ u[\ldots[\alpha](w)v\ldots] : \bot}{\mu\alpha.u[\ldots[\alpha](w)v\ldots] : B}\,\mathrm{PC}$$

Properties

Theorem

The reduction is not confluent (due solely to the μ'-rule).

Theorem

The type is preserved during a reduction.

Theorem

1. *The reduction is weakly normalizing (it may lead to a non-reduced form).*
2. *Without the μ' rule, the reduction is strongly normalizing (every reduction leads to a non-reduced form).*

Programming Theorem

- $N = (a \to (a \to a)) \to a$.
- $\forall n \in \mathbb{N}, \underline{n} = \lambda x.\lambda f.\underbrace{(f)\ldots(f)}_{n \text{ times}}\, x$.

Lemma

1. $\forall n \in \mathbb{N}, \vdash \underline{n} : N$.
2. *If* $\vdash t : N$, *then* $\exists n \in N, \forall x, f \notin Fv(t), ((t)x)f \twoheadrightarrow \underbrace{(f)\ldots(f)}_{n \text{ times}}\, x$.

Theorem

If $\vdash t : N \to N$, *then* $\forall n \in \mathbb{N}, \exists m \in N, \forall x, f \notin Fv(t)$,

$$(((t)\underline{n})x)f \twoheadrightarrow \underbrace{(f)\ldots(f)}_{m \text{ times}}\, x.$$

What can be programmed ?

- Without the classical part (i.e., without the μ and [.]), that is, in λ-calculus, we can program all computable functions.
- In a typed system, we have many fewer programmable functions (it depends on the type system).
- The classical part (i.e., with μ and [.]) only provides additional ways of programming.
- In classical logic, we can prove more formulas.

Example

$$\dfrac{\dfrac{\dfrac{}{x : (A \to B) \to A \vdash x : (A \to B) \to A} \quad \dfrac{\dfrac{\dfrac{}{y : A, \alpha : \neg A, \beta : \neg B \vdash y : A}}{y : A, \alpha : \neg A, \beta : \neg B \vdash [\alpha]y : \bot}}{\alpha : \neg A \vdash \lambda y.\mu\beta.[\alpha]y : A \to B}}{x : (A \to B) \to A, \alpha : \neg A \vdash (x)\lambda y.\mu\beta.[\alpha]y : A}}{x : (A \to B) \to A, \alpha : \neg A \vdash [\alpha](x)\lambda y.\mu\beta.[\alpha]y : \bot}}{x : (A \to B) \to A \vdash \mu\alpha.[\alpha](x)\lambda y.\mu\beta.[\alpha]y : A}}{\vdash \lambda x.\mu\alpha.[\alpha](x)\lambda y.\mu\beta.[\alpha]y : ((A \to B) \to A) \to A}$$

call/CC : "call with current continuation" of Scheme

$$\mathcal{CC} = \lambda x.\mu\alpha.[\alpha](x)\lambda y.\mu\beta.[\alpha]y$$

Let's look at the reduction of $((\mathcal{CC})\lambda k.u)\overline{v}$ depending on :

(1) u does not use k

(2) u uses it by giving $(k)w\overline{w'}$

We obtain :

(1) $((\mathcal{CC})\lambda k.u)\overline{v} \twoheadrightarrow (u)\overline{v}$

(2) $((\mathcal{CC})\lambda k.u)\overline{v} \twoheadrightarrow (w)\overline{v}$

Conclusion

- Natural deduction is a very practical logical system for writing mathematical proofs.
- Natural deduction is not practical enough for establishing metatheorems.
- Natural deduction is very practical for extracting programs in intuitionistic and classical logics.
- Some difficulties in understanding the algorithmic behavior of proofs in classical logic.
- Difficulties in defining “call-by-name” and “call-by-value” strategies.

Outline

Typed system (P.-L. Curien and H. Herbelin : 2000)

- We consider the implicative sequent calculus.
- **Sequent** : $\Gamma \vdash \triangle$ (Γ & $\triangle$ are sets of formulas).

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

$$\frac{\Gamma, A \vdash B, \triangle}{\Gamma \vdash A \to B, \triangle} \qquad \frac{\Gamma \vdash A, \triangle \qquad \Gamma, B \vdash \triangle}{\Gamma, A \to B \vdash \triangle}$$

- **Curry-Howard correspondence**

$$\Gamma = x_1 : A_1, \ldots, x_n : A_n \qquad \triangle = \alpha_1 : B_1, \ldots, \alpha_m : B_m$$

$$\Gamma \vdash \boxed{v : A}, \triangle \qquad \Gamma, \boxed{e : A} \vdash \triangle \qquad c : (\Gamma \vdash \triangle)$$

The typed $\overline{\lambda}\mu\tilde{\mu}$-calculus

$$\frac{}{\Gamma, x : A \vdash \boxed{x : A}, \Delta} \qquad \frac{}{\Gamma, \boxed{\alpha : A} \vdash \alpha : A, \Delta}$$

$$\frac{\Gamma, x : A \vdash \boxed{v : B}, \Delta}{\Gamma \vdash \boxed{\lambda x.v : A \to B}, \Delta} \qquad \frac{\Gamma \vdash \boxed{v : A}, \Delta \qquad \Gamma, \boxed{e : B} \vdash \Delta}{\Gamma, \boxed{v \star e : A \to B} \vdash \Delta}$$

$$\frac{\Gamma \vdash \boxed{v : A}, \Delta \qquad \Gamma, \boxed{e : A} \vdash \Delta}{\langle v, e \rangle : (\Gamma \vdash \Delta)}$$

$$\frac{c : (\Gamma \vdash \alpha : A, \Delta)}{\Gamma \vdash \boxed{\mu\alpha.c : A}, \Delta} \qquad \frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma, \boxed{\tilde{\mu}x.c : A} \vdash \Delta}$$

Other rules

- A complet symetric system.
- **A new connector "$-$"**

  $A - B$ means $A \wedge \neg B$ and is equivalent to $\neg(\neg B \to \neg A)$
- **Two logical rules**

$$\frac{\Gamma \vdash \boxed{v : A}, \triangle \qquad \Gamma, \boxed{e : B} \vdash \triangle}{\Gamma \vdash \boxed{e \bullet v : A - B}, \triangle} \qquad \frac{\Gamma, \boxed{e : A} \vdash \alpha : B, \triangle}{\Gamma, \boxed{\overline{\lambda}\alpha.e : A - B} \vdash \triangle}$$

Terms and reduction rules

- **Terms :**

$$\begin{array}{lcccccccc} c & ::= & & & & & \langle v, e \rangle & & \\ v & ::= & x & | & \lambda x.v & | & \mu\alpha.c & | & e \bullet v \\ e & ::= & \alpha & | & \overline{\lambda}\alpha.e & | & \tilde{\mu}x.c & | & v \star e \end{array}$$

- **Reduction rules :**

$$\begin{array}{lll} \langle \lambda x.v, v' \star e \rangle & \to_{\lambda} & \langle v', \tilde{\mu}x.\langle v, e \rangle \rangle \\ \langle e' \bullet v, \overline{\lambda}\alpha.e \rangle & \to_{\overline{\lambda}} & \langle \mu\alpha.\langle v, e \rangle, e' \rangle \\ \langle \mu\alpha.c, e \rangle & \to_{\mu} & c[\alpha := e] \\ \langle v, \tilde{\mu}x.c \rangle & \to_{\tilde{\mu}} & c[x := v] \\ \mu\alpha.\langle v, \alpha \rangle & \to_{sv} & v \qquad \alpha \notin Fv(v) \end{array}$$

The λ-reduction

$$\dfrac{\dfrac{\vdots}{\Gamma_1 \vdash v' : A, \triangle_1} \qquad \dfrac{\vdots}{\Gamma_2, e : B \vdash \triangle_2}}{\Gamma_1, \Gamma_2, v' \star e : A \to B \vdash \triangle_1, \triangle_2} \qquad \dfrac{\dfrac{\vdots}{\Gamma_3, x : A \vdash v : B, \triangle_3}}{\Gamma_3 \vdash \lambda x.v : A \to B, \triangle_3}$$

$$\langle \lambda x.v, v' \star e \rangle : (\Gamma_1, \Gamma_2, \Gamma_3 \vdash \triangle_1, \triangle_2, \triangle_3)$$

$$\rightsquigarrow_\lambda$$

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma_3, x : A \vdash v : B, \triangle_3} \qquad \dfrac{\vdots}{\Gamma_2, e : B \vdash \triangle_2}}{\langle v, e \rangle : (\Gamma_2, \Gamma_3, x : A \vdash \triangle_2, \triangle_3)}}{\Gamma_2, \Gamma_3, \tilde{\mu}x.\langle v, e \rangle : A \vdash \triangle_2, \triangle_3} \qquad \dfrac{\vdots}{\Gamma_1 \vdash v' : A, \triangle_1}$$

$$\langle v', \tilde{\mu}x.\langle v, e \rangle \rangle : (\Gamma_1, \Gamma_2, \Gamma_3 \vdash \triangle_1, \triangle_2, \triangle_3)$$

Properties

Theorem

The reduction is not confluent (critical pair $\langle \mu\alpha.c_1, \tilde{\mu}x.c_2 \rangle$).

Theorem

The type is preserved during a reduction.

Theorem

The reduction is strongly normalizing (every reduction leads to a non-reduced form).

Theorem

We have a coding of the $\lambda\mu$-calculus into the $\lambda\mu\tilde{\mu}$-calculus and a coding of the $\lambda\mu\tilde{\mu}$-calculus into the $\lambda\mu$-calculus.

Example

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y : A \vdash y : A} \quad \dfrac{}{\alpha : A \vdash \alpha : A, \beta : B}}{\langle y, \alpha \rangle : (y : A \vdash \alpha : A, \beta : B)}}{y : A \vdash \mu\beta.\langle y, \alpha \rangle : B, \alpha : A}}{\vdash \lambda y.\mu\beta.\langle y, \alpha \rangle : A \to B, \alpha : A} \quad \dfrac{}{\alpha : A \vdash \alpha : A}}{(\lambda y.\mu\beta.\langle y, \alpha \rangle) \star \alpha : (A \to B) \to A \vdash \alpha : A}}{\dfrac{\dfrac{\langle x, (\lambda y.\mu\beta.\langle y, \alpha \rangle) \star \alpha \rangle : (x : (A \to B) \to A \vdash \alpha : A)}{x : (A \to B) \to A \vdash \mu\alpha.\langle x, (\lambda y.\mu\beta.\langle y, \alpha \rangle) \star \alpha \rangle : A}}{\vdash \lambda x.\mu\alpha.\langle x, (\lambda y.\mu\beta.\langle y, \alpha \rangle) \star \alpha \rangle : ((A \to B) \to A) \to A}}$$

call/CC : "call with current continuation" of Scheme

$$\mathcal{CC} = \lambda x.\mu\alpha.\langle x, (\lambda y.\mu\beta.\langle y, \alpha\rangle) \star \alpha\rangle$$

Let's look at the reduction of $\langle \mathcal{CC}, e \star v\rangle$.

We have :

(1) $\langle \mathcal{CC}, v \star e\rangle \twoheadrightarrow \langle v, k_e \star e\rangle$

(2) $\langle k_e, v' \star e'\rangle \twoheadrightarrow \langle v', e\rangle$

New presentation

If $\mathcal{F}$ is the set of implicative propositional formulas, we denote $\mathcal{F}^{\perp} = \{A^{\perp} \,/\, A \in \mathcal{F}\}$. We consider the set of formulas :

$$\mathcal{F} \cup \mathcal{F}^{\perp} \cup \{\perp\}$$

$$\frac{(x_i; A_i)_{1 \leq i \leq n} \ldots ([x'_j : A'_j])_{1 \leq j \leq n'} \ldots (\alpha_k : B_k^{\perp})_{1 \leq k \leq m} \ldots ([\alpha'_\ell : B'^{\perp}_\ell])_{1 \leq \ell \leq m'}}{t : A}$$

$$\Gamma \vdash v : A \qquad \Gamma \vdash e : A^{\perp} \qquad \Gamma \vdash c : \perp$$

New deduction rules

$$\frac{x : A}{x : A} \qquad \frac{\alpha : A^{\perp}}{\alpha : A^{\perp}}$$

$$\frac{\begin{array}{c}\vdots\\ t : A \quad y : B\end{array}}{t : A} \qquad \frac{\begin{array}{c}\vdots\\ t : A \quad \alpha : B^{\perp}\end{array}}{t : A}$$

$$\frac{\begin{array}{c}[x : A]\\ \vdots\\ v : B\end{array}}{\lambda x.v : A \to B} \qquad \frac{\begin{array}{cc}\vdots & \vdots\\ v : A & e : B^{\perp}\end{array}}{v \star e : (A \to B)^{\perp}}$$

New deduction rules

$$\frac{\begin{array}{cc}\vdots & \vdots \\ v : A & e : A^{\perp}\end{array}}{\langle v, e\rangle : \perp}$$

$$\frac{\begin{array}{c}[\alpha : A^{\perp}] \\ \vdots \\ c : \perp\end{array}}{\mu\alpha.c : A} \qquad \frac{\begin{array}{c}[x : A] \\ \vdots \\ c : \perp\end{array}}{\tilde{\mu}x.c : A^{\perp}}$$

Programming in $\lambda\mu\tilde{\mu}$-calculus

- Not well studied in the literature.
- We encode $\mu\alpha.\langle u, v \star \alpha\rangle$ as $[u]v$.
  - $N = (a \to (a \to a)) \to a$.
  - $\forall n \in \mathbb{N}, \widehat{n} = \lambda x.\lambda f.\underbrace{[f]\ldots[f]}_{n \text{ times}} x$.
  - $\widehat{s} = \lambda n.\lambda x.\lambda f.\mu\alpha.\langle f, \mu\beta.\langle n, f \star x \star \beta\rangle \star \alpha\rangle$.

Lemma

- $\forall n \in \mathbb{N}, \vdash \widehat{n} : N$.
- $\vdash \widehat{s} : N \to N$.
- $\forall n \in \mathbb{N}, [\widehat{s}]\widehat{n} \twoheadrightarrow \widehat{n+1}$.

Programming in $\lambda\mu\tilde{\mu}$-calculus

- The values

$$\mu\alpha.\langle\lambda x.\mu\beta.\langle\widehat{0},\alpha\rangle,\alpha\rangle$$

$$\lambda x.\lambda f.\mu\alpha.\langle f,(\mu\beta.\langle x,\alpha\rangle)\star\alpha\rangle$$

$$\lambda x.\lambda f.\mu\alpha.\langle f,x\star(\tilde{\mu}y.\langle x,\alpha\rangle)\rangle$$

$$\lambda x.\lambda f.\mu\alpha.\langle f,x\star(\tilde{\mu}y.\langle f,y\star\alpha\rangle)\rangle$$

  have the type N.

- Can we characterize the values of type N?
- How can we find the integer hidden in a value of type N?
- Do we have the equivalent of the rule μ' to do this?
- Do we therefore have a good programming theorem?

Conclusion

- The sequent calculus is not practical enough for making mathematical proofs.
- The sequent calculus is very practical for establishing metatheorems.
- The sequent calculus is not practical for extracting programs.
- Programs extracted from mathematical proofs are very practical to execute.
- Simple definitions of “call-by-name” and “call-by-value” strategies :
  - call-by-value : priority to the μ-reduction
  - call-by-name : priority to the $\tilde{\mu}$-reduction

Other systems

- A completely symmetric system due to F. Barbanera and S. Berardi (1996) : the λ^{Sym}_{Prop}-calculus.

- Many systems based on linear logic, in particular, the system L due to G.Munch-Maccagnoni (2009).

Outline

Motivation

- Linear logic was introduced by J.-Y. Girard in 1987 by examining a semantics (coherence spaces) for typed lambda calculus : $A \to B = !A \multimap B$.
- Linear logic allows for a more precise modeling of resources compared to classical logic.
- It offers an approach to handle situations where the multiple use of a resource is prohibited.

Motivation

- The structural rules of classical logic allow for free manipulation of propositions, which can lead to undesirable uses of resources.

- The contraction rule allows resources to be merged, which is not always desirable.

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

- The weakening rule allows resources to be introduced without restriction, which can distort resource management.

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

Motivation

- This freedom to remove and add formulas allows for different ways of presenting logical rules, particularly with regard to the management of sets of hypotheses.

$$\frac{\Gamma \vdash A, \triangle \qquad \Gamma \vdash B, \triangle}{\Gamma \vdash A \wedge B, \triangle} \qquad \frac{\Gamma_1 \vdash A, \triangle_1 \qquad \Gamma_2 \vdash B, \triangle_2}{\Gamma_1, \Gamma_2 \vdash A \wedge B, \triangle_1, \triangle_2}$$

$$\frac{\Gamma \vdash A, \triangle}{\Gamma \vdash A \vee B, \triangle} \qquad \frac{\Gamma \vdash A, B, \triangle}{\Gamma \vdash A \vee B, \triangle}$$

- Linear logic eliminates duplication and forgetting of resources by imposing strict constraints on their use.

- Linear logic guarantees that each resource is used exactly once.

Introduction

- This focus on structural rules leads to the duplication of the logical connectors $\wedge$ and $\vee$.
- The connector $\rightarrow$ also changes its meaning and enforces the use of the left-hand side exactly once to prove the right-hand side.
- The management of structural rules also imposes modalities to properly analyze the phenomena of duplication and contraction.
- Neutral elements are also introduced for the connectives. We will therefore have two notions of true and two notions of false !

Connectors : and/or

- The $\otimes$ connector represents a strong conjunction. If we have $A \otimes B$, you also have a copy of both A and B.
- The $\&$ connector represents a weak conjunction. If we have $A \,\&\, B$, you can choose either A or B, but not both.
- The $\parr$ connector expresses an additive disjunction. The formula $A \parr B$ means that one of the two formulas is true, but we do not know which one in advance.
- The $\oplus$ connector represents a rigid form of disjunction. The formula $A \oplus B$ means that one of the two formulas is true, but it will always be the same one.

Other connectors

- The $\multimap$ connector represents an implication where the resource on the left side of the implication is consumed exactly once to produce the resource on the right side, also exactly once.
- The formula $!A$ indicates that we have an arbitrary number of copies of A.
- By duality, the formula $?A$ means that we know A is true a certain number of times, but we do not know how many times.
- Each connector "and/or" admits a constant as a neutral element.

Example (Lafont's meal)

$$20\$ \multimap (\text{tomatoes} \oplus \text{lettuce}) \otimes \text{hamburger} \otimes (\text{cheese} \;\&\; \text{ice cream}) \otimes !\text{water}.$$

This means, for exactly 20\$, the customer can have :

- An appetizer of either tomatoes or lettuce, with the choice made by the restaurant.
- A main course of hamburger.
- A dessert of either cheese or ice cream, with the customer choosing. Both are available.
- Unlimited water, meaning the customer can have as many servings as they like.

Formulas

- The set of formulas is constructed from :
  - A set of propositional variables $\mathcal{V} = \{a, b, c, \ldots\}$
  - the constants : 0 1 $\bot$ $\top$
  - the connectives : $.^{\bot}$ $\oplus$ $\otimes$ $\&$ $\parr$ $\multimap$
  - the modalities : $!$ $?$

- Contextes : $\Gamma, \triangle, \ldots$ (multisets)

- Sequent : $\Gamma \vdash \triangle$

Axioms

$$\frac{\Gamma \vdash \Delta}{\Gamma, 1 \vdash \Delta} \qquad \frac{}{\vdash 1} \qquad \frac{}{\bot \vdash} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}$$

$$\frac{}{\Gamma, 0 \vdash \Delta} \qquad \frac{}{\Gamma \vdash \top, \Delta}$$

$$\frac{}{A \vdash A}$$

Structural rules

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

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

Connective rules

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta} \qquad \frac{\Gamma_1 \vdash A, \Delta_1 \qquad \Gamma_2 \vdash B, \Delta_2}{\Gamma_1, \Gamma_2 \vdash A \otimes B, \Delta_1, \Delta_2}$$

$$\frac{\Gamma_1, A \vdash \Delta_1 \qquad \Gamma_2, B \vdash \Delta_2}{\Gamma_1, \Gamma_2, A \parr B \vdash \Delta_1, \Delta_2} \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}$$

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

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

Connective rules

$$\frac{\Gamma_1 \vdash A, \Delta_1 \qquad \Gamma_2, B \vdash \Delta_2}{\Gamma_1, \Gamma_2, A \multimap B \vdash \Delta_1, \Delta_2}$$

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

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, A^{\perp} \vdash \Delta}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma \vdash A^{\perp}, \Delta}$$

$$\frac{\Gamma_1 \vdash A, \Delta_1 \qquad \Gamma_2, A \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}$$

De Morgan lows

We define an equivalence on formulas by :

$$A \equiv B \quad \text{iff} \quad \vdash_{\mathrm{LL}} A \multimap B \text{ and } \vdash_{\mathrm{LL}} B \multimap A.$$

Theorem

- $A^{\perp\perp} \equiv A$,
- $\bot^{\perp} \equiv 1, \quad 1^{\perp} \equiv \bot, \quad \top^{\perp} \equiv 0, \quad 0^{\perp} \equiv \top$,
- $(A \oplus B)^{\perp} \equiv A^{\perp} \,\&\, B^{\perp}, \quad (A \,\&\, B)^{\perp} \equiv A^{\perp} \oplus B^{\perp}$,
- $(A \otimes B)^{\perp} \equiv A^{\perp} \parr B^{\perp}, \quad (A \parr B)^{\perp} \equiv A^{\perp} \otimes B^{\perp}$,
- $(!A)^{\perp} \equiv ?A^{\perp}, \quad (?A)^{\perp} \equiv !A^{\perp}$,
- $A \multimap B \equiv A^{\perp} \parr B$.

New system

- Two sets of propositional variables $\mathcal{V} = \{a, b, c, \ldots\}$ and $\mathcal{V}^{\perp} = \{a^{\perp}, b^{\perp}, c^{\perp}, \ldots\}$.

- The set of formulas is constructed from the sets $\mathcal{V}$, $\mathcal{V}^{\perp}$, the constants 0, 1, $\perp$, $\top$, the connectives $\oplus$, $\otimes$, $\&$, $\parr$, and the modalities !, ?.

- The linear negation $.^{\perp}$ is defined for propositional variables and extends to all formulas through the following relations :
  - $a^{\perp\perp} = a, \quad \perp^{\perp} = 1, \quad 1^{\perp} = \perp, \top^{\perp} = 0, \quad 0^{\perp} = \top,$
  - $(A \oplus B)^{\perp} = A^{\perp} \,\&\, B^{\perp}, \quad (A \,\&\, B)^{\perp} = A^{\perp} \oplus B^{\perp},$
  - $(A \otimes B)^{\perp} = A^{\perp} \parr B^{\perp}, \quad (A \parr B)^{\perp} = A^{\perp} \otimes B^{\perp},$
  - $(!A)^{\perp} = ?A^{\perp}$ and $(?A)^{\perp} = !A^{\perp}$.

- We define $A \multimap B$ by $A^{\perp} \parr B$.

Derivation rules

$$\frac{}{\vdash 1}\ 1 \qquad \frac{\vdash \Gamma}{\vdash \bot, \Gamma}\ \bot \qquad \frac{}{\vdash \top, \Gamma}\ \top$$

$$\frac{}{\vdash A, A^{\bot}}\ \text{ax}$$

$$\frac{\vdash \Gamma}{\vdash ?A, \Gamma}\ \text{weak} \qquad \frac{\vdash ?A, ?A, \Gamma}{\vdash ?A, \Gamma}\ \textbf{con}\text{traction}$$

$$\frac{\vdash A, ?\Gamma}{\vdash !A, ?\Gamma}\ \textbf{pro}\text{motion} \qquad \frac{\vdash A, \Gamma}{\vdash ?A, \Gamma}\ \textbf{der}\text{eliction}$$

Derivation rules

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

$$\frac{\vdash A, \Gamma \qquad \vdash B, \Gamma}{\vdash A \,\&\, B, \Gamma} \,\& \qquad \frac{\vdash A, \Gamma}{\vdash A \oplus B, \Gamma} \oplus_1 \qquad \frac{\vdash B, \Gamma}{\vdash A \oplus B, \Gamma} \oplus_2$$

$$\frac{\vdash A, \Gamma \qquad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta} \text{ cut}$$

Examples

Theorem

- $A \,\&\, \top \equiv A, \qquad A \,\&\, A \equiv A, \qquad A \,\&\, B \equiv B \,\&\, A,$
  $A \,\&\, (B \,\&\, C) \equiv (A \,\&\, B) \,\&\, C.$
- $A \oplus 0 \equiv A, \qquad A \oplus A \equiv A, \qquad A \oplus B \equiv B \oplus A$
  $A \oplus (B \oplus C) \equiv (A \oplus B) \oplus C.$
- $A \otimes 1 \equiv A, \qquad A \otimes 0 \equiv 0, \qquad A \otimes B \equiv B \otimes A$
  $A \otimes (B \otimes C) \equiv (A \otimes B) \otimes C.$
- $A \parr \bot \equiv A, \qquad A \parr \top \equiv \top, \qquad A \parr B \equiv B \parr A$
  $A \parr (B \parr C) \equiv (A \parr B) \parr C.$
- $A \otimes (B \oplus C) \equiv (A \otimes B) \oplus (A \otimes C),$
  $A \parr (B \,\&\, C) \equiv (A \parr B) \,\&\, (A \parr C).$

Examples

Theorem

- $A^{\perp} \equiv A \multimap \perp$, $\quad 1 \multimap A \equiv A$,
  $A \multimap (B \multimap C) \equiv (A \otimes B) \multimap C$,
  $A \multimap (B \,\&\, C) \equiv (A \multimap B) \,\&\, (A \multimap C)$,
  $(A \oplus B) \multimap C \equiv (A \multimap C) \,\&\, (B \multimap C)$.

- $!!A \equiv !A$, $\quad !A \equiv (!A) \otimes (!A)$, $\quad !A \equiv 1 \,\&\, (!A)$,
  $!A \equiv A \,\&\, (!A)$.

- $??A \equiv ?A$, $\quad ?A \equiv (?A) \parr (?A)$, $\quad ?A \equiv \top \oplus (?A)$,
  $?A \equiv A \oplus (?A)$.

- $?0 \equiv \perp$, $\quad !\top \equiv 1$, $\quad !(A \,\&\, B) \equiv (!A) \otimes (!B)$,
  $?(A \oplus B) \equiv (?A) \parr (?B)$.

Example : $((A \otimes (B \oplus C)) \multimap ((A \otimes B) \oplus (A \otimes C))$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}}\,\mathrm{ax} \quad \dfrac{}{\vdash B, B^{\perp}}\,\mathrm{ax}}{\vdash A \otimes B, A^{\perp}, B^{\perp}}\,\otimes}{\vdash (A \otimes B) \oplus (A \otimes C), A^{\perp}, B^{\perp}}\,\oplus_1 \quad \dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}}\,\mathrm{ax} \quad \dfrac{}{\vdash C, C^{\perp}}\,\mathrm{ax}}{\vdash A \otimes C, A^{\perp}, C^{\perp}}\,\otimes}{\vdash (A \otimes B) \oplus (A \otimes C), A^{\perp}, C^{\perp}}\,\oplus_2}{\vdash (A \otimes B) \oplus (A \otimes C), A^{\perp}, B^{\perp} \,\&\, C^{\perp}}\,\&}{\vdash (A \otimes B) \oplus (A \otimes C), A^{\perp} \parr (B^{\perp} \,\&\, C^{\perp})}\,\parr}{\vdash_{\mathrm{LL}} ((A \otimes (B \oplus C)) \multimap ((A \otimes B) \oplus (A \otimes C))}\,\parr$$

Example : $(?A \parr ?B) \multimap ?(A \oplus B)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}}\,\text{ax}}{\vdash A \oplus B, A^{\perp}}\,\oplus_1}{\vdash ?(A \oplus B), A^{\perp}}\,\text{der}}{\vdash ?(A \oplus B), !A^{\perp}}\,\text{pro} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B, B^{\perp}}\,ax}{\vdash A \oplus B, B^{\perp}}\,\oplus_2}{\vdash ?(A \oplus B), B^{\perp}}\,\text{der}}{\vdash ?(A \oplus B), !B^{\perp}}\,\text{pro}}{\dfrac{\dfrac{\dfrac{\vdash ?(A \oplus B), ?(A \oplus B), !A^{\perp} \otimes !B^{\perp}}{\vdash ?(A \oplus B), !A^{\perp} \otimes !B^{\perp}}\,\text{con}}{\vdash_{\text{LL}} (?A \parr ?B) \multimap ?(A \oplus B)}\,\parr}\,\otimes}$$

Two translations

- $k : \mathrm{LL} \to \mathrm{LK}$

$$k(a) = a, \quad k(a^{\perp}) = \neg a,$$
$$k(\bot) = k(0) = \bot, \quad k(\top) = k(1) = \neg \bot,$$
$$k(A \otimes B) = k(A \,\&\, B) = k(A) \wedge k(B),$$
$$k(A \oplus B) = k(A \parr B) = k(A) \vee k(B)$$
$$k(!A) = k(?A) = k(A).$$

- $\ell : \mathrm{LK} \to \mathrm{LL}$

$$\ell(a) = a, \quad \ell(\bot) = \bot, \quad \ell(\neg A) = ?!\ell(A)^{\perp},$$
$$\ell(A \to B) = ?!\ell(A)^{\perp} \parr ?\ell(B) = !?\ell(A) \multimap ?\ell(B),$$
$$\ell(A \wedge B) = ?\ell(A) \& ?\ell(B), \quad \ell(A \vee B) = ?\ell(A) \parr ?\ell(B).$$

If $\Gamma = A_1, \ldots, A_n$, we write :

$$k(\Gamma) = k(A_1), \ldots, k(A_n), \quad \ell(\Gamma) = \ell(A_1), \ldots, \ell(A_n),$$
$$?\Gamma = ?A_1, \ldots, ?A_n, \quad !\Gamma = !A_1, \ldots, !A_n,$$
$$\Gamma^{\perp} = A_1^{\perp}, \ldots, A_n^{\perp}, \quad \neg\Gamma = \neg A_1, \ldots, \neg A_n.$$

The encoding of LK in LL

Lemma

1. *For any formula A of* LL, *we have* $\vdash_{\mathrm{LK}} k(A^{\perp}) \equiv \neg k(A)$.
2. *For any formula A of* LK, *we have* $\vdash_{\mathrm{LK}} k(\ell(A)) \equiv A$.

Theorem

1. *If* $\vdash_{\mathrm{LL}} \Gamma$, *then* $\vdash_{\mathrm{LK}} k(\Gamma)$.
2. $\Gamma \vdash_{\mathrm{LK}} \triangle$ *iff* $\vdash_{\mathrm{LL}} ?!\ell(\Gamma)^{\perp}, ?\ell(\triangle)$.

Properties of LL

Theorem

The system LL has the cut-elimination property.

Theorem

The system LL has the subformula property.

Theorem

The system LL is not decidable.

Theorem

The system MALL is decidable.

Outline

Questions

- How can the notion of truth be defined for formulas in linear logic ?
- How can the different connectives (the two $\wedge$ and the two $\vee$) be distinguished ?
- One idea is to use a simple algebraic structure and apply set-theoretic operations ($\cap$ and $\cup$) as well as algebraic operations ($\otimes$ and $\oplus$).
- We interpret each formula as a subset of a monoid, and the notion of truth is linked to the existence of the neutral element in this subset.
- The basic operation on subsets will be linear implication, as well as a particular subset to interpret $\bot$ and thus negation.

Phase space

1. A phase space is defined as a non-empty set M equipped with a commutative, associative multiplication $x.y$ and a neutral element 1 ($(M, ., 1)$ is a commutative monoid) and also of a subset $\perp\!\!\!\perp$ of M.

2. For any $X, Y \subseteq M$, we define the subsets
   - $X.Y = \{x.y \ / \ x \in X,\ y \in Y\}$,
   - $X \multimap Y = \{a \in M \ / \ \forall x \in X,\ a.x \in Y\}$,
   - $X^{\perp\!\!\!\perp} = X \multimap \perp\!\!\!\perp$.

3. We call a subset X of M a fact if it satisfies $X = X^{\perp\!\!\!\perp\perp\!\!\!\perp}$ i.e. $\exists Z \subseteq M, X = Z^{\perp\!\!\!\perp}$

Phase model

A phase model is a phase space $(M, ., 1, \perp\!\!\!\perp)$ equipped with :

- a map $b \mapsto b^*$ that associates to each variable b a fact b^*.
- of a submonoid K of the submonoid $I(M) = \{x \in \perp\!\!\!\perp^{\perp\!\!\!\perp} \ / \ x \in \{x^2\}^{\perp\!\!\!\perp\perp\!\!\!\perp}\}$.

We extend this function $.^*$ to all formulas as follows :

$$(a^\perp)^* = (a^*)^{\perp\!\!\!\perp}, \quad \perp^* = \perp\!\!\!\perp, \quad 1^* = \perp\!\!\!\perp^{\perp\!\!\!\perp} = \{1\}^{\perp\!\!\!\perp\perp\!\!\!\perp},$$
$$\top^* = M, \quad 0^* = (\top^*)^{\perp\!\!\!\perp} = \emptyset^{\perp\!\!\!\perp\perp\!\!\!\perp},$$
$$(A \parr B)^* = ((A^*)^{\perp\!\!\!\perp}.(B^*)^{\perp\!\!\!\perp})^{\perp\!\!\!\perp} = (A^*)^{\perp\!\!\!\perp} \multimap B^*,$$
$$(A \otimes B)^* = (A^*.B^*)^{\perp\!\!\!\perp\perp\!\!\!\perp},$$
$$(A \,\&\, B)^* = A^* \cap B^*, \quad (A \oplus B)^* = (A^* \cup B^*)^{\perp\!\!\!\perp\perp\!\!\!\perp}.$$
$$(?A)^* = ?A^* = ((A^*)^{\perp\!\!\!\perp} \cap K)^{\perp\!\!\!\perp}, \quad (!A)^* = !A^* = (A^* \cap K)^{\perp\!\!\!\perp\perp\!\!\!\perp}.$$

It is easy to verify that for every formula A, the set A^* is a fact.

Correctness of LL

Let $\mathcal{M} = (M, ., 1, \perp\!\!\!\perp, K, .^*)$ be phase model.

- If $\Gamma = A_1, \ldots, A_n$, we write $\Gamma^{*\perp\!\!\!\perp} = (A_1^*)^{\perp\!\!\!\perp} \ldots (A_n^*)^{\perp\!\!\!\perp}$.
- We say that $\mathcal{M}$ satisfies Γ si $\Gamma^{*\perp\!\!\!\perp} \subseteq \perp\!\!\!\perp$.
- In particular, $\mathcal{M}$ satisfies a formula A if $1 \in A^*$.

Theorem

If $\vdash_{\mathrm{LL}} \Gamma$*, then every phase model satisfies* Γ*.*

Corollary

If $\vdash_{\mathrm{LL}} A$*, then, for every phase model* $(M, ., 1, \perp\!\!\!\perp, K, .^*)$*,* $1 \in A^*$*.*

A particular phase model

We define a particular phase model denoted LL^* as follows :

- $M = \{\Gamma \ / \ \Gamma$ is a multiset of formulas $\}$,
- . is the concatenation of multisets,
- $1 = \emptyset$,
- $\perp\!\!\!\perp = \{\Gamma \ / \ \vdash_{\mathrm{LL}} \Gamma$ without the cut rule $\}$,
- For all $a \in \mathcal{V}$, $a^* = \{\Gamma \ / \ \vdash_{\mathrm{LL}} \Gamma, a$ without the cut rule$\}$,
- $K = \{?\Gamma \ / \ \Gamma$ is a multiset of formulas$\}$.

Completeness theorem

Lemma

In the model LL^**, for all formula A,* $A^* \subseteq \{A\}^{\perp\!\!\!\perp}$.

Theorem

We have the equivalence between the following properties :

1. $\vdash_{\mathrm{LL}} A$.
2. *Any phase model satisfies A.*
3. *The model* LL^* *satisfies A.*
4. $\vdash_{\mathrm{LL}} A$ *without the cut rule.*

Example : $\not\vdash_{\mathrm{LL}} ?(a \parr b) \multimap ?a \oplus ?b$

- Consider a cut-free derivation of $?(a \parr b) \multimap ?a \oplus ?b$.
- Then $\vdash_{\mathrm{LL}} !(a^\perp \otimes b^\perp), ?a \oplus ?b$, thus $\vdash_{\mathrm{LL}} !(a^\perp \otimes b^\perp), ?a$ or $\vdash_{\mathrm{LL}} !(a^\perp \otimes b^\perp), ?b$.
- Suppose that $\vdash_{\mathrm{LL}} !(a^\perp \otimes b^\perp), ?a$, then $\vdash_{\mathrm{LL}} ?(a \parr b) \multimap ?a$.
- Consider the phase model defined by $M = \{x^n \ / \ n \in \mathbb{Z}\}$, $\perp\!\!\!\perp = \{x^n \ / \ n \in \mathbb{N}\}$ and $K = \{1\}$.
- We take $a^* = \{x^n \ / \ n \geq 1\}$ et $b^* = \{x^n \ / \ n \geq -2\}$.
- It is easy to verify that $(?(a \parr b))^* = M$ and $(?a)^* = \perp\!\!\!\perp$.

Hence a contradiction.

Finite models property

Theorem

The system LL *does not have the property finite models property .*

Consider the formula F

$$\big(!\alpha \otimes !(a \otimes b) \otimes !(a \otimes b \multimap 1)\big) \multimap b$$

$$=$$

$$?a^{\perp} \parr ?(?a^{\perp} \parr ?b^{\perp}) \parr ?(\alpha \otimes b \otimes \perp) \parr b$$

We verify that :

- $\nvdash_{\mathrm{LL}} F$.
- Every finite phase model satisfies F.

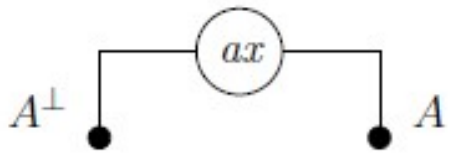
Outline

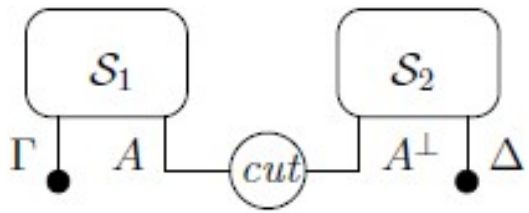
Proof nets

- Proof Nets are graphical representations of proofs in linear logic.
- They simplify the analysis of proofs and respect the unique use of resources.
- They provide a clear and intuitive view of the connections within the proofs.
- They also allow for the algorithmic verification of the validity of proofs.

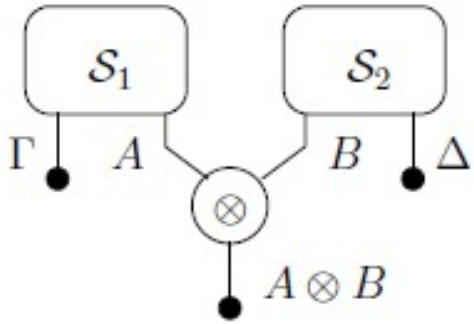
Proof nets for MLL

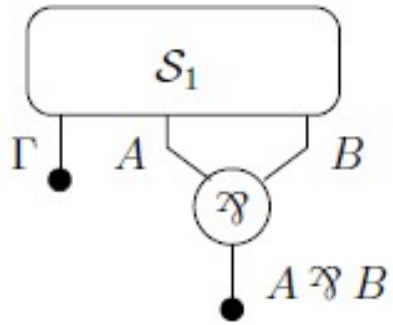
Here is the graphical presentation of the rules of MLL :





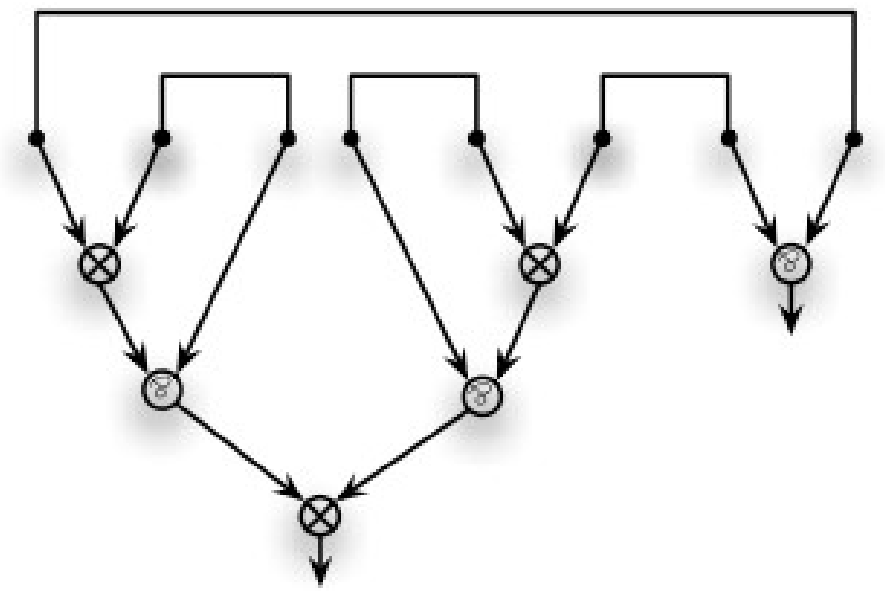
Proof nets for MLL





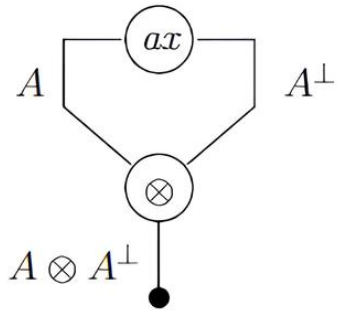
Example

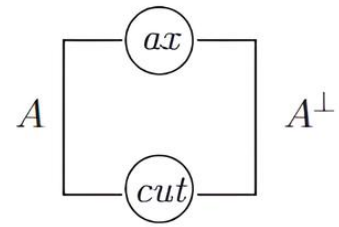
Here are an example of a proof net of MLL :

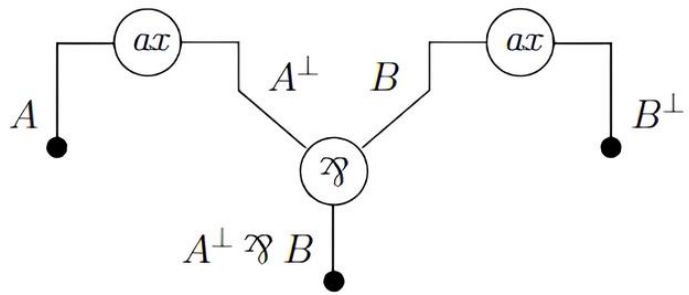


Problem

Here are a few examples of proof structures which do not correspond to any proof of MLL :







Correctness criterion

- A correctness criterion is a criterion used to characterize proof nets corresponding to proofs in MLL.

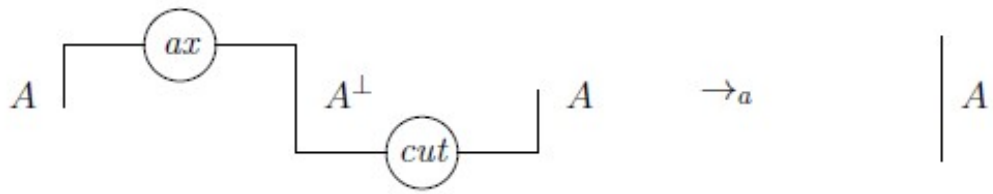
- A switch on a proof net is a choice of orientation for each $\parr$-link, meaning that each $\parr$-link is connected to only one of its two premises.

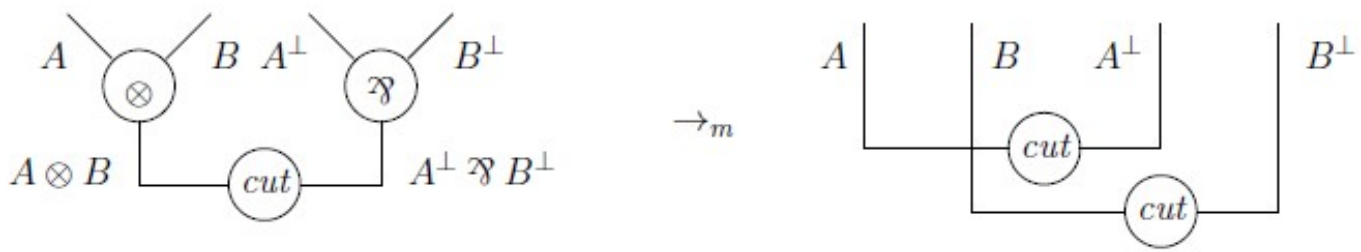
Theorem (Danos-Regnier)

A proof net is correct if and only if it is connected and acyclic for every choice of switch, that is, if and only if for every choice of switch, the resulting graph is a tree.

Cut Elimination

We consider two reductions steps :





Properties

Theorem (Confuence)

The reduction of multiplicative proof nets is confuent.

Theorem (Strong Normalization)

The reduction of multiplicative proof nets is strongly normalizing.

Outline

Introduction

- How can the algorithmic content of proofs in linear logic be extracted ?
- Should we restrict ourselves to a logical subsystem ?
- Should we keep the presentation with sequents (everything on the right, right and left, multiple formulas on the right) ?
- What additional results can we obtain from doing this work ?
- We provide an example of work on this topic without going into details (system L).

Terms and reduction rules

Syntax

$$
\begin{array}{lcl}
t,\, u & ::= & x \mid \mu x.\, c \mid (t, u) \mid \mu\,(x, u)\,.\, c \mid () \mid \mu().\, c \\
 & & 1.t \mid 2.t \mid \{\mu\,(1.x)\,.\, c_1\,,\ \mu\,(2.y)\,.\, c_2\} \mid \{\} \\
c & ::= & \langle\, t \,|\, u \,\rangle
\end{array}
$$

Reduction

$$
\begin{array}{lcl}
\langle\, t \,|\, \mu x.\, c \,\rangle & \rightsquigarrow & c[\, x \backslash t \,] \\
\langle\, (t, u) \,|\, \mu\,(x, y)\,.\, c \,\rangle & \rightsquigarrow & c[\, x \backslash t, y \backslash u \,] \\
\langle\, () \,|\, \mu().\, c \,\rangle & \rightsquigarrow & c \\
\langle\, 1.t \,|\, \{\mu\,(1.x)\,.\, c_1\,,\ \mu\,(2.y)\,.\, c_2\} \,\rangle & \rightsquigarrow & c_1[\, x \backslash t \,] \\
\langle\, 2.t \,|\, \{\mu\,(1.x)\,.\, c_1\,,\ \mu\,(2.y)\,.\, c_2\} \,\rangle & \rightsquigarrow & c_2[\, y \backslash t \,]
\end{array}
$$

Typing system

TYPING

$$\frac{}{x{:}A \vdash x : A}\ \mathrm{id} \qquad \frac{\Gamma \vdash t : A \quad \Delta \vdash u : A^{\perp}}{\Gamma, \Delta \vdash \langle t \,|\, u \rangle}\ \mathrm{cut}$$

$$\frac{\Gamma, x{:}A \vdash c}{\Gamma \vdash \mu x.\, c : A^{\perp}}\ \mu$$

$$\frac{\Gamma \vdash t : A \quad \Delta \vdash u : B}{\Gamma, \Delta \vdash (t, u) : A \otimes B}\ \otimes \qquad \frac{\Gamma, x{:}A, y{:}B \vdash c}{\Gamma \vdash \mu\,(x, y)\,.\, c : A^{\perp} \parr B^{\perp}}\ \parr$$

$$\frac{}{\vdash () : 1}\ 1 \qquad \frac{\Gamma \vdash c}{\Gamma \vdash \mu().\, c : \perp}\ \perp$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash 1.t : A \oplus B}\ \oplus\mathrm{l} \qquad \frac{\Gamma, x{:}A \vdash c_1 \quad \Gamma, y{:}B \vdash c_2}{\Gamma \vdash \{\mu\,(1.x)\,.\, c_1\,,\, \mu\,(2.y)\,.\, c_2\} : A^{\perp} \& B^{\perp}}\ \&$$

$$\frac{\Gamma \vdash u : B}{\Gamma \vdash 2.u : A \oplus B}\ \oplus\mathrm{r}$$

No rule for 0

$$\frac{}{\Gamma \vdash \{\} : \top}\ \top$$

Derived syntatx and typing rules

Derived syntax

$$
\begin{array}{lcl}
\lambda x.\, t & = & \mu\,(x, \alpha)\,.\,\langle\, t \,|\, \alpha \,\rangle \\
t\ u & = & \mu\alpha.\,\langle\, t \,|\, (u, \alpha) \,\rangle \\
\{1 = t\,,\, 2 = u\} & = & \{\mu\,(1.\alpha)\,.\,\langle\, t \,|\, \alpha \,\rangle\ ,\ \mu\,(2.\alpha)\,.\,\langle\, u \,|\, \alpha \,\rangle\} \\
t.1 & = & \mu\alpha.\,\langle\, t \,|\, 1.\alpha \,\rangle \\
t.2 & = & \mu\alpha.\,\langle\, t \,|\, 2.\alpha \,\rangle
\end{array}
$$

Derived typing rules

$$\frac{\Gamma, x{:}A \vdash t : B}{\Gamma \vdash \lambda x.\, t : A \multimap B}\ \lambda$$

$$\frac{\Gamma \vdash t : A \multimap B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}\ \text{app}$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash u : B}{\Gamma \vdash \{1 = t\,,\, 2 = u\} : A \& B}\ \text{record}$$

$$\frac{\Gamma \vdash t : A \& B}{\Gamma \vdash t.1 : A}\ \pi_1$$

$$\frac{\Gamma \vdash t : A \& B}{\Gamma \vdash t.2 : B}\ \pi_2$$

Conclusion

- Linear logic is a rich enough system to express properties that are not necessarily mathematical.
- It has very good properties and allows for encoding both intuitionistic and classical logic.
- It is also possible to extract the algorithmic content from proofs and thus produce a programming language.
- Several works have been carried out to improve the program extraction process using the concept of polarity.

General conclusion

- The presentation of proof systems is important both for writing comprehensible mathematical proofs and for extracting programs that behave effectively.
- It is interesting to seek a programming theorem for the calculus encoding sequent calculus, which begins with a characterization of integers.
- The study of linear logic to achieve this objective is very interesting. One must find a balance between a system that is practical for making mathematical proofs and programs derived from these proofs that behave well.