

# Calcul des séquents

Diane Gallois-Wong

2014

# Introduction

Théorie de la preuve (fin XIX<sup>e</sup>s.) : formalisation mathématique d'un exposé d'arguments visant à convaincre un interlocuteur de la validité d'une assertion.

Calcul des séquents (Gentzen, 1936) :

Intérêt informatique : algorithme de recherche de preuves

Exemple d'approche mathématique : transfert vers la théorie des catégories

# Les formules

Logique intuitionniste propositionnelle : formules construites à partir

- de variables propositionnelles ;
- de la constante  $\perp$  (*faux*) ;
- des connecteurs binaires  $\wedge$  (*et*),  $\vee$  (*ou*) et  $\rightarrow$  (*implique*).

$\neg A$  est une simple notation signifiant  $A \rightarrow \perp$ .

Logique classique propositionnelle : même construction, en ajoutant la constante  $\top$  (*vrai*), et le connecteur unaire  $\neg$  (*non*).

- ① Mémoire : Calcul des séquents et catégorie
  - Définitions sur les calculs des séquents
  - Construction d'une catégorie à partir d'un calcul de séquents
  
- ② Stage : Recherche de preuves certifiée en logique intuitionniste
  - Prouveur de logique intuitionniste basé sur le calcul **LSJ**
  - Perspective de certification : comparaison d'implémentations

- 1 Mémoire : Calcul des séquents et catégorie
  - Définitions sur les calculs des séquents
  - Construction d'une catégorie à partir d'un calcul de séquents

Définitions sur les calculs des séquents à travers l'exemple du calcul **LK** puis celui du calcul **LJ**. Ces deux calculs ont été définis par Gentzen. **LK** correspond à la logique classique et **LJ** à la logique intuitionniste.

# Les séquents

Chaque calcul des séquents a sa propre définition d'un *séquent*.

## Définition

Un **séquent** de **LK** consiste en deux listes de formules  $\Gamma$  et  $\Delta$ .

On le note  $\Gamma \vdash \Delta$ .

Formules de  $\Gamma$  : “hypothèses”

Formules de  $\Delta$  : “conclusions”

Un séquent  $\Gamma \vdash \Delta$  correspond à la formule  $(\bigwedge_{G \in \Gamma} G) \rightarrow (\bigvee_{D \in \Delta} D)$  en logique classique.

# Les règles

Règles de la forme :  $\frac{\text{prémisses}}{\text{conclusion}}$  (*nom de la règle*)

Signification :

Si les prémisses sont valides, alors la conclusion est aussi valide.

Exemples de règles de **LK** :

$$\frac{}{A \vdash A} (id) \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta} (\wedge R)$$

# Familles de règles

Identité :  $\frac{}{A \vdash A} (id)$

Coupure :  $\frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} (cut)$

## Règles logiques

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta} (\wedge R)$$

$$\frac{}{\Gamma, \perp \vdash \Delta} (\perp L)$$

$$\frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \rightarrow B, \Delta} (\rightarrow R)$$

## Règles structurelles

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta} (weakening L)$$

$$\frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta} (contraction R)$$

$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}{\Gamma_1, B, A, \Gamma_2 \vdash \Delta} (exchange L)$$



## Preuve d'un séquent

Une **preuve**, ou **arbre de preuve**, est un arbre tel que

- chaque nœud est étiqueté par un séquent et une règle ;
- les séquents associés à un nœud donné et à ses fils forment une application de la règle associée à ce nœud.

Un **preuve d'un séquent** donné est une preuve telle que la racine est étiquetée par ce séquent.

$$\frac{\frac{\overline{A \vdash A}}{A, B \vdash A} (id) \quad \frac{\frac{\overline{B \vdash B}}{B, A \vdash A} (id) \quad \frac{\overline{B, A \vdash A}}{A, B \vdash A} (exchange L)}{A, B \vdash A \wedge B} (\wedge R)$$

*Preuve du séquent  $A, B \vdash A \wedge B$*

# Prouvabilité d'un séquent dans un calcul

## Définition

*Un séquent est **prouvable dans un calcul** s'il existe une preuve de ce séquent.*

## Proposition

*Une formule  $A$  est valide en logique classique si, et seulement si, le séquent  $\vdash A$  est prouvable dans **LK**.*

## Calcul **LJ** et logique intuitionniste

Calcul **LJ** obtenu à partir de **LK** en se restreignant à des séquents avec exactement une formule à droite.

**Séquent** de **LJ** : une liste de formules  $\Gamma$  et une formule  $D$  ; noté  $\Gamma \vdash D$ .  
Correspond à la formule  $(\bigwedge_{G \in \Gamma} G) \rightarrow D$  en logique intuitionniste.

### Définition / Proposition

*Une formule  $A$  est **prouvable en logique intuitionniste** si, et seulement si, le séquent  $\vdash A$  est prouvable dans **LJ**.*

Pour prouver le séquent  $\vdash A \vee \neg A$  dans **LK**, on passe par un séquent  $\vdash A, \neg A$  qui n'existe pas dans **LJ**.

## Les règles de LJ

Règles de **LJ** : adaptation de celles de **LK** à des séquents avec exactement une formule à droite.

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta} (\wedge R) \quad \dashrightarrow \quad \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} (\wedge R)$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} (cut) \quad \dashrightarrow \quad \frac{\Gamma \vdash A \quad \Gamma', A \vdash D}{\Gamma, \Gamma' \vdash D} (cut)$$

Disparition des règles structurelles à droite :  $\frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta} (weakening R)$

$$\frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta} (contraction R) \quad \frac{\Gamma \vdash \Delta_1, A, B, \Delta_2}{\Gamma \vdash \Delta_1, B, A, \Delta_2} (exchange R)$$

## 1 Mémoire : Calcul des séquents et catégorie

- Définitions sur les calculs des séquents
- Construction d'une catégorie à partir d'un calcul de séquents

Construction d'une catégorie à partir d'un calcul de séquents correspondant à la logique linéaire intuitionniste, en s'appuyant sur une preuve constructive du théorème d'élimination de la coupure détaillée par P.-A. Melliès.

Ce calcul sera appelé **LLI**.

Les séquents de **LLI** sont les mêmes que ceux de **LJ**.

## Le théorème d'élimination de la coupure

$$\begin{array}{l} \text{Règle de coupure : } \frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} (cut) \quad \text{dans } \mathbf{LK} \\ \frac{\Gamma \vdash A \quad \Gamma', A \vdash D}{\Gamma, \Gamma' \vdash D} (cut) \quad \text{dans } \mathbf{LJ} \text{ ou } \mathbf{LLI} \end{array}$$

Signification importante :

Si on prouve  $A$ , on peut ensuite se servir de  $A$  comme hypothèse.

### Théorème d'élimination de la coupure

*Si on enlève la règle de coupure du calcul **LLI**, on obtient un calcul des séquents équivalent, c'est-à-dire que les séquents prouvables restent les mêmes.*

Ce théorème est également vérifié pour **LK** et **LJ**.

Théorème très important en recherche automatisée de preuves.

## Procédé d'élimination de la coupure pour **LLI**

Reformulation du théorème d'élimination de la coupure :

*Pour toute preuve d'un séquent, il existe une preuve du même séquent dans laquelle la règle de coupure n'apparaît pas.*

Preuve constructive (pour **LLI**) : définition précise d'un **procédé d'élimination de la coupure**.

Relation binaire sur les preuves :  $p \triangleright p'$  si ce procédé permet de transformer  $p$  en  $p'$ .

Clôture symétrique et transitive de  $\triangleright$  : relation d'équivalence appelée **équivalence selon le procédé d'élimination de la coupure**.

## Définition

Une **catégorie** consiste en des **objets** (notés  $A, B, \dots$ ) et des **morphismes** (notés  $f, g, \dots$ ), avec une loi binaire partielle  $\circ$  sur les morphismes, tels que

- à chaque morphisme  $f$  est associé un couple d'objets  $(A, B)$  ; on note  $f : A \rightarrow B$ , et on dit que  $A \rightarrow B$  est le type de  $f$ .
- si  $f : A \rightarrow B$  et  $g : B \rightarrow C$ , il existe un morphisme  $g \circ f : A \rightarrow C$ .
- la loi  $\circ$  est **associative** : si  $f : A \rightarrow B$  et  $g : B \rightarrow C$  et  $h : C \rightarrow D$ , alors  $h \circ (g \circ f) = (h \circ g) \circ f$ .
- pour tout  $A$ , il existe une **identité**  $\text{id}_A : A \rightarrow A$  vérifiant  $\text{id}_A \circ f = f$  si  $f : B \rightarrow A$  et  $g \circ \text{id}_A = g$  si  $g : A \rightarrow B$ .

Exemple :            objets : ensembles,    morphismes : fonctions



À chaque preuve  $p$  on associe une *dénotation*  $[p]$ .

Invariance selon le procédé d'élimination de la coupure : si  $p$  et  $p'$  sont équivalentes selon ce procédé alors  $[p] = [p']$ .

Composition : si  $p_1$  preuve de  $A \vdash B$  et  $p_2$  preuve de  $B \vdash C$ , alors on pose  $[p_2] \circ [p_1] = [p]$  où  $p$  est la preuve de  $A \vdash C$  suivante :

$$p : \frac{\frac{\dots p_1 \dots}{A \vdash B} \quad \frac{\dots p_2 \dots}{B \vdash C}}{A \vdash C} (cut)$$

On associe aussi une dénotation  $[A]$  à chaque formule  $A$ .

- Catégorie  $\mathcal{CP}$  :
- objets : dénnotations des formules
  - morphismes : dénnotations des preuves de séquents de la forme  $A \vdash B$ , de type  $[A] \rightarrow [B]$
  - identité sur  $[A]$  : dénotation de  $\frac{}{A \vdash A} (id)$

- Catégorie  $\mathcal{CP}$  :
- objets : dénnotations des formules
  - morphismes : dénnotations des preuves de séquents de la forme  $A \vdash B$ , de type  $[A] \rightarrow [B]$
  - identité sur  $[A]$  : dénnotation de  $\frac{}{A \vdash A} (id)$

Associativité :

$$\begin{array}{c}
 \frac{\frac{\frac{}{A \vdash B} (p_1)}{A \vdash B} \quad \frac{\frac{}{B \vdash C} (p_2)}{B \vdash C}}{A \vdash C} (cut) \quad \frac{\frac{}{C \vdash D} (p_3)}{C \vdash D}}{A \vdash D} (cut) \\
 \frac{}{A \vdash D} ([p_3] \circ ([p_2] \circ [p_1])) \\
 \end{array}
 \quad
 \begin{array}{c}
 \frac{\frac{\frac{}{A \vdash B} (p_1)}{A \vdash B} \quad \frac{\frac{\frac{}{B \vdash C} (p_2)}{B \vdash C} \quad \frac{\frac{}{C \vdash D} (p_3)}{C \vdash D}}{B \vdash D}}{A \vdash D} (cut) \\
 \frac{}{A \vdash D} ([p_3] \circ [p_2]) \circ [p_1]
 \end{array}$$

$$[p_3] \circ ([p_2] \circ [p_1]) = ([p_3] \circ [p_2]) \circ [p_1]$$

Propriétés de l'identité sur  $A$  :

$$\frac{\frac{}{A \vdash A} (id) \quad \frac{\frac{}{A \vdash B} (p)}{A \vdash B}}{A \vdash B} (cut) \quad \frac{}{A \vdash B} (p)$$

$$[p] \circ id_A = [p]$$

De même pour  $id_A \circ [p'] = [p']$   
si  $p'$  est une preuve de  $B \vdash A$ .

# Perspectives

- À partir de certains connecteurs binaires de la logique linéaire intuitionniste, on peut munir cette catégorie  $\mathcal{CP}$  d'opérateurs afin qu'elle corresponde à des définitions importantes en théorie des catégories (catégorie monoïdale symétrique, catégorie monoïdale fermée, catégorie cartésienne...).
- Analogie importante avec la théorie des nœuds.
- Construction similaire d'une catégorie associée à d'autres calculs de séquents. Notamment, on peut le faire pour **LK**, mais on obtient une catégorie dégénérée.

- 1 Mémoire : Calcul des séquents et catégorie
  - Définitions sur les calculs des séquents
  - Construction d'une catégorie à partir d'un calcul de séquents
- 2 Stage : Recherche de preuves certifiée en logique intuitionniste
  - Prouveur de logique intuitionniste basé sur le calcul **LSJ**
  - Perspective de certification : comparaison d'implémentations

Plusieurs prouveurs existants basés sur des calculs des séquents, mais emploi de structures de données complexes

Calcul **LSJ** (M. Ferrari, C. Fiorentini et G. Fiorino, 2012) : propriétés intéressantes permettant d'utiliser des structures de données relativement simples, ce qui est favorable à une certification.

## 2 Stage : Recherche de preuves certifiée en logique intuitionniste

- Prouveur de logique intuitionniste basé sur le calcul **LSJ**
- Perspective de certification : comparaison d'implémentations

Implémentation d'un prouveur de logique intuitionniste basé sur une légère variante du calcul **LSJ**, à partir du pseudo-code donné dans l'article de M. Ferrari, C. Fiorentini et G. Fiorino.

Comparaison avec des prouveurs existants.

Langage utilisé : OCaml.

## Les séquents de **LSJ**

*Multiensembles* : collections où le nombre d'éléments est pris en compte, mais non l'ordre des éléments.

### Définition

Un **séquent de LSJ** consiste en trois multiensembles de formules  $\Theta$ ,  $\Gamma$  et  $\Delta$ . On le note  $\Theta ; \Gamma \vdash \Delta$ .

Un séquent  $\emptyset ; \Gamma \vdash \Delta$  correspond à la formule  $(\bigwedge_{G \in \Gamma} G) \rightarrow (\bigvee_{D \in \Delta} D)$  en logique intuitionniste.

### Proposition

Une formule  $A$  est prouvable en logique intuitionniste si, et seulement si, le séquent  $\emptyset ; \emptyset \vdash A$  est prouvable dans **LSJ**.

Un séquent  $\Theta ; \Gamma \vdash \Delta$  ne correspond pas toujours à une formule.  
Interprétation d'un tel séquent en termes de modèles de Kripke.

## Les règles de **LSJ**

Huit règles :  $id$ ,  $\perp L$ ,  $\wedge L$ ,  $\wedge R$ ,  $\vee L$ ,  $\vee R$ ,  $\rightarrow L$ ,  $\rightarrow R$ .

Seules les règles  $\rightarrow L$  et  $\rightarrow R$  agissent sur  $\Theta$ .

$$\frac{}{\Theta; A, \Gamma \vdash A, \Delta} (id) \quad \frac{}{\Theta; \perp, \Gamma \vdash \Delta} (\perp L) \quad \frac{\Theta; \Gamma \vdash A, B, \Delta}{\Theta; \Gamma \vdash A \vee B, \Delta} (\vee R)$$

Pas de règle de coupure, pas de règles structurelles.

## Le calcul **LSJℓ**, légère variante de **LSJ**

Séquent de **LSJℓ** : deux multiensembles  $\Gamma$  et  $\Delta$  de couples “*indice : formule*”, et un indice  $n$ , les indices étant des entiers naturels. On note  $\Gamma \vdash_n \Delta$ .

$\Gamma' \vdash_n \Delta'$  représente  $\Theta ; \Gamma \vdash \Delta$  où  $\left\{ \begin{array}{l} \Theta = \Gamma'_{n+1} \\ \Gamma = \Gamma'_{\leq n} \\ \Delta = \Delta'_n \end{array} \right.$

$$\frac{\Theta ; A, \Gamma \vdash B, \Delta \quad \emptyset ; A, \Theta, \Gamma \vdash B}{\Theta ; \Gamma \vdash A \rightarrow B, \Delta} (\rightarrow R)$$

$$\frac{0 : A, \Gamma' \vdash_n n : B, \Delta' \quad 0 : A, \Gamma' \vdash_{n+1} n+1 : B, \Delta'}{\Gamma' \vdash_n n : A \rightarrow B, \Delta'} (\rightarrow R)$$

$A$  est prouvable en logique intuitionniste si et seulement si  $\emptyset ; \emptyset \vdash A$  est prouvable dans **LSJ**, c'est-à-dire  $\emptyset \vdash_0 0 : A$  est prouvable dans **LSJℓ**.



## Idée de l'algorithme de recherche de preuve

Récursivement, pour déterminer si un séquent  $\sigma$  est prouvable :  
 pour toute application de règle possible de la forme  $\frac{\sigma_1 \quad \dots \quad \sigma_p}{\sigma}$ , on détermine si chaque  $\sigma_i$  est prouvable.

S'il y en a une telle que toutes les prémisses sont prouvables (notamment si la règle n'a pas de prémisses),  $\sigma$  est prouvable (on a un arbre de preuve de  $\sigma$ ). Sinon,  $\sigma$  n'est pas prouvable.

Terminaison assurée par des propriétés de **LSJ**.

Une prémisses  $\sigma_i$  d'une règle  $\frac{\sigma_1 \quad \dots \quad \sigma_p}{\sigma}(\mathcal{R})$  est **inversible** si on a : si la conclusion  $\sigma$  est prouvable, alors  $\sigma_i$  est aussi prouvable.

Une règle est **inversible** si toutes ses prémisses sont inversibles.

# Priorités

Une prémisses  $\sigma_i$  d'une règle  $\frac{\sigma_1 \quad \dots \quad \sigma_p}{\sigma}(\mathcal{R})$  est **inversible** si on a : si la conclusion  $\sigma$  est prouvable, alors  $\sigma_i$  est aussi prouvable.

Une règle est **inversible** si toutes ses prémisses sont inversibles.

De la règle la plus prioritaire à la moins prioritaire :

- les axiomes  $id$  et  $\perp L$  : pas de prémisses.
- $\wedge L$  et  $\vee R$  : règles inversibles à une seule prémisses.
- $\wedge R$  et  $\vee L$  : règles inversibles à deux prémisses.
- $\rightarrow L$  et  $\rightarrow R$  : règles non inversibles.  $\rightarrow L$  a trois prémisses dont deux sont inversibles ;  $\rightarrow R$  a deux prémisses dont une est inversible.

## Localité des règles

Une règle  $\frac{\sigma_1 \quad \dots \quad \sigma_p}{\sigma} (\mathcal{R})$  est **locale** si on peut retrouver  $\sigma$  à partir de n'importe quelle prémisses  $\sigma_i$  et d'une information de taille "raisonnable".

Si toutes les règles sont locales, on peut effectuer l'algorithme de recherche de preuve en conservant un seul séquent en mémoire à tout moment.

Intérêt de **LSJℓ** par rapport à **LSJ** : toutes les règles sont locales.

$$\frac{\Theta; A, \Gamma \vdash B, \Delta \quad \emptyset; A, \Theta, \Gamma \vdash B}{\Theta; \Gamma \vdash A \rightarrow B, \Delta} (\rightarrow R)$$

$$\frac{0:A, \Gamma' \vdash_n n:B, \Delta' \quad 0:A, \Gamma' \vdash_{n+1} n+1:B, \Delta'}{\Gamma' \vdash_n n:A \rightarrow B, \Delta'} (\rightarrow R)$$

## Propriété de la sous-formule et indexation

Récursivement,  $A$  est une **sous-formule** de  $B$  si  $A = B$  ou si  $B = B_1 c B_2$  avec  $c$  un connecteur et  $A$  est une sous-formule de  $B_1$  ou de  $B_2$ .

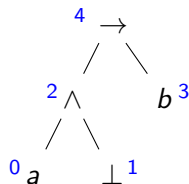
**LSJ** donc aussi **LSJ** $\ell$  vérifient la propriété de la sous-formule : toute formule apparaissant dans une preuve d'un séquent  $\sigma$  est une sous-formule d'une formule de  $\sigma$ .

Conséquence : toute formule rencontrée dans une recherche de preuve sur une formule  $A$  (i.e. sur le séquent  $\emptyset \vdash_0 0:A$ ) est une sous-formule de  $A$ .

# Propriété de la sous-formule et indexation

Conséquence : toute formule rencontrée dans une recherche de preuve sur une formule  $A$  (i.e. sur le séquent  $\emptyset \vdash_0 0:A$ ) est une sous-formule de  $A$ .

Représentation des formules par des entiers grâce à une phase préliminaire d'indexation.



Numéro	0	1	2	3	4
Description	$a$	$\perp$	$0 \wedge 1$	$b$	$2 \rightarrow 3$

# Efficacité de LSJ

Prouveur testé sur les formules de la bibliothèque ILTP<sup>1</sup>.

Quatre prouveurs de logique intuitionniste propositionnelle sont répertoriés sur ILTP.

Difficulté d'une formule :  $0.25 * \text{nombre de prouveurs ne terminant pas en moins de cinq minutes}$

(tableau : comparaison de quelques temps<sup>2</sup> avec les difficultés données par ILTP. Notamment, SYJ209 : difficulté 1, notre prouveur termine en moins d'une milliseconde : grâce à l'inversibilité d'une prémisse de  $\rightarrow R$ , linéaire en la taille de la formule initiale au lieu de factoriel)

---

1. <http://www.cs.uni-potsdam.de/ti/iltp/>

2. Tests effectués sous MacOS avec un processeur 2.7 GHz Intel Core i7.

- 2 Stage : Recherche de preuves certifiée en logique intuitionniste
  - Prouveur de logique intuitionniste basé sur le calcul **LSJ**
  - Perspective de certification : comparaison d'implémentations

Objectif à long terme : certification en Coq d'un prouveur de logique intuitionniste propositionnelle.

Certification elle-même non abordée, mais modifications d'implémentation visant à faciliter une telle certification. Comparaison des différents prouveurs ainsi implémentés.

## Le langage $T$

$T$  : langage simple pour faciliter la certification.

Petit langage fonctionnel ; seul type de données : des arbres binaires construits à partir d'un arbre vide et de feuilles étiquetées par des entiers naturels.

Réalisation par D. Larchey-Wendling d'un compilateur certifié en Coq de  $T$  vers un langage exécutable par une machine abstraite assez simple, et d'un programme certifié simulant l'exécution de cette machine abstraite.

$M$  : langage associé à cette machine abstraite.

Mettre cette image ? (avec une meilleure qualité) Décoratif, mais je n'ai pas le temps d'en parler et certains pourraient trouver cela frustrant...

```
let mem arg =  
  match arg with <x,l> =>  
    if isnull(l) then 0 else  
      match l with <hd,tl> =>  
        if call equal <hd,x> then 1 else call mem <x,tl>
```



# Compilation de fonctions adaptées à la formule

$$\frac{i:A, \Gamma \vdash_n \Delta \quad i:B, \Gamma \vdash_n \Delta}{i:A \vee B, \Gamma \vdash_n \Delta} (\vee L)$$

formule principale :  $A \vee B$

Juste après la phase d'indexation : compilation, pour chaque numéro de formule, de fonctions de transformation du séquent dans le cas où cette formule est la formule principale. Entre 0 et 6 fonctions pour chaque numéro de formule.

Par exemple, compilation en  $\mathbf{T}$  de ces fonctions pour chaque numéro de formule, puis ajout de code en  $\mathbf{T}$  fixe : on obtient un programme en  $\mathbf{T}$  adapté à la formule donnée en entrée.

## Différents prouveurs implémentés

- **“Prouveur simple”** : le plus simple et le plus efficace, intégralement en OCaml. Structures de données élaborées pour représenter les multiensembles du séquent.
- **“Prouveur simple avec listes”** : comme le précédent, mais structures élaborées remplacées par simples listes.
- **“Prouveur  $T$ ”** : compilation d'un programme en  $T$  adapté à la formule en entrée, exécution à l'aide d'un interpréteur.
- **“Prouveur  $T$   $M$ ”** : compilation du même programme en  $T$ , compilation de  $T$  vers  $M$ , simulation de l'exécution par la machine abstraite à laquelle  $M$  correspond.
- **“Prouveur compilé Caml”** : compilation d'un programme en OCaml adapté à la formule en entrée, compilation par `ocamlc`, lancement de l'exécutable.

# Répartition du code ?

(Indiquer nombre de lignes de code)

Utilitaires (gestion des tests)

Analyseur ILTP

Indexation

Prouveur simple (sauf l'indexation)

Structures de données pour le séquent

Autres prouveurs (sauf l'indexation)

Compilation de fonctions selon la formule

Code trees fixe

Total

# Comparaison des différents prouveurs implémentés

	"simple"	"simple listes"	"compilé Caml"	" $T$ "	" $T M$ "
total	0.05 s	0.08 s	.61 s	14 s	57 s
1 <sup>ère</sup> compil.	—	—	0.01 s	0.02 s	0.02 s
2 <sup>ème</sup> compil.	—	—	0.28 s	—	0.02 s
exécution	—	—	0.32 s	14 s	57 s

SYJ201+1.002

taille 99

environ 70 000 appels

	"simple"	"simple listes"	"compilé Caml"	" $T$ "	" $T M$ "
total	0.30 s	1.4 s	5.0 s	250 s	> 300 s
1 <sup>ère</sup> compil.	—	—	0.03 s	0.05 s	0.06 s
2 <sup>ème</sup> compil.	—	—	0.8 s	—	0.12 s
exécution	—	—	4.1 s	250 s	> 300 s

SYJ207+1.004

taille 213

environ 500 000 appels

# Bilan

- Confirmation de l'intérêt du calcul **LSJ** en recherche de preuves en logique intuitionniste propositionnelle, en termes d'efficacité comme de perspective de certification.
- Impact fort des méthodes employées pour faciliter la certification sur l'efficacité, mais améliorations à envisager : par exemple, ajout d'effets de bords (même limités à une seule variable globale) au langage **T**.

# Références I

Roy Dyckhoff.

Contraction-free sequent calculi for intuitionistic logic.

*J. Symb. Log.*, 57(3) :795–807, 1992.

Mauro Ferrari, Camillo Fiorentini, and Guido Fiorino.

Contraction-Free Linear Depth Sequent Calculi for Intuitionistic Propositional Logic with the Subformula Property and Minimal Depth Counter-Models.

*Journal of Automated Reasoning*, 51(2) :129–149, 2013.

Didier Galmiche, Dominique Larchey-Wendling, and Daniel Méry.

STRIP : Structural sharing and intuitionistic proof-search.

Interne A00-R-342 — galmiche00h, 2000.

Rapport interne.

## Références II

Jean-Yves Girard.

Linear logic : Its syntax and semantics.

In *Proceedings of the Workshop on Advances in Linear Logic*, pages 1–42, New York, NY, USA, 1995. Cambridge University Press.

Jacob M Howe.

Two loop detection mechanisms : a comparison.

In *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 188–200. Springer, 1997.

Dominique Larchey-Wendling.

*Preuves, réfutations et contre-modèles dans des logiques intuitionnistes.*

PhD thesis, Nancy 1, 2000.

Th. doct. : informatique.

# Références III

Paul-André Melliès.

Categorical semantics of linear logic.

*Panoramas et Syntheses*, 27 :15–215, 2009.

Thomas Raths, Jens Otten, and Christoph Kreitz.

The ILTP Library.

<http://www.cs.uni-potsdam.de/ti/iltp/>.

Anne Sjerp Troelstra.

*Lectures on linear logic*.

1992.