Calcul des séquents et transfert vers une catégorie

Mémoire encadré par Paul-André Melliès (PPS)

Recherche de preuves compilée et certifiée

Stage effectué au LORIA de Nancy
sous la direction de Didier Galmiche
et Dominique Larchey-Wendling

Diane Gallois-Wong

L3 — 2014

Introduction

Théorie de la preuve (fin XIX^{e}s.) : formalisation mathématique d'un exposé d'arguments visant à convaincre un interlocuteur de la validité d'une assertion donnée.

Calcul des séquents (Gentzen, 1936)

- Intérêt pratique : algorithme de recherche de preuves (stage)
- Exemple d'approche mathématique : transfert vers la théorie des catégories (mémoire)

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

Formule

Logique classique propositionnelle : formules construites à partir

- de variables propositionnelles ;
- des constantes $\perp$ (*faux*) et $\top$ (*vrai*) ;
- du connecteur unaire $\neg$ (*non*) ;
- des connecteurs binaires $\wedge$ (*et*), $\vee$ (*ou*) et $\rightarrow$ (*implique*).

Logique classique propositionnelle : même construction, en enlevant la constante $\top$, et le connecteur $\neg$

$\neg A$: notation signifiant $A \rightarrow \perp$

Les séquents

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

Définition

Un **séquent** *de* **LK** *consiste en deux listes de formules* Γ *et* Δ*.*
On le note $\Gamma \vdash \Delta$*.*

Formules de Γ : "hypothèses". Formules de Δ : "conclusions".

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

Les règles

Règles de la forme : $\dfrac{\textit{prémisses}}{\textit{conclusion}}\,(\textit{nom de la règle})$

Signification :

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

Exemples de règles de **LK** :

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

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

Preuve d'un séquent

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

- chaque nœud est étiqueté par un séquent ;
- les séquents associés à un nœud donné et à ses fils forment une application d'une règle du calcul.

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

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\,(id)}{A, B \vdash A}\,(weakening\ \ L) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}\,(id)}{B, A \vdash A}\,(weakening\ \ L)}{A, B \vdash A}\,(exchange\ \ L)}{}}{A, B \vdash A \wedge B}\,(\wedge R)$$

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**.

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

Calcul de séquents **LLI**, correspondant à la logique linéaire intuitionniste.

Construction basée sur une preuve constructive donnée du théorème d'élimination de la coupure.

Les séquents du calcul **LLI**

Séquent de **LLI** : une liste de formules Γ et une formule D ; noté $\Gamma \vdash D$.

Formules de la logique linéaire intuitionniste : diffèrent de celles de la logiques classiques (autres connecteurs et constantes) mais ce n'est pas important ici.

Le théorème d'élimination de la coupure

Règle de coupure : $$\frac{\Gamma \vdash A, \Delta \qquad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(cut)$$ dans **LK**

$$\frac{\Gamma \vdash A \qquad \Gamma', A \vdash D}{\Gamma, \Gamma' \vdash D}\,(cut)$$ dans **LLI**

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**.

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**.

Catégorie : définition

Définition

Une **catégorie** *consiste en des* **objets** *(notés A,B,...) et des* **morphismes** *(notés f ,g,...), 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* $f : A \to B$*, et on dit que* $A \to B$ *est le type de* f*.*
- *si* $f : A \to B$ *et* $g : B \to C$*, il existe un morphisme* $g \circ f : A \to C$*.*
- *la loi* $\circ$ *est* **associative** *: si* $f : A \to B$ *et* $g : B \to C$ *et* $h : C \to D$*, alors* $h \circ (g \circ f) = (h \circ g) \circ f$*.*
- *pour tout* A*, il existe une* **identité** $id_A : A \to A$ *vérifiant* $id_A \circ f = f$ *si* $f : B \to A$ *et* $g \circ id_A = g$ *si* $g : A \to B$*.*

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 :

$$p \quad : \qquad \frac{\overset{p_1}{\overline{\cdots\cdots}}\atop A \vdash B \qquad \overset{p_2}{\overline{\cdots\cdots}}\atop B \vdash C}{A \vdash C}\,(cut)$$

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

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 :

$$p \ : \quad \dfrac{\dfrac{p_1}{A \vdash B} \qquad \dfrac{p_2}{B \vdash C}}{A \vdash C}\,(cut)$$

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

Catégorie $\mathcal{CP}$:

- objets : dénotations des formules
- morphismes : dénotations des preuves de séquents de la forme $A \vdash B$, de type $[A] \to [B]$
- identité sur $[A]$: dénotation de $\dfrac{}{A \vdash A}\,(id)$

Catégorie $\mathcal{CP}$:

- objets : dénotations des formules
- morphismes : dénotations des preuves de séquents de la forme $A \vdash B$, de type $[A] \to [B]$
- identité sur $[A]$: dénotation de $\dfrac{}{A \vdash A}\,(id)$

Associativité :

$$\dfrac{\dfrac{\overset{p_1}{A \vdash B} \qquad \overset{p_2}{B \vdash C}}{A \vdash C}\,(cut) \qquad \overset{p_3}{C \vdash D}}{A \vdash D}\,(cut) \qquad\qquad \dfrac{\overset{p_1}{A \vdash B} \qquad \dfrac{\overset{p_2}{B \vdash C} \qquad \overset{p_3}{C \vdash D}}{B \vdash D}\,(cut)}{A \vdash D}\,(cut)$$

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

Propriétés de l'identité sur A :

$$\dfrac{\dfrac{}{A \vdash A}\,(id) \qquad \overset{p}{A \vdash B}}{A \vdash B}\,(cut) \qquad\qquad \overset{p}{A \vdash B}$$

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

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

Perspectives

- À partir de certains connecteurs 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.

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

Prouveur : programme qui prend en entrée une formule et renvoie si elle est prouvable.

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 : 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 en termes d'efficacité.

Langage utilisé : OCaml.

Les séquents de **LSJ**

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

Séquent de **LSJ** : trois multiensembles de formules Θ, Γ et Δ.
Notation : $\Theta ; \Gamma \vdash \Delta$.

Un séquent $\emptyset ; \Gamma \vdash \Delta$ correspond à la formule $\left(\bigwedge_{G \in \Gamma} G\right) \rightarrow \left(\bigvee_{D \in \Delta} D\right)$ 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.
Sémantique d'un tel séquent en termes de modèles de Kripke.

Les règles de **LSJ**

Huit règles : id, $\bot L$, $\wedge L$, $\wedge R$, $\vee L$, $\vee R$, $\to L$, $\to R$.

Seules les règles $\to L$ et $\to R$ agissent sur Θ.

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

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

Séquent de **LSJℓ** : deux multiensembles Γ et Δ de couples "*indice : formule*", et un indice n, les indices étant des entiers naturels. Notation : $\Gamma \vdash_n \Delta$.

$\Gamma' \vdash_n \Delta'$ représente $\Theta\,;\Gamma \vdash \Delta$ où $\begin{cases} \Theta = \Gamma'_{n+1} \\ \Gamma = \Gamma'_{\leq n} \\ \Delta = \Delta'_n \end{cases}$

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

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

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 σ est prouvable : pour toute application de règle possible de la forme $\dfrac{\sigma_1 \quad \dots \quad \sigma_p}{\sigma}$, on détermine si chaque σ_i est prouvable.
S'il y en a une telle que toutes les prémisses sont prouvables (notamment si la règle n'a pas de prémisse), σ est prouvable (on a un arbre de preuve de σ). Sinon, σ n'est pas prouvable.

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

Une prémisse σ_i d'une règle $\dfrac{\sigma_1 \quad \dots \quad \sigma_p}{\sigma}\,(\mathcal{R})$ est **inversible** si on a : si la conclusion σ est prouvable, alors σ_i est aussi prouvable.
Une règle est **inversible** si toutes ses prémisses sont inversibles.

Priorités

Une prémisse σ_i d'une règle $\dfrac{\sigma_1 \quad \dots \quad \sigma_p}{\sigma}\,(\mathcal{R})$ est **inversible** si on a : si la conclusion σ est prouvable, alors σ_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 $\bot L$: pas de prémisse.
- $\wedge L$ et $\vee R$: règles inversibles à une seule prémisse.
- $\wedge R$ 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 $\dfrac{\sigma_1 \quad \dots \quad \sigma_p}{\sigma}\,(\mathcal{R})$ est **locale** si la différence d'information entre σ et n'importe quelle prémisse σ_i est 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 **LSJ**ℓ par rapport à **LSJ** : toutes les règles sont locales.

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

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

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 binaire et A est une sous-formule de B_1 ou de B_2.

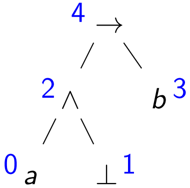
LSJ donc aussi **LSJ**ℓ vérifient la propriété de la sous-formule :
toute formule apparaissant dans une preuve d'un séquent σ est une sous-formule d'une formule de σ.

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	$\bot$	$0 \wedge 1$	b	$2 \to 3$

Efficacité de **LSJ**

Prouveur testé sur les formules de la bibliothèque ILTP [1].

Formule	Temps [2]	Difficulté
SYJ202+1.004	$>$ 300 s	0.00
SYJ207+1.005	9.1 s	0.50
SYJ211+1.008	20 s	0.25
SYJ211+1.009	166 s	0.50
SYJ211+1.010	$>$ 300 s	0.75
SYJ209+1.011	0.1 ms	1.00
SYJ209+1.020	0.4 ms	1.00

Quatre prouveurs de logique intuitionniste propositionnelle sont répertoriés par ILTP.

Difficulté d'une formule : 0.25 * nombre de prouveurs ne terminant pas en moins de cinq minutes.

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 $\boldsymbol{T}$

$\boldsymbol{T}$: 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 $\boldsymbol{T}$ vers un langage exécutable par une machine abstraite assez simple, et d'un programme certifié simulant l'exécution de cette machine abstraite.
$\boldsymbol{M}$: langage associé à cette machine abstraite.

Compilation de fonctions adaptées à la formule

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

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.

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

Différents prouveurs implémentés

- "**Prouveur simple**" : le premier implémenté 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 $\boldsymbol{T}$**" : compilation d'un programme en $\boldsymbol{T}$ adapté à la formule en entrée, exécution à l'aide d'un interpréteur.
- "**Prouveur $\boldsymbol{T}$ $\boldsymbol{M}$**" : compilation du même programme en $\boldsymbol{T}$, compilation de $\boldsymbol{T}$ vers $\boldsymbol{M}$, simulation de l'exécution par la machine abstraite à laquelle $\boldsymbol{M}$ correspond.
- "**Prouveur compilé Caml**" : compilation d'un programme en OCaml adapté à la formule en entrée, compilation par ocamlc, lancement de l'exécutable.

Différentes parties du programme

	Nombre de lignes de code
Utilitaires (gestion des tests)	950
Analyseur ILTP	170
Commun aux prouveurs (93% : indexation)	190
Prouveur simple (sauf commun)	1000
Strutures de données pour le séquent	330
Compilation de fonctions selon la formule	300
Prouveur compilé Caml (sauf commun et compilation fonctions formule)	400
Prouveurs $\boldsymbol{T}$ et $\boldsymbol{T}$ $\boldsymbol{M}$ (sauf commun et compilation fonctions formule)	1400
Code en $\boldsymbol{T}$ fixe	560
Analyseur $\boldsymbol{T}$	220
Total	4200

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
$1^{ère}$ compil.	—	—	0.01 s	0.02 s	0.02 s
$2^{ème}$ 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
$1^{ère}$ compil.	—	—	0.03 s	0.05 s	0.06 s
$2^{ème}$ 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 intuitioinniste 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 $\boldsymbol{T}$.

Références

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.

Paul-André Melliès.
Categorical semantics of linear logic.
Panoramas et Syntheses, 27 :15–215, 2009.

Ce sont les références principales du stage et du mémoire respectivement.
Autres références : cf. document écrit.