Représentation de fonctions

notation: $!^k A = !\ldots!A$ (k fois).

Π preuve de $x : N \vdash t : !^l N$.
on dit que π *représente la fonction* $f : \mathbb{N} \to \mathbb{N}$ si:
pour tout entier n la preuve obtenue en coupant $\vdash \underline{n} : N$ avec Π se réduit en $\vdash \underline{n'} : !^l N$, où $n' = f(n)$.

Proposition 10 *S'il existe une dérivation* Π *de* $x : N \vdash_{IELL} t : !^k N$ *alors* t *représente une fonction élémentairement récursive.*

En effet:
soit $n \in \mathbb{N}$ et Π'_n la preuve de $\vdash_{IELL} (t\ \underline{n}) : !^k N$ obtenue par coupure.
Alors $d(\Pi'_n) = max(1, d(\Pi))$ qui ne dépend pas de n.
De plus, $|\Pi'_n| = n + |\Pi| + cste$.

Theorem 11 *les fonctions représentables dans IEAL sont exactement les fonctions élémentairement récursives.*

Itération dans IEAL

on peut définir pour tout A un itérateur $iter_A$:

$$iter_A = \lambda f x n.\ (n\ f\ x)\ : !(A \multimap A) \multimap !A \multimap N \multimap !A$$

alors $(iter_A\ F\ t)\ \underline{n} \to (F\ (F\ \ldots (F\ t)\ldots))$ (n fois)
exemples:
$double : N \multimap N$
$exp = \lambda n.(iter_N\ double\ \underline{1})\ n\ : N \multimap !N$
remarque: exp ne peut pas être itérée: $(iter \exp\ \underline{1})$ non ELL typable.
$coerc_1 = \lambda n.(iter_N\ succ\ \underline{0})\ n : N \multimap !N$
coercition de N vers $!N$. plus généralement: $coerc_i : N \multimap !^i N$, pour $i \in \mathbb{N}$
conséquence: un terme $\vdash t : !^i N \multimap A$ peut être remplacé par $\vdash t' : N \multimap A$ identique extensionnellement:

$$t' = \lambda n.(t\ (coerc_i\ n))$$

Réduire la complexité ?

comment passer de ELL à une complexité polynomiale ?

essayons de limiter la *propagation des contractions*...

idée: on va distinguer dans les réseaux entre

- des boîtes *non* dupliquables,
- des boîtes dupliquables, mais à 1 seule entrée.

pour cela on considère une nouvelle modalité: §.

Logique linéaire light (LLL) [Girard95]

Version intuitionniste: ILLL.
Langage de formules

$$A, B ::= \alpha \mid A \multimap B \mid A \otimes B \mid !A \mid \S A \mid \forall \alpha.A$$

règles:

$$\frac{B \vdash A}{!B \vdash !A} \qquad \frac{\vdash A}{\vdash !A}$$

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

les autres règles sont inchangées.

- on considérera la version affine : light affine logic (ILAL);
- comme pour IEAL on utilisera ILAL comme un système de types pour le λ-calcul.

ILAL: remarques

- la règle $\S$ peut être vue comme une sorte de déreliction multiple, avec un marqueur $\S$;
- d'un point de vue typage: $!A$ sous-type de $\S A$;
- d'un point de vue sémantique:
  $!$, $\S$ sont des foncteurs, et on a les principes
  $!A \multimap (!A \otimes !A)$
  $!A \multimap \S A \quad \S A \otimes \S B \multimap \S(A \otimes B)$

ILAL et λ-calcul

$$\frac{}{x{:}A \vdash x{:}A}\ Id \qquad \frac{\Gamma_1 \vdash u{:}A \quad x{:}A, \Gamma_2 \vdash t{:}C}{\Gamma_1, \Gamma_2 \vdash t\{x/u\}{:}C}\ Cut$$

$$\frac{\Gamma_1 \vdash u{:}A_1 \quad x{:}A_2, \Gamma_2 \vdash t{:}C}{\Gamma_1, y{:}A_1 \multimap A_2, \Gamma_2 \vdash t\{x/(y\ u)\}{:}C}\ \multimap l \qquad \frac{x{:}A_1, \Gamma \vdash t{:}A_2}{\Gamma \vdash \lambda x.t{:}A_1 \multimap A_2}\ \multimap r$$

$$\frac{x{:}A\{\alpha/B\}, \Gamma \vdash t{:}C}{x{:}\forall\alpha.A, \Gamma \vdash t{:}C}\ \forall l \qquad \frac{\Gamma \vdash t{:}A}{\Gamma \vdash t{:}\forall\alpha.A}\ \forall r \ (\alpha \text{ non libre dans } \Gamma)$$

$$\frac{\Gamma \vdash t{:}C}{\Delta, \Gamma \vdash t{:}C}\ Weak \qquad \frac{x{:}!A, y{:}!A, \Gamma \vdash t{:}C}{z{:}!A, \Gamma \vdash t\{x/z, y/z\}{:}C}\ Cntr$$

$$\frac{x{:}B \vdash t{:}A}{x{:}!B \vdash t{:}!A}\ !r \qquad \frac{\vdash t{:}A}{\vdash t{:}!A}\ !r$$

$$\frac{\Gamma, \Delta \vdash t{:}A}{!\Gamma, \S\Delta \vdash t{:}\S A}\ \S r$$

Traduction ILAL -> IEAL

traduction $(.)^o : ILAL \longrightarrow IEAL$:

- $(!A)^o = (\S A)^o = !A^o$,
- $(.)^o$ commute aux autres connecteurs.

Ceci donne une traduction des preuves ILAL vers les preuves IEAL.
En fait il s'agit même d'une simulation.
Bien sûr, toutes les preuves IEAL ne sont pas dans l'image de $(.)^o$.

exercice: mq toute preuve IEAL *sans coupure* est l'image d'une preuve ILAL par $(.)^o$.

Comme pour IEAL, on a une application d'oubli : $(.)^- : ILAL \to F$.

Types de données ILAL

- entiers unaires

IEAL: N^{IEAL} : $\forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha)$,

ILAL: N^{ILAL} : $\forall\alpha.!(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)$.

- listes binaires

IEAL: W^{IEAL} : $\forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha)$,

ILAL W^{ILAL} : $\forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)$

Exemples

addition

$$\begin{array}{lcl} add & = & \lambda nmfx.(n\ f)\ (m\ f\ x) \\ & : & N \multimap N \multimap N \end{array}$$

double

$$\begin{array}{lcl} double & = & \lambda nfx.(n\ f)\ (n\ f\ x) \\ & : & !N \multimap \S N \end{array}$$

Itération dans ILAL

on a dans ILAL comme type pour l'itérateur:

$$iter_A = \lambda fxn.\ (n\ f\ x) : !(A \multimap A) \multimap \S A \multimap N \multimap \S A$$

$(iter_A\ F\ t)\ \underline{n} \to (F\ (F \ldots (F\ t) \ldots))$ (n fois)

exemples:

$add : N \multimap N \multimap N$, $(add\ \underline{2}) : N \multimap N$

$double' = \lambda n.(iter_N\ (add\ \underline{2})\ \underline{0})\ n\ : N \multimap \S N$

$double'$ ne peut pas être itérée.

similairement:

$m : N \vdash (add\ m) : N \multimap N$, donc $m : !N \vdash (add\ m) : !(N \multimap N)$

la multiplication est obtenue par:

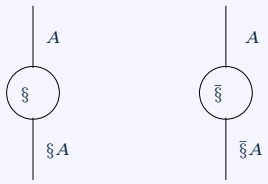
$mult' = \lambda nm.(iter_N\ (add\ m)\ \underline{0})\ n\ : !N \multimap N \multimap \S N$

Structures de preuves LLL

on représente les preuves par des structures de preuves de LLL *classique*.

$$(\S A)^\perp = \bar{\S} A^\perp \quad (\bar{\S} A)^\perp = \S A^\perp$$

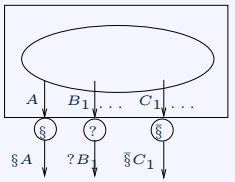
nouveaux noeuds:



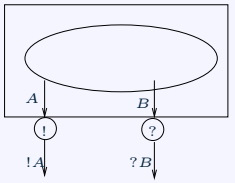
Structures de preuves LLL: boîtes

2 sortes de boîtes exponentielles:

§-boîte:

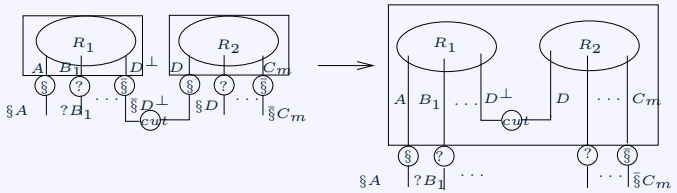


!-boîte:



Réduction des réseaux LLL

comme dans ELL, avec en plus le cas boîte-boîte $\S/\bar{\S}$:



Élimination des coupures (LLL)

Proposition 12 (simulation LLL-Λ) *Si R est un réseau LLL correspondant à une preuve de conclusion $\Gamma \vdash t : A$ et si $R \to R'$ par réduction, alors R' correspond à une preuve de conclusion $\Gamma \vdash t' : A$, avec $t \to t'$.*

Borne de complexité sur la réduction

Comme les réseaux ELL, les réseaux LLL vérifient la ppté de stratification.

Theorem 13 *Si R est un réseau de LLL, de profondeur d, alors la réduction de R en sa forme normale peut être faite en un nombre d'étapes en $O((d+1).|R|^{2^{d+1}})$.*

Remarques:

- si la profondeur est fixée, alors le nombre d'étapes est polynomial en $|R|$;
- cette borne est obtenue avec une stratégie *par niveaux*.

Stratégie de normalisation

convention: pas d'axiome sur formules $!A$, $?A$ (on ajoute des boîtes)
Stratégie similaire à celle utilisée pour ELL:

- tours à profondeurs croissantes, de $i = 0$ à $i = d$.
- **tour** i:
  - phase (a): on élimine les coupures multiplicatives/axiomes/quantificateurs / § à profondeur i,
  - phase (b): on répète l'étape suivante jusqu'à ne plus avoir de coupure exponentielle à prof. i:

    étape: on réduit une coupure maximale pour $\prec$ à prof. i.

Calcul de bornes

dém. du théorème sur le nombre d'étapes de normalisation.
Soit R un réseau sans coupure à profondeur $< i$ et avec uniquement des coupures $!/?$ à profondeur i.

Definition 2 *Soit N un noeud ! ou § à prof. i dans R. On appelle potentiel de N, $pot(N)$:*

- $pot(N) = 0$ *si N est un noeud § ou ! sans coupure dessous,*
- *sinon* $pot(N) = c + pot(N_1) + \cdots + pot(N_n)$, *où:*
  - *c est le nombre de noeuds contractions de l'arbre coupé contre N,*
  - *les N_i sont les portes principales des boîtes à prof. i dont une porte auxilliaire est au-dessus de la coupure contre N.*

Remarque: $pot(N)$ est bien définie car $\prec$ est un ordre partiel.

Bornes: suite

Lemma 14

$$pot(N) \leq |R|_i - 1.$$

Definition 3 *Soit $B_i(R) = \sum_N (1 + 2.pot(N))$, où la somme porte sur les portes principales N de boîtes exp. à profondeur i.*

Lemma 15

$$B_i(R) \leq 2|R|_i^2.$$

Lemma 16 *$B_i(R)$ décroît strictement à chaque étape de la phase b (réduction de coupure exponentielle).*

Bornes: suite

Soit $t_i(R) = \sum_M (1 + pot(N_M))$, où:

- la somme porte sur les noeuds M à prof. $\geq (i+1)$,
- N_M est la porte principale de la boîte à prof. i contenant M.

Lemma 17 *Si R (resp. R') est le réseau au début (resp. à la fin) du tour i:*

- *$t_i(R)$ n'augmente pas au cours du tour i;*
- $|R'|_{(i+1)+} \leq t_i(R) \leq |R|_{i+}^2$.

Notons $R^{(i)}$ le réseau à la fin du tour i.

Lemma 18 *Pour $0 \leq i \leq (d-1)$ on a:* $|R^{(i)}|_{(i+1)+} \leq |R|^{2^{i+1}}$.

Le nombre d'étapes de la procédure est majoré par:

$$x = 3.|R|_0^2 + 3.|R^{(0)}|_1^2 + 3.|R^{(1)}|_2^2 + \cdots + 3.|R^{(d-1)}|_d^2$$

on obtient: $x \leq 3(d+1)|R|^{2^{d+1}}$.

Bornes: suite

Proposition 19 *Un réseau R de LLL de profondeur d peut être réduit en temps $O((d+1)|R|^{2^{d+2}})$.*

notation: $\S^k A = \S \ldots \S A$ (k fois).

Conséquence:

si $\vdash t : W \multimap \S^k W$ alors t représente une fonction de FP (calculable en

temps polynomial).