LL et paradigmes logiques du calcul
Partie III: Logique linéaire, types et complexité
MPRI

Patrick Baillot

LIPN

CNRS - Université Paris 13

patrick.baillot@lipn.univ-paris13.fr

http://www-lipn.univ-paris13.fr/~baillot/

29/11/2006

Introduction

la LL est née de la décomposition de $\Rightarrow$ en $\multimap$ et ! :

analyse.

un statut logique est ainsi donné à la duplication d'argument

$\longrightarrow$ *synthèse*: modifier les règles du ! pour avoir une discipline plus stricte:
duplication modérée et *via élimination des coupures* caractérisation de classes de complexité.

Introduction (2)

pourquoi faire ...?

- un modèle logique, *implicite*, structuré, de caractérisation de la complexité, alternatif aux machines de Turing, circuits booléns. . . domaine de la *complexité implicite*
- étendre la correspondance de Curry-Howard à Ptime et aux classes de complexité, avec comme bénéfices associés:
  - langage de *programmation* ptime, ou *types* pour la vérification de complexité
  - systèmes pour *preuves* de terminaison ptime
  - sémantique: compréhension mathématique du calcul ptime: catégories, jeux ?

Quelques objectifs de cette partie du cours

- une application des réseaux de preuves pour l'étude du lambda-calcul,
- calculer avec (des variantes de) la LL,
- illustrer l'utilisation de la LL pour les systèmes de types,
- applications de la LL à la complexité.

Plan

1. Logique linéaire élémentaire (ELL) et complexité élémentairement récursive:
   un premier pas, un système simple . . .
2. Logique linéaire light (LLL) et complexité Ptime:
   la complexité Ptime, via les réseaux de preuves.
3. types light simplifiés (DLAL):
   la complexité Ptime, dans le λ-calcul.
4. types linéaires pour le *calcul en place*.

Préliminaires: machines de Turing, complexité

notations sur les listes:
liste: $\langle a_1, \ldots, a_n \rangle$, concaténation: $l_1 :: l_2$
$|l|$ longueur de la liste.

Definition 1 *Machine de Turing: $\mathcal{M} = (Q, \Sigma, \delta)$, où:*

- Q *ensemble fini d'états, contenant q_i (resp. q_f) état initial (resp. état final),*
- $\Sigma = \{0, 1, \mathtt{b}\}$ *alphabet,*
- $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R\}$ *fonction de transition.*

Configurations: $config = Q \times \Sigma^\star \times \Sigma^\star$

$(q, l_1, l_2) \in config$

l_1 (resp. l_2) est la partie gauche (resp. droite) de la bande de lecture.
Le symbole lu est le premier élément de l_2.

Préliminaires: suite

À partir de δ on définit: $step : config \to config$
configuration initiale (entrée= $w_0 \in \{0,1\}^\star$): $C_0 = (q_i, \langle\,\rangle, w_0)$

- $\mathcal{M}$ à partir de C_0 *s'arrête* sur $C' = step^n(C_0)$ ssi n est le plus petit entier tel que $step^n(C_0)$ a pour état q_f.
- $g : \{0,1\}^\star \to \{0,1\}^\star$
  $\mathcal{M}$ *représente* g ssi pour tout $w_0 \in \{0,1\}^\star$, $\mathcal{M}$ à partir de C_0 s'arrête sur $C = (q_f, w_1, w_2)$, avec:
  - $w_1 = g(w_0)$,
  - $w_2 = \langle \mathtt{b}, \ldots, \mathtt{b} \rangle$
- $\phi : \mathbb{N} \to \mathbb{N}$
  $\mathcal{M}$ *est de complexité en temps* ϕ ssi pour tout w_0, $\mathcal{M}$ à partir de C_0 s'arrête après au plus $\phi(|w_0|)$ étapes.

Préliminaires: suite

$g : \{0,1\}^\star \to \{0,1\}^\star$
La fonction g est *calculable en temps* ϕ (de complexité ϕ) s'il existe une machine $\mathcal{M}$ de complexité en temps ϕ qui la représente.

$FP = \{$fonctions $\{0,1\}^\star \to \{0,1\}^\star$ calculables en temps polynomial$\}$

$P = \{$fonctions $\{0,1\}^\star \to \{0,1\}$ calculables en temps polynomial$\}$

tour d'exponentielles: $K(d, n) = 2^{2^{\cdots^{2^n}}}$ (hauteur d)

Fonctions de temps *élémentaire* (ou *élémentairement récursives*):

$FElem = \{$fonctions g t.q. $\exists d$ t.q. g est calculable en temps $K(d, .)\}$

ex. de fonction primitive récursive mais non élémentaire: $n \to K(n, 2)$.

Logique linéaire élémentaire (ELL)

- correspond aux fonctions *élémentaires* : temps $2^{2^{\cdots^{2^n}}}$, à hauteur fixée...
- des applications en réduction optimale: simplification de l'algorithme de Lamping,
- GoI (géométrie de l'interaction)/ algèbre dynamique plus simples que pour LL.

ELL : définition

Pour ELL on garde le même langage de formules que pour LL, mais on va changer les règles d'introduction des modalités !, ?:

- on enlève une règle: déreliction,
- et on remplace la promotion par :

$$\frac{\vdash B_1, \ldots, B_n, A}{\vdash ?B_1, \ldots, ?B_n, !A}\ !\mathrm{E}$$

- la contraction et l'affaiblissement sont inchangés.

exercice: *on considère la règle digging:*

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

Mq le système avec comme règles exponentielles: ($!E$*, dig, der, contr., affaibl.) est équivalent à LL.*

ELL : quelques propriétés

- les principes $!A \multimap A$ (déreliction) et $!A \multimap !!A$ (digging) ne sont pas démontrables dans ELL;
- contrairement à LL : un nombre infini de modalités (suites de $!, ?$) à équivalence près;
- étapes d'élimination des coupures: $!E/!E$, $!E/contr.$, $!E/affaibl.$
- traduction ELL $\rightarrow$ LL, compatible avec élimination des coupures (*simulation* de ELL vers LL) (à faire en exercice).

IELL : version intuitionniste

pour étudier l'expressivité de ELL on se restreint à une version intuitionniste: IELL, avec 2nd ordre (à la système F).
le langage de formules est celui de ILL (sans additifs)

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

Ainsi:

- séquents de la forme $\Gamma \vdash A$
- pour avoir plus de souplesse, on peut ajouter l'affaiblissement général : logique *affine* élémentaire (IEAL);
- pour calculer : on peut voir IELL/IEAL comme un système de types pour le λ-calcul.
- la règle pour ! s'écrit:

$$\frac{B_1, \ldots, B_n \vdash A}{!B_1, \ldots, !B_n \vdash !A}\ !\ \mathrm{E}$$

Lambda-calcul

- lambda-termes:

$$t, u ::= x \mid \lambda x.t \mid (t\ u)$$

notations: $\lambda x_1 x_2.t$ pour $\lambda x_1.\lambda x_2.t$

$(t\ u\ v)$ pour $((t\ u)\ v)$

substitution (du terme u à var. x): $t\{x/u\}$

substitutions simultanées $t\{x/u_1, y/u_2\}$

substitution dans une formule $A\{\alpha/B\}$

$A \multimap B \multimap C$ pour $A \multimap (B \multimap C)$

- β-réduction:
clôture par contexte de:

$$(\lambda x.t)\ u \xrightarrow{1} t\{x/u\}$$

$\rightarrow$ clôture réflexive et transitive de $\xrightarrow{1}$.

IEAL et λ-calcul

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

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

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

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

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

Dans les règles binaires: contextes (Γ_1 et Γ_2) des deux prémisses ont variables disjointes.

Opération d'oubli: de IEAL à F

application $(.)^- : IEAL \rightarrow F$ définie par:

$$(!A)^- = A^-, \quad (A \multimap B)^- = A^- \rightarrow B^-,$$

Proposition 1 *Si* $\Gamma \vdash_{IEAL} t : A$ *alors* $\Gamma^- \vdash_F t : A^-$.

Si $A^- = T$: A est une *décoration* de T dans IEAL.

Types de données IEAL

- entiers unaires

système F:	ILL:	IEAL:
N^F		N^{IEAL}
$\forall\alpha.(\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)$	$\forall\alpha.!(\alpha \multimap \alpha) \multimap (\alpha \multimap \alpha)$	$\forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha)$

Exemple: $\underline{2}$, dans F:

$$\underline{2} = \lambda f^{(\alpha\rightarrow\alpha)}.\lambda x^{\alpha}.(f\ (f\ x))\ .$$

- listes binaires

système F:	IEAL:
W^F	W^{IEAL}
$\forall\alpha.(\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)$	$\forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha)$

Exemple: $w = \langle 1, 0, 0\rangle$, dans F:

$$\underline{w} = \lambda s_0^{(\alpha\rightarrow\alpha)}.\lambda s_1^{(\alpha\rightarrow\alpha)}.\lambda x^{\alpha}.(s_1\ (s_0\ (s_0\ x)))\ .$$

Structures de preuves ELL

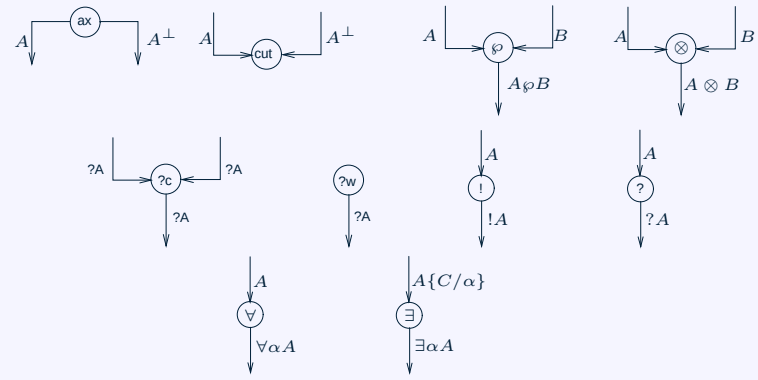
on va représenter les preuves par des réseaux de ELL *classique*

$A \multimap B = A^{\perp} \wp B$

$A^{\perp}$ défini par:

$$(A \otimes B)^{\perp} = A^{\perp} \wp B^{\perp} \quad (A \wp B)^{\perp} = A^{\perp} \otimes B^{\perp}$$
$$(\forall \alpha.A)^{\perp} = \exists \alpha.A^{\perp} \quad (\exists \alpha.A)^{\perp} = \forall \alpha.A^{\perp}$$
$$(!A)^{\perp} = ?A^{\perp} \quad (?A)^{\perp} = !A^{\perp}$$
$$\alpha^{\perp\perp} = \alpha$$

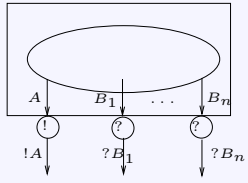
Structures de preuves ELL



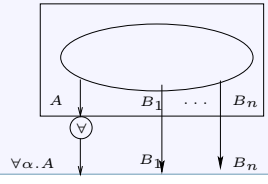
Structures de preuves: boîtes

2 boîtes sont disjointes ou l'une est incluse dans l'autre.

- boîte exponentielle (!-boîte)

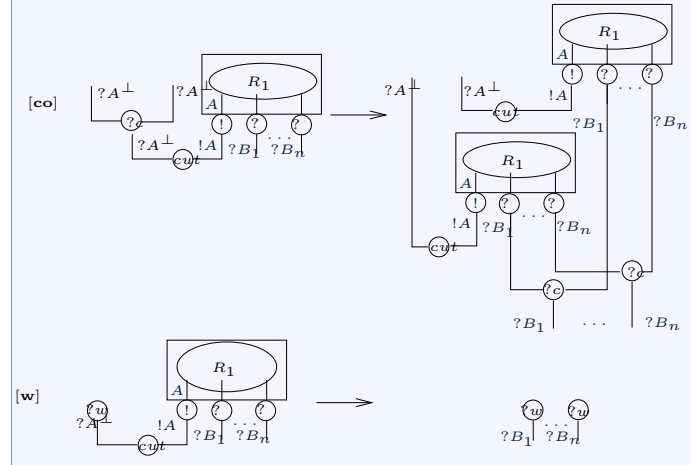


- boîte $\forall$: $\alpha \notin FV(B_i), 1 \leq i \leq n$



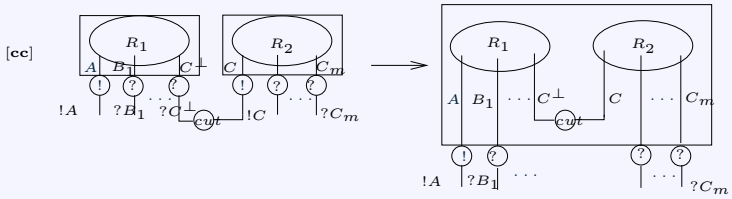
Réduction des réseaux

réduction de coupure contraction ou affaibl.: comme dans réseaux LL



Réduction des réseaux (suite)

réduction de coupure boîte-boîte: fusion



Exemples de termes

addition

$$add = \lambda nmfx.(n\ f)\ (m\ f\ x)$$
$$: N \multimap N \multimap N$$

multiplication

$$mult = \lambda nmf.(n\ (m\ f))$$
$$: N \multimap N \multimap N$$

carré

$$square = \lambda nf.(n\ (n\ f))$$
$$: !N \multimap !N$$

Normalisation des réseaux et réduction des termes

Proposition 2 (simulation ELL-Λ) *Si R est un réseau ELL 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 $t \to t'$.*

Proposition 3 *Si R est un réseau ELL correspondant à une preuve de $\Gamma \vdash t : A$ et si R est sans coupure, alors t est sous forme normale.*

Proposition 4 *Si $\Gamma \vdash t : A$, $t \to t'$ et t' est sous forme normale alors $\Gamma \vdash t' : A$.*

Normalisation des réseaux

R réseau ELL. e arête de R.
profondeur de e, $d(e)$: nombre de boîtes exponentielles qui contiennent e.
profondeur de noeud N: idem.
profondeur de R, $d(R)$: maximum des profondeurs de ses arêtes.

taille de R, $|R|$: nombre de noeuds de R
$|R|_i$: nombre de noeuds à profondeur i
$|R|_{i+}$: nombre de noeuds à profondeur $\geq i$

Proposition 5 (Stratification) *La profondeur d'une arête d'un réseau ELL ne change pas après une étape de réduction.*

Noter que ceci n'est pas vrai dans LL: la profondeur peut diminuer (étape déreliction) ou augmenter (étape boîte-boîte).

Borne de complexité sur la réduction

$K(0,n) = n$, $K(k+1,n) = 2^{K(k,n)}$.

Theorem 6 *Si R est un réseau de ELL, alors la réduction de R en sa forme normale peut être faite en un nombre d'étapes inférieur à $K(d+1, |R|)$.*

Remarques:

- hauteur de la tour ne dépend que de la profondeur, et pas de la taille;
- aucune référence aux formules apparaissant sur le réseau;
- cette borne est obtenue en appliquant une stratégie *par niveaux*.

Définitions pour la normalisation

?-noeud: noeud contraction, affaiblissement ou porte auxilliaire de !-boîte.
On dira qu'un ?-noeud dans R est *actif* s'il est *au-dessus* d'un noeud cut.

On considère R sans coupure à prof. $\leq i-1$, et avec uniquement des coupures exponentielles à profondeur i.
On définit $\prec_1$ sur les noeuds cut à prof. i par:

$N \prec_1 N'$ ssi:
il existe une !-boîte à prof. i dont N est au-dessous de la porte principale, et N' est au-dessous d'une porte auxill.

soit $\prec$ la clôture réflexive et transitive de $\prec_1$.

Lemma 7 *la relation $\prec$ est un ordre partiel.*

Stratégie de normalisation par niveaux

convention: pas d'axiome sur formules $!A$, $?A$ (on ajoute des boîtes)

- on procède par tours à profondeurs croissantes, de $i=0$ à $i=d$. à la fin du tour i: le réseau n'a plus de coupures à profondeur $\leq i$
- **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.

Cette stratégie est normalisante.

Calcul de bornes sur nombre d'étapes

dém. du théorème 6.

- tour i, phase (a): $|R|_i$ diminue strictement à chaque étape, donc le nombre d'étapes est majoré par $|R|_i$;
- tour i, phase (b):

  Lemma 8 *une étape de la phase (b) diminue strictement le nombre de ?-noeuds actifs.*

  Lemma 9 *si R est le réseau au début de phase (b), le nombre d'étapes de phase (b) est majoré par $|R|_i$.*

  Fait: à chaque étape de phase (a), $|R|_{(i+1)+}$ est inchangée.
  Fait: à chaque étape de phase (b), $|R|_{(i+1)+}$ est au plus multiplié par 2.
  Notons $R^{(i)}$ le réseau à la fin du tour i.