Université de Paris Cité M1 Informatique

Preuves assistées par ordinateur

Examen du 17 mai 2024 – durée : 3h

Documents autorisés

Calcul des séquents et machines abstraites

On se propose de montrer que le calcul des séquents LK de Gentzen est en correspondance preuve-programme avec la notion de « machine abstraite » utilisée pour l'interprétation bas niveau des programmes en compilation.

On commencera par isoler une version nommée LJT de la version intuitionniste LJ du calcul des séquents qui se trouve être en correspondance avec une machine abstraite standard utilisée pour la compilation du λ-calcul en « appel par nom ».

On continuera par montrer que cette machine abstraite pour l'appel par nom se généralise de manière naturelle aux « opérateurs de contrôles » qui eux-mêmes correspondent à la logique classique.

On terminera par la mise au point d'une variante LK_{tq} de LK que l'on peut mettre en correspondance avec un λ-calcul symétrique avec contrôle relativement simple. Ce calcul généralisé permettra de compiler tant en appel par nom qu'en appel par valeur.

Notez que plusieurs questions peuvent être traitées indépendamment.

A – Le calcul des séquents de Gentzen

Introduit en 1934, le calcul des séquents LK de Gentzen est le premier système logique à être équipé d'une procédure de normalisation. Dans sa version originale, c'est un formalisme pour la logique classique construit sur des séquents de la forme $\Gamma \vdash \Delta$ où Γ et Δ sont des listes de formules respectivement interprétées comme conjonction et disjonction de leurs composantes.

On se restreint à un ensemble de formules propositionnelles construites à partir d'atomes, d'implications et de conjonctions :

$$A, B, C \quad ::= \quad X \mid A \to B \mid A \wedge B$$

Les contextes d'hypothèses Γ et de conclusions Δ étant définis comme des listes de formules et la virgule « Γ, Γ' » comme une concaténation de listes, on peut énoncer les règles d'inférence de LK comme suit :

$$\frac{}{\Gamma, A, \Gamma' \vdash \Delta, A, \Delta'}\ Ax \qquad \frac{\Gamma, \Gamma' \vdash \Delta, A, \Delta \qquad \Gamma, A, \Gamma' \vdash \Delta, \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ Cut$$

$$\frac{\Gamma \vdash \Delta, A \to B, \Delta', B}{\Gamma \vdash \Delta, A \to B, \Delta'}\ \to_R \qquad \frac{\Gamma, A \to B, \Gamma' \vdash \Delta, A \qquad \Gamma, A \to B, \Gamma', B \vdash \Delta}{\Gamma, A \to B, \Gamma' \vdash \Delta}\ \to_L$$

$$\frac{\Gamma \vdash \Delta, A \wedge B, \Delta', A \qquad \Gamma \vdash \Delta, A \wedge B, \Delta', B}{\Gamma \vdash \Delta, A \wedge B, \Delta'} \wedge_R$$

$$\frac{\Gamma, A \wedge B, \Gamma', A \vdash \Delta}{\Gamma, A \wedge B, \Gamma' \vdash \Delta} \wedge_L^1 \qquad \frac{\Gamma, A \wedge B, \Gamma', B \vdash \Delta}{\Gamma, A \wedge B, \Gamma' \vdash \Delta} \wedge_L^2$$

où les règles de contraction, affaiblissement et échange sont intégrées aux autres règles.

La version intuitionniste LJ de LK s'obtient en se restreignant à une formule à droite et en cessant de garder une copie de la formule introduite dans les règles d'introduction à droite (donc Δ vide pour les séquents mentionnant explicitement un A à droite et Δ réduit à une seule formule sinon).

B – LJT : une version appel par nom du calcul des séquents intuitionniste de Gentzen

On se propose de mettre un peu de structure dans LJ en isolant une version « par nom » qui correspondra à un λ-calcul évalué en appel par nom.

Pour formuler LJT, nous reposons sur deux types de séquents :

— $\Gamma \vdash A$ qui porte un focus sur la conclusion

— $\Gamma; A \vdash B$ qui porte un focus sur une hypothèse particulière A

Les règles structurelles de LJT sont les suivantes (notez en particulier la présence de deux règles d'axiomes) :

$$\frac{A \in \Gamma}{\Gamma \vdash A}\ Ax_R \qquad \frac{}{\Gamma; A \vdash A}\ Ax_L \qquad \frac{\Gamma \vdash A \qquad \Gamma; A \vdash B}{\Gamma \vdash B}\ Cut$$

On donne aussi les règles logiques pour la conjonction et l'implication :

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \to_R \qquad \frac{\Gamma \vdash A \qquad \Gamma; B \vdash C}{\Gamma; A \to B \vdash C} \to_L$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge_R \qquad \frac{\Gamma; A \vdash C}{\Gamma; A \wedge B \vdash C} \wedge_L^1 \qquad \frac{\Gamma; B \vdash C}{\Gamma; A \wedge B \vdash C} \wedge_L^2$$

On se donne le début d'une définition en Coq d'un type inductif représentant les preuves de LJT.

```
Inductive formula :=
| Atom : nat -> formula
| Impl : formula -> formula -> formula
| And : formula -> formula -> formula.

Definition context := list formula.

Reserved Notation "A ∈ Γ" (at level 70).
Reserved Notation "Γ ⊢_R A" (at level 70).
Reserved Notation "Γ ; A ⊢_L B" (at level 70).

Inductive incontext : formula -> context -> Type :=
```

```
| InNow A Γ : A ∈ (A::Γ)
| InLater A B Γ : A ∈ Γ -> A ∈ (B::Γ)
where "A ∈ Γ" := (incontext A Γ).

Inductive seqR : context -> formula -> Type :=
| AxR Γ A : A ∈ Γ -> Γ ⊢_R A
...
with seqL : context -> formula -> formula -> Type :=
| AxL Γ A : Γ ; A ⊢_L A
...
where "Γ ⊢_R A" := (seqR Γ A)
and "Γ ; A ⊢_L B" := (seqL Γ A B).
```

1. Complétez les « ... » dans la définition précédente, incluant la règle de coupure et les règles pour l'implication et la conjonction.
2. Montrer, en l'exprimant sous la forme de fonctions des prémisses vers la conclusion que les variantes suivantes de coupure sont admissibles :

$$\frac{\Gamma, \Gamma' \vdash A \qquad \Gamma, A, \Gamma' \vdash B}{\Gamma, \Gamma' \vdash B}\ Cut_R^S \qquad \frac{\Gamma, \Gamma' \vdash A \qquad \Gamma, A, \Gamma'; B \vdash C}{\Gamma, \Gamma'; B \vdash C}\ Cut_L^S$$

   On pourra exprimer ces fonctions soit informellement (par analyse de cas sur la forme des prémisses) soit comme des fonctions mutuellement récursives en Coq (utilisant la fonction de concaténation de listes `++` pour la concaténation de contextes).
3. Montrer, en l'exprimant sous la forme d'une fonction des prémisses vers la conclusion que la quatrième forme suivante de coupure est aussi admissible :

$$\frac{\Gamma, \Gamma'; A \vdash B \qquad \Gamma; B \vdash C}{\Gamma, \Gamma'; A \vdash C}\ Cut_P$$

C – Machine abstraite appel par nom

On appelle machine abstraite un système de réécriture sur des états composés d'un « accumulateur » (= un programme = un terme) et une « pile » (= un contexte d'évaluation pour ce programme). Il existe une machine abstraite simple pour l'appel par nom appelée « machine de Krivine » (KAM) et définie comme suit :

$$\begin{array}{lcll} c & ::= & \langle t \| e \rangle & \text{états} \\ t, u & ::= & x \mid \lambda x.t \mid (t, u) \mid te & \text{programmes} \\ e, f & ::= & \mathsf{empty} \mid t \cdot e \mid \pi_1 \cdot e \mid \pi_2 \cdot e & \text{piles} \end{array}$$

où, intuitivement, vu du λ-calcul habituel, e représente une suite d'éliminations appliquée à un trou [ ], avec empty représentant le trou [ ] tout seul (= une pile vide), et où te représente le terme du λ-calcul habituel obtenu en remplaçant [ ] par t dans e.

Les règles de réécriture suivantes s'appliquent alors aux états :

$$\begin{array}{lcl} \langle \lambda x.t \| u \cdot e \rangle & \to & \langle t[x := u] \| e \rangle \\ \langle (t_1, t_2) \| \pi_1 \cdot e \rangle & \to & \langle t_1 \| e \rangle \\ \langle (t_1, t_2) \| \pi_2 \cdot e \rangle & \to & \langle t_2 \| e \rangle \\ \langle te \| e' \rangle & \to & \langle t \| e @ e' \rangle \end{array}$$

où la substitution $t[x := u]$ est définie comme d'habitude par récursion sur t et où $e@e'$ dénote la concaténation de piles, c'est-à-dire le remplacement de empty dans e par e'.

En particulier, on peut remarquer que les états entre termes et piles où aucune règle ne s'applique sont les états de la forme $\langle V \| \mathsf{empty} \rangle$ avec V une valeur décrite par la syntaxe $V ::= \lambda x.t \mid (t_1, t_2)$ ou ceux de la forme $\langle x \| e \rangle$ qui bloquent en attente d'une valeur pour x.

En pratique, pour faciliter la manipulation formelle des termes, on va utiliser des variables de de Bruijn pour les lieurs, c'est-à-dire qu'une variable x associée à un lieur $\lambda x.t$ sera représentée par un entier n indiquant le nombre de λ à sauter avant de trouver le bon λ. En particulier, le λ va cesser aussi de déclarer un nom puisque seule sa position par rapport aux autres λ va maintenant compter. La nouvelle syntaxe est :

$$\begin{array}{lcl} c & ::= & \langle t \| e \rangle \\ t, u & ::= & n \mid \lambda t \mid (t, u) \mid te \\ e, f & ::= & \mathsf{empty} \mid t \cdot e \mid \pi_1 \cdot e \mid \pi_2 \cdot e \end{array}$$

4. Définir dans la syntaxe de Coq des types mutuellement inductifs `termn` et `stackn` caractérisant la syntaxe des termes et piles de la KAM appel par nom avec index de de Bruijn.
5. Écrire une fonction récursive qui prend un état et applique une règle de réécriture tant que c'est possible pour retourner une valeur si jamais le calcul s'arrête sur une forme $\langle V \| \mathsf{empty} \rangle$. Pour garantir la terminaison, on paramétrera la fonction par un entier N (du « fuel ») qui dit de s'arrêter quand même au pire au bout de N réductions. On donnera le résultat dans un type option :

   ```
   Inductive option A := None : option A | Some : A -> option A.
   ```

   en renvoyant `None` si le calcul ne retourne pas une valeur, et retournera `Some` de la valeur sinon.

On pourrait ensuite montrer que la KAM simule la réduction des λ-termes habituels quand tu est interprété par $t(u \cdot \mathsf{empty})$ et $\pi_i(t)$ par $t(\pi_i \cdot \mathsf{empty})$ mais ce ne sera pas notre propos aujourd'hui.

Au contraire, on va considérer un système de typage LJT^{expl} pour la KAM (basé sur la syntaxe avec de de Bruijn). Ce système a lui aussi deux types de séquents, $\Gamma \vdash t : A$ pour typer les termes et $\Gamma; A \vdash e : B$ pour typer les piles attendant un argument de type A pour retourner un résultat de type B.

$$\frac{|\Gamma'| = n}{\Gamma, A, \Gamma' \vdash n : A}\ Var \qquad \frac{}{\Gamma; \mathsf{empty} : A \vdash A}\ Empty \qquad \frac{\Gamma \vdash t : A \qquad \Gamma; e : A \vdash B}{\Gamma \vdash te : B}\ Comp$$

$$\frac{\Gamma, A \vdash t : B}{\Gamma \vdash \lambda t : A \to B}\ Abs \qquad \frac{\Gamma \vdash t : A \qquad \Gamma; e : B \vdash C}{\Gamma; t \cdot e : A \to B \vdash C}\ App$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash (t, u) : A \wedge B}\ Pair \qquad \frac{\Gamma; e : A \vdash C}{\Gamma; \pi_1 \cdot e : A \wedge B \vdash C}\ Proj_1 \qquad \frac{\Gamma; e : B \vdash C}{\Gamma; \pi_2 \cdot e : A \wedge B \vdash C}\ Proj_2$$

On a clairement une correspondance entre LJT et LJT^{expl}, le second explicitant une syntaxe de terme pour les preuves du premier.

6. Définir dans la syntaxe de Coq des types mutuellement inductifs

```
termn_of_type : context -> termn -> formula -> Type
stackn_of_type : context -> stackn -> formula -> formula -> Type
```

caractérisant le système de typage des termes et piles de la KAM appel par nom.

7. Écrire des fonctions récursives $\mathcal{E}_R$ et $\mathcal{E}_L$ qui extraient respectivement de dérivations p et q de $\Gamma \vdash A$ et $\Gamma; A \vdash B$ un terme $\mathcal{E}_R(p)$ et une pile $\mathcal{E}_L(q)$ tels que $\Gamma \vdash \mathcal{E}_R(p) : A$ et $\Gamma; \mathcal{E}_L(q) : A \vdash B$.

En passant, on dira qu'une preuve est η-expansée si les axiomes sont restreints au cas d'atomes. On peut se donner des règles d'η-expansion qui expansent tout axiome portant sur un connecteur particulier en une preuve équivalente dont les axiomes portent sur les composantes du connecteur.

Formulées dans le langage des termes, les η-expansions ont la forme suivante :

$$\begin{array}{lcll} x & \to & \lambda y.x(y \cdot \mathsf{empty}) & \text{si } x \text{ est de type } A \to B \\ x & \to & (x(\pi_1 \cdot \mathsf{empty}), x(\pi_2 \cdot \mathsf{empty})) & \text{si } x \text{ est de type } A \wedge B \end{array}$$

8. Écrire une fonction récursive *eta* qui à toute formule A associe la preuve η-expansée de $A \vdash A$ (on pourra raisonner au choix dans LJT^{expl} ou dans LJT).

On peut aussi regarder quelques exemples particuliers.

9. Donner des preuves de $A \to B \to A$ et de $(A \to B \to C) \to (A \to B) \to (A \to C)$ en se plaçant au choix dans LJT^{expl} ou dans LJT en explicitant par ailleurs le terme associé dans la KAM.

D – LKT : une version appel par nom du calcul des séquents classique de Gentzen

Pour formuler LKT, nous relâchons la contrainte sur le nombre de conclusions et reposons sur trois types de séquents :

— $\Gamma \vdash \Delta; A$ qui porte un focus sur une conclusion particulière A

— $\Gamma; A \vdash \Delta$ qui porte un focus sur une hypothèse particulière A

— $\Gamma \vdash \Delta$ qui ne porte de focus sur aucune formule

Les règles structurelles de LKT sont les suivantes, où la coupure a été décomposée en deux morceaux, transitant par un séquent sans aucun focus :

$$\frac{A \in \Gamma}{\Gamma \vdash \Delta; A}\ Ax_R \qquad \frac{A \in \Delta}{\Gamma; A \vdash \Delta}\ Ax_L$$

$$\frac{\Gamma \vdash \Delta; A \qquad \Gamma; A \vdash \Delta}{\Gamma \vdash \Delta}\ Cut \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta; A}\ Focus$$

On donne aussi les règles logiques pour la conjonction et l'implication :

$$\frac{\Gamma, A \vdash \Delta; B}{\Gamma \vdash \Delta; A \to B}\ \to_R \qquad \frac{\Gamma \vdash \Delta; A \qquad \Gamma; B \vdash \Delta}{\Gamma; A \to B \vdash \Delta}\ \to_L$$

$$\frac{\Gamma \vdash \Delta; A \qquad \Gamma \vdash \Delta; B}{\Gamma \vdash \Delta; A \wedge B}\ \wedge_R \qquad \frac{\Gamma; A \vdash \Delta}{\Gamma; A \wedge B \vdash \Delta}\ \wedge^1_L \qquad \frac{\Gamma; B \vdash \Delta}{\Gamma; A \wedge B \vdash \Delta}\ \wedge^2_L$$

E – Machine abstraite appel par nom avec contrôle

On étend la machine de Krivine avec un opérateur de « contrôle » qui permet de lier la pile à une variable pour la réutiliser plus tard. La syntaxe modifiée est :

$$\begin{array}{lcll} c & ::= & \langle t \| e \rangle & \text{états} \\ t, u & ::= & x \mid \lambda x.t \mid (t,u) \mid \mu\alpha.c & \text{programmes} \\ e, f & ::= & \alpha \mid t \cdot e \mid \pi_1 \cdot e \mid \pi_2 \cdot e & \text{piles} \end{array}$$

où l'ancien te a été repensé comme étant un terme qui capture la pile courante et continue avec l'évaluation de l'état $\langle t \| e \rangle$ dans lequel empty a été remplacé par la pile courante, c'est-à-dire où te est généralisé en un $\mu\alpha.\langle t \| e \rangle$ avec μ un opérateur liant la pile, de telle sorte que l'ancien te soit repensé comme $\mu\alpha.\langle t \| e[\mathsf{empty} := \alpha] \rangle$.

Les règles de réécriture sont modifiées pour capturer explicitement la pile :

$$\begin{array}{lcl} \langle \lambda x.t \| u \cdot e \rangle & \to & \langle t[x := u] \| e \rangle \\ \langle (t_1, t_2) \| \pi_1 \cdot e \rangle & \to & \langle t_1 \| e \rangle \\ \langle (t_1, t_2) \| \pi_2 \cdot e \rangle & \to & \langle t_2 \| e \rangle \\ \langle \mu\alpha.c \| e \rangle & \to & c[\alpha := e] \end{array}$$

où l'on voit effectivement que la dernière règle appliquée à $\langle \mu\alpha.\langle \overline{t \| e} \rangle \| e' \rangle$ simule bien l'ancienne règle quand le empty dans l'ancien e est maintenant un α (autrement dit, c'est maintenant la substitution de α par e' qui réalise l'ancienne concaténation de e et e').

On considère maintenant le système de typage LKT^{expl} suivant pour la KAM avec contrôle (avec indices de de Bruijn à la fois pour les variables de terme et les variables de piles) :

$$\frac{|\Gamma'| = n}{\Gamma, A, \Gamma' \vdash \Delta; n : A}\ Var_R \qquad \frac{|\Delta'| = n}{\Gamma; n : A \vdash \Delta, \Delta'}\ Var_L$$

$$\frac{\Gamma \vdash \Delta; t : A \qquad \Gamma; e : A \vdash \Delta}{\langle t \| e \rangle : (\Gamma \vdash \Delta)}\ Comp \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta; \mu c : A}\ Activ$$

$$\frac{\Gamma, A \vdash \Delta; t : B}{\Gamma \vdash \Delta; \lambda t : A \to B}\ Abs \qquad \frac{\Gamma \vdash \Delta; t : A \qquad \Gamma; e : B \vdash \Delta}{\Gamma; t \cdot e : A \to B \vdash \Delta}\ App$$

$$\frac{\Gamma \vdash \Delta; t : A \qquad \Gamma \vdash \Delta; u : B}{\Gamma \vdash \Delta; (t,u) : A \wedge B}\ Pair \qquad \frac{\Gamma; e : A \vdash \Delta}{\Gamma; \pi_1 \cdot e : A \wedge B \vdash \Delta}\ Proj_1 \qquad \frac{\Gamma; e : B \vdash \Delta}{\Gamma; \pi_2 \cdot e : A \wedge B \vdash \Delta}\ Proj_2$$

Encore une fois, on peut observer une correspondance directe entre le calcul des séquents LKT et le système de typage LKT^{expl}.

10. Donner une preuve de la loi de Peirce $((A \to B) \to A) \to A$ au choix dans LKT^{expl} ou dans LKT en explicitant par ailleurs le terme associé dans la KAM avec contrôle.

F – $\mathbf{LK}_{tq}$: une symétrisation de LKT

Le calcul LKT est asymétrique, et, de fait, il n'est pas suffisant pour interpréter les programmes du λ-calcul en appel par valeur. On va le symétriser en ajoutant une règle pour mettre le focus sur une formule aussi à gauche.

Les règles étendues de LK_{tq} sont les suivantes :

$$\frac{A \in \Gamma}{\Gamma \vdash \Delta; A}\ Ax_R \qquad \frac{A \in \Delta}{\Gamma; A \vdash \Delta}\ Ax_L$$

$$\frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta; A}\ Focus_R \qquad \frac{\Gamma \vdash \Delta; A \qquad \Gamma; A \vdash \Delta}{\Gamma \vdash \Delta}\ Cut \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma; A \vdash \Delta}\ Focus_L$$

$$\frac{\Gamma \vdash \Delta, A \qquad \Gamma; B \vdash \Delta}{\Gamma \vdash \Delta; A \to B}\ \to_R \qquad \frac{\Gamma \vdash \Delta; A \qquad \Gamma; B \vdash \Delta}{\Gamma; A \to B \vdash \Delta}\ \to_L$$

$$\frac{\Gamma \vdash \Delta; A \qquad \Gamma \vdash \Delta; B}{\Gamma \vdash \Delta; A \wedge B}\ \wedge_R \qquad \frac{\Gamma; A \vdash \Delta}{\Gamma; A \wedge B \vdash \Delta}\ \wedge^1_L \qquad \frac{\Gamma; B \vdash \Delta}{\Gamma; A \wedge B \vdash \Delta}\ \wedge^2_L$$

G – Une symétrisation de la machine abstraite avec contrôle

On peut étendre la machine abstraite précédente avec un constructeur de pile $\tilde{\mu}x.c$ demandant l'évaluation du terme avant de continuer. Le calcul résultant s'appelle le $\mu\tilde{\mu}$-calcul, ou aussi sytème L :

c	$::=$	$\langle t \| e \rangle$	états
t, u	$::=$	$x \mid \mu\alpha.c \mid \lambda x.t \mid (t, u)$	programmes
e, f	$::=$	$\alpha \mid \tilde{\mu}x.c \mid t \cdot e \mid \pi_1[e] \mid \pi_2[e]$	piles

où, intuitivement, le nouveau $\tilde{\mu}x.c$ représente le terme avec un trou « **let** $x = [\,]$ **in** c ».

On peut étendre la machine avec une nouvelle règle permettant de gérer le nouveau constructeur de pile :

$\langle \lambda x.t \| u \cdot e \rangle$	$\to$	$\langle t[x := u] \| e \rangle$
$\langle (t_1, t_2) \| \pi_1 \cdot e \rangle$	$\to$	$\langle t_1 \| e \rangle$
$\langle (t_1, t_2) \| \pi_2 \cdot e \rangle$	$\to$	$\langle t_2 \| e \rangle$
$\langle \mu\alpha.c \| e \rangle$	$\to$	$c[\alpha := e]$
$\langle t \| \tilde{\mu}x.c \rangle$	$\to$	$c[x := t]$

On se retrouve avec une machine non-déterministe puisqu'il y a par exemple deux manières d'évaluer $\langle \mu\alpha.c \| \tilde{\mu}x.c' \rangle$. Cependant, on gagne la possibilité d'interpréter l'évaluation en appel par valeur.

11. Soit la syntaxe suivante du λ-calcul standard avec paires :

$$t, u \quad ::= \quad x \mid \lambda x.t \mid tu \mid (t, u) \mid \pi_1(t) \mid \pi_2(t)$$

Imaginer une compilation du λ-calcul standard avec paires vers le $\mu\tilde{\mu}$-calcul utilisant $\tilde{\mu}$ de telle sorte que la compilation simule la réduction en appel par valeur, c'est-à-dire que la réduction de $(\lambda x.t)u$ en $t[x := u]$ n'ait lieu que quand V est une valeur.