Examen du cours Calcul des Constructions Inductives

B. Barras, H. Herbelin et C. Paulin

24 mars 2003

L'examen dure trois heures, les notes de cours sont autorisées.
Merci de rédiger l'exercice 1 sur une feuille séparée.

1 Modélisation à l'aide de définitions inductives

Le but de cet exercice est d'utiliser des définitions inductives pour modéliser des suites infinies.

On introduit un paramètre de A de type Set et la définition suivante `Stream` qui permet de coder des suites infinies d'objets de type A.

Variable `A` : `Set`.
Inductive `Stream` : `Set` := `mk` : (`C` :`Set`) (`C` → `A`) → (`C` → `C`) → `C` → `Stream`.

Le terme (`mk` $C\ h\ t\ x$) de type `Stream` peut se voir comme une machine possédant un état de type C dont la valeur courante est x et deux méthodes h et t. La méthode h permet de calculer une sortie $(h\ x)$ de type A à partir de l'état courant x tandis que la métode t permet de changer l'état de la machine x en $(t\ x)$. Le terme (`mk` $C\ h\ t\ x$) permet donc de construire une suite infinie d'éléments de A :

$$(h\ x), (h\ (t\ x)), (h\ (t\ (t\ x)))\ldots(h\ (t^k\ x))\ldots$$

En suivant cette intuition, on peut introduire deux fonctions calculant respectivement la tête et le reste d'un objet de type `Stream`.

Definition `hd` : `Stream` → `A` := [s] `Cases s of (mk C h t x)` ⇒ `(h x) end`.

Definition `tl` : `Stream` → `Stream` := [s] `Cases s of (mk C h t x)` ⇒ `(mk C h t (t x)) end`.

1. L'étudiant Gudule remarque que le type `Stream` est un type inductif à un seul constructeur et cherche donc à définir la fonction de projection qui à (`mk` $C\ h\ t\ x$) associe C. Il écrit donc :

   Definition `pi1` : `Stream` → `Set` := [s] `Cases s of (mk C h t x)` ⇒ `C end`.

   Cette définition est-elle acceptée par Coq ? Pourquoi ?
2. Définir la fonction `nth` qui à une stream s et un entier n associe l'élément d'indice n de la suite infinie associée, en supposant que l'on démarre à 0, c'est-à-dire que (`nth` s 0) = (`hd` s).
3. (a) Soit a de type A, donner une expression de type `Stream` dont tous les éléments sont égaux à a.

   (b) On se donne maintenant une fonction f de type `nat` → A. Donner une expression s de type `Stream` dont l'élement d'indice i (ie (`nth` $s\ i$)) est égal à $(f\ i)$.
4. Soit une fonction f de type $A \to A$. Définir une fonction `map` de type `Stream` → `Stream` qui applique la fonction f à tous les élements d'une stream. C'est-à-dire que la stream correspondant à la suite infinie $a_0 \ldots a_n \ldots$ est transformée en une stream associée à la suite $(f\ a_0)\ldots(f\ a_n)\ldots$.
   Justifier informellement mais précisément l'égalité suivante pour toute stream s et entier n :

   $$(\texttt{nth}\ (\texttt{map}\ s)\ n) = (f\ (\texttt{nth}\ s\ n))$$

5. Soit toujours une fonction f de type $A \to A$. On définit une fonction `iter` de type $A \to$ `Stream` telle que (`iter` a) représente la stream formée par $a, (f\ a), \ldots, (f^k\ a), \ldots$

   Definition `iter` : `A` $\to$ `Stream` := [a](`mk A` [x]`x f a`).

   On souhaite montrer la proprieté suivante appelée P :

   (`a :A`)(`iter (f a)`) = (`map (iter a)`)

   Les deux termes (`iter` $(f\ a)$) et (`map` (`iter` a)) sont-il convertibles ? Montrer que (`hd` (`iter` $(f\ a)$)) et (`hd` (`map` (`iter` a))) sont convertibles.

6. L'étudiant Gudule se dit que l'égalité de Leibniz n'est pas la bonne notion et se propose donc de définir une égalité spécifique pour les streams de manière inductive, il introduit donc :

   Inductive `eqS` : `Stream` $\to$ `Stream` $\to$ `Prop` :=
   `eqS_intro` : (`s1`,`s2` :`Stream`)(`eqS` (`tl s1`) (`tl s2`))$\to$(`hd s1`)=(`hd s2`)$\to$(`eqS s1 s2`).

   (a) Est-il possible de montrer la propriété P en prenant pour relation d'égalité la relation `eqS` ?

   (b) Est-il possible de montrer la réflexivité de `eqS` : (s : `Stream`)(`eqS` s s) ?

   (c) Proposer une autre notion d'égalité qui permette de démontrer la propriété P.

2 Extraction, récursion non structurelle, types singletons

A) On considère la réalisation des points-fixes bien fondés du Calcul des Constructions Inductives (CCI). Pour cela, on considère un langage de réalisateurs qui contient un opérateur de point-fixe.

On se place dans un contexte déclarant un type A : Set et une relation $R : A \to A \to$ Prop notée avec le symbole infixe <. On définit l'accessibilité d'un élément a de A suivant la relation R (l'accessibilité de a exprime que toute suite décroissante selon R à partir de a est finie)

```
Inductive Acc : A->Prop :=
  Acc_intro : (x:A)((y:A)(y<x)->(Acc y))->(Acc x)
```

On se propose de montrer que le schéma suivant d'élimination récursive non dépendante Acc_{rec} de Acc vers Set

$$\forall P : (A \to \mathsf{Set}).\ (\forall x : A.\ (\forall y : A.\ y < x \to P(y)) \to P(x)) \to \forall a : A.\ Acc(a) \to P(a)$$

est dérivable et réalisable par un opérateur de point-fixe.

1. Montrer par analyse de cas vers Prop que $\forall a : A.\ Acc(a) \to \forall b : A.\ b < a \to Acc(b)$.
2. En déduire que Acc_{rec} est prouvable par point-fixe bien fondé. En donner la preuve sous la forme d'un terme Coq. Quel est l'argument de décroissance ?
3. On considère l'extraction présentée en cours (effacement des propositions et dépendances de terme). Quel est le type extrait de l'énoncé de Acc_{rec} ?
4. Si A : Set, on définit l'extraction d'un point-fixe bien fondé $Fix\{f/n : A := M\}$ comme suit (n est l'indice de l'argument décroissant de f)

$$\mathcal{E}(Fix\{f/n : A := M\}) = Fix\{f : \mathcal{E}(A) := \mathcal{E}(M)\}$$

   En particulier, si l'argument de décroissance est dans Prop, le terme extrait n'est plus forcément intrinsèquement bien fondé.

   Quelle est l'extraction de la preuve de Acc_{rec} obtenue plus haut ?

B) On étudie maintenant l'élimination vers Set du type inductif

```
Inductive and [A,B:Prop] : Prop :=  conj : A->B->A/\B
```

Ce type est un type singleton pour lequel l'élimination est autorisée vers Set dans le CCI.

1. Montrer qu'en fait l'élimination non dépendante vers Set
$$\forall A, B : \mathsf{Prop}.\ \forall P : \mathsf{Set}.\ (A \to B \to P) \to A \wedge B \to P$$
est dérivable à partir de l'élimination vers Prop.
2. Montrer qu'à partir du schéma d'élimination dépendante de $\wedge$ vers Prop, on peut montrer la surjectivité du constructeur de conjonction
$$\forall p : A \wedge B.\ (\pi_1(p), \pi_2(p)) = p$$
où (a, b) dénote $(\texttt{conj}\ a\ b)$ et π_1 et π_2 les projections de type $A \wedge B \to A$ et $A \wedge B \to B$.
3. On suppose donné le schéma d'élimination non dépendante eq_{rec} vers Set de l'égalité de Leibniz sur Prop
$$\forall A : \mathsf{Prop}.\ \forall P : A \to \mathsf{Set}.\ \forall x : A.\ P(x) \to x = y \to P(y)$$
Montrer que l'élimination dépendante de $\wedge$ vers Set
$$\forall A, B : \mathsf{Prop}.\ \forall P : (A \wedge B) \to \mathsf{Set}.\ (\forall a : A.\ \forall b : B.\ P((a, b))) \to \forall p : A \wedge B.\ P(p)$$
est dérivable à partir de l'élimination dépendante de $\wedge$ vers Prop et eq_{rec}.

3 Paradoxes

L'objet de cet exercice est de montrer plusieurs paradoxes dans des extensions de CCI. Les preuves demandées devront être précises même s'il n'est pas nécessaire de détailler complètement les termes de CCI. On prendra soin de préciser lors des éliminations, quel schéma est utilisé (par exemple : élimination dépendante de la disjonction sur la sorte Prop).

Dans la suite $A \leftrightarrow B$ dénote l'équivalence dans Prop, quelques soient les sortes de A et B. Pareillement, $\neg A$ dénote la négation dans Prop quelque soit la sorte de A et $\Sigma a : A.P$ (respectivement $\exists a : A.P$) dénote l'existentielle dans Set (respectivement dans Prop) quelques soient les sortes de A et P. Enfin, $\emptyset$ dénote l'ensemble vide dans Set.

Un **petit type** est un type de Set ou Prop. On appelle **plongement de Prop vers un petit type** la donnée de B, p et r vérifiant les propriétés

- B est un petit type
- $p : \mathsf{Prop} \to B$
- $r : B \to \mathsf{Prop}$
- $\forall A : \mathsf{Prop}.\ (r\ (p\ A)) \leftrightarrow A$ est prouvable

Pareillement, on obtient un **plongement de Set vers un petit type** en remplaçant Prop par Set dans la définition précédente.

On peut montrer que l'existence d'un plongement de Prop (ou de Set) dans un petit type rend le calcul des constructions (CC) incohérent car alors on peut encoder dans le CC le système U^- qui est incohérent[1]

A) On considère le type inductif large

[1] Le système U^- est une logique avec quantification sur un langage qui contient les termes du système F et le type des propositions Prop. Formellement, on a $\mathsf{Prop} : \mathsf{Type}_1 : \mathsf{Type}_2$ et on peut construire les produits

$(\mathsf{Prop}, \mathsf{Prop}, \mathsf{Prop})$	implication
$(\mathsf{Type}_1, Prop, Prop)$	quantification
$(\mathsf{Type}_1, \mathsf{Type}_1, \mathsf{Type}_1)$	fonctions (et arités)
$(\mathsf{Type}_2, \mathsf{Type}_1, \mathsf{Type}_1)$	fonctions (et arités) imprédicatives

Un plongement de Prop vers un petit type permet d'encoder le produit $(\mathsf{Type}_2, \mathsf{Type}_1, \mathsf{Type}_1)$ via le produit prédicatif $(\mathsf{Type}_1, \mathsf{Type}_1, \mathsf{Type}_1)$ qui lui est admis dans le CC, d'où la possibilité de traduire les paradoxes de U^- dans le CC.
Alternativement, Prop peut être remplacé par Set et on obtient le même résultat pour les plongements de Set vers un petit type.

```
Inductive S : Set := I : Set -> S.
```

1. Quels sont les schémas d'élimination autorisés pour ce type ?
2. Montrer que si on autorisait l'élimination forte à partir d'un type large, on pourrait exhiber un plongement des propositions vers un petit type.

B) On veut montrer l'incohérence de la logique classique dans Set.

1. Si B : Set et P : Set $\to B \to$ Prop, montrer que l'axiome du choix suivant

$$(\forall A : \mathsf{Set}.\ \Sigma b : B.(P\ A\ b)) \to (\Sigma f : \mathsf{Set} \to B. \forall A : \mathsf{Set}.\ (P\ A\ b))$$

   est dérivable dans le CCI. Quel est le genre de schéma d'élimination utilisé ?
2. On peut montrer $(true = false) \to \emptyset$ dans le CCI. Quel est le genre de schéma d'élimination utilisé pour discriminer ainsi entre $true$ et $false$?
3. Montrer que la logique classique dans Set (càd $\forall A : \mathsf{Set}.\ A + (A \to \emptyset)$) entraîne la propriété

$$\forall A : \mathsf{Set}.\ \Sigma b : bool. b = true \leftrightarrow A.$$

   A-t-on eu besoin de la discrimination entre $true$ et $false$?
4. Montrer $\Sigma p : \mathsf{Set} \to bool. \forall A : \mathsf{Set}.\ p(A) = true \leftrightarrow A$.
5. Exhiber alors un plongement des propositions vers un petit type. En déduire l'incohérence de la logique classique dans Set pour le CCI.

C) On reprend l'étude précédente en remplaçant certaines occurrences de Set par Prop.

1. Si on prend comme axiome classique $\forall A : \mathsf{Prop}.\ \{A\} + \{\neg A\}$, pourra-t-on toujours dériver

$$(\forall A : \mathsf{Prop}.\ \Sigma b : B.(P\ A\ b)) \to (\Sigma f : \mathsf{Prop} \to B. \forall A : \mathsf{Prop}.\ (P\ A\ b))\ ?$$

   L'axiome $\forall A : \mathsf{Prop}.\ \{A\} + \{\neg A\}$ est-il alors cohérent dans le CCI ?
2. On définit $bool$ dans Prop.

```
Inductive bool_P : Prop := true_P : bool_P | false_P : bool_P.
```

   Peut-on montrer $\neg true_P = false_P$ de la même manière que $true = false \to \emptyset$ était prouvable ?
3. Montrer que $(\forall A : \mathsf{Prop}.\ \exists b : B.(P\ A\ b)) \to (\exists f : \mathsf{Prop} \to B. \forall A : \mathsf{Prop}.\ (P\ A\ b))$ reste prouvable avec B : Prop et l'existentielle dans Prop.
4. En déduire que la logique classique dans Prop (càd $\forall A : \mathsf{Prop}.\ A \vee \neg A$) entraîne $\neg\neg true_P = false_P$ et donc $true_P = false_P$.
5. Quel schéma d'élimination permet alors de conclure à l'indiscernabilité des preuves

$$\forall A : \mathsf{Prop}.\ \forall a_1, a_2 : A.\ a_1 = a_2$$

.