

Un codage de l'élimination dépendante

Sylvain Lebresne

Motivation

Exprimer les schémas d'élimination **dépendant** des définitions inductives à partir de leurs schémas d'élimination **non dépendant** et des Σ -types.

Intuition

Idée : Ne peut-on pas “plonger les dépendances” dans un couple dépendant (qui n’est pas dépendant, lui), utiliser le schéma d’élimination non dépendant “pour préserver les réductions”, puis “restituer les dépendances à la sortie” ?

Présentation du système

La syntaxe

Règles de typage

Règles de réduction

Le codage

Principe

Le cas des booléens

Le cas des naturels

Le cas de l'égalité

Les autres types de données inductifs

Conclusions

La syntaxe du système

$$\begin{aligned}
 u, v, A, B:: &= x \mid c \mid \textit{Set} \mid \textit{Type} \mid \textit{fix}_f(u) \\
 &\mid \Pi x:A. B \mid \lambda x:A. v \mid u v \\
 &\mid \Sigma x:A. B \mid \langle u, v \rangle \mid \pi_1(u) \mid \pi_2(u)
 \end{aligned}$$

Remarque :

- ▶ Dans un souci d'amélioration de la clarté de cette exposé, nous utiliserons les lettres majuscules A, B, \dots pour ce qui intuitivement un type, et les minuscules u, v, \dots pour ce qui est intuitivement un terme.
- ▶ De plus, nous utiliserons le raccourci $A \rightarrow B$ pour $\Pi x:A. B$ lorsque $x \notin FV(B)$.
- ▶ Nous utiliserons aussi $\forall x:A. B$ parfois (souvent), à la place de $\Pi x:A. B$.

Jugements

- ▶ Les contextes sont des listes ordonnées de paire $(x:A)$ où x est un nom de variable et A un terme.
- ▶ On définit 2 jugements :
 - ▶ “ $\Gamma \vdash$ ” qui dit que “le contexte Γ est bien formé”.
 - ▶ “ $\Gamma \vdash u : A$ ” qui dit que “dans le contexte Γ , u est de type A ”.

Les règles de typages (1/2)

$$\frac{}{\Gamma \vdash \perp_{wf}}$$

$$\frac{\Gamma \vdash A : s \quad x \notin DV(\Gamma)}{\Gamma, (x:A) \vdash} \text{var}_{wf}$$

$$\frac{\Gamma \vdash (x:T) \in \Gamma}{\Gamma \vdash x : A} \text{var}$$

$$\frac{\Gamma \vdash \Pi x:A. B : s \quad \Gamma, (x:A) \vdash u : B}{\Gamma \vdash \lambda x:A. u : \Pi x:A. B} \text{lamb}$$

$$\frac{\Gamma \vdash u : \Pi x:A. B \quad \Gamma \vdash v : A}{\Gamma \vdash u v : B\{x := v\}} \text{app}$$

$$\frac{\Gamma \vdash A : s1 \quad \Gamma, (x:A) \vdash B : s2}{\Gamma \vdash \Pi x:A. B : \max(s1, s2)} \text{prod}$$

$$\frac{\Gamma \vdash u : T \quad \Gamma \vdash T' : s \quad T = T'}{\Gamma \vdash u : T'} \text{conv}$$

Les règles de typages (2/2)

$$\frac{\Gamma \vdash u : A \quad \Gamma \vdash v : B\{x := u\}}{\Gamma \vdash \langle u, v \rangle : \Sigma x:A. B} \textit{pair} \qquad \frac{\Gamma \vdash A : s1 \quad \Gamma, (x:A) \vdash B : s2}{\Gamma \vdash \Sigma x:A. B : \max(s1, s2)} \textit{sig}$$

$$\frac{\Gamma \vdash u : \Sigma x:A. B}{\Gamma \vdash \pi_1(u) : A} \textit{proj}_1 \qquad \frac{\Gamma \vdash u : \Sigma x:A. B}{\Gamma \vdash \pi_2(u) : B\{x := \pi_1(u)\}} \textit{proj}_2$$

$$\frac{\Gamma, (f:X) \vdash t : X}{\Gamma \vdash \textit{fix}_f(t) : X} \textit{fix}$$

Les règles de réduction

$$(\lambda x:A. t) u \triangleright t\{x := u\}$$

$$\lambda x:A. (M x) \triangleright M \text{ (si } x \notin FV(M)\text{)}$$

$$fix_f(t) \triangleright t\{f := fix_f(t)\}$$

$$\pi_1 \langle u, v \rangle \triangleright u$$

$$\pi_2 \langle u, v \rangle \triangleright v$$

Le Codage

Grossièrement

$$Ind^{ndp} \quad P \quad f_1 \quad \dots \quad f_n \quad i : P$$

$$Ind^{dep} \quad P(x) \quad f'_1 \quad \dots \quad f'_n \quad i : P(i)$$

Les booléens et leur schéma d'élimination non dépendant

On ajoute à notre système les constantes suivantes :

Les booléens et leur schéma d'élimination non dépendant

On ajoute à notre système les constantes suivantes :

▶ *bool* : *Set*

Les booléens et leur schéma d'élimination non dépendant

On ajoute à notre système les constantes suivantes :

- ▶ *bool* : *Set*
- ▶ *true* : *bool*
- ▶ *false* : *bool*

Les booléens et leur schéma d'élimination non dépendant

On ajoute à notre système les constantes suivantes :

- ▶ $bool : Set$
- ▶ $true : bool$
- ▶ $false : bool$
- ▶ $bool^{ndp} : \forall P:Set, P \rightarrow P \rightarrow bool \rightarrow P$

Les booléens et leur schéma d'élimination non dépendant

On ajoute à notre système les constantes suivantes :

- ▶ $bool : Set$
- ▶ $true : bool$
- ▶ $false : bool$
- ▶ $bool^{ndp} : \forall P:Set, P \rightarrow P \rightarrow bool \rightarrow P$

Ainsi que les règles de réduction du schéma d'élimination non dépendant :

$$\begin{array}{l}
 bool^{ndp} P x y true \triangleright x \\
 bool^{ndp} P x y false \triangleright y
 \end{array}$$

Ce que l'on veut !

On souhaite maintenant définir un terme $bool^{dep}$ de type

$$\forall P:(bool \rightarrow Set), (P \ true) \rightarrow (P \ false) \rightarrow \forall b:bool, P \ b$$

et ayant les mêmes réductions que $bool^{ndp}$.

Le schéma d'élimination dépendant

On définit ce schéma d'élimination dépendant par :

$$bool^{dep} = \lambda P:bool \rightarrow Set. \lambda x:(P \text{ true}). \lambda y:(P \text{ false}). \lambda b:bool. \pi_2(M_{P,x,y,b})$$

Le schéma d'élimination dépendant

On définit ce schéma d'élimination dépendant par :

$$bool^{dep} = \lambda P:bool \rightarrow Set. \lambda x:(P \text{ true}). \lambda y:(P \text{ false}). \lambda b:bool. \pi_2(M_{P,x,y,b})$$

où

$$M_{P,x,y,b} = bool^{ndp} (\sum z:bool. P z) \langle true, x \rangle \langle false, y \rangle b$$

Mais ...

Malheureusement, un tel $bool^{dep}$ à le type :

$$\forall P:bool \rightarrow Set, (P \text{ true}) \rightarrow (P \text{ false}) \rightarrow \forall b:bool, P(\pi_1(M_{P,x,y,b}))$$

Mais ...

Malheureusement, un tel $bool^{dep}$ à le type :

$$\forall P:bool \rightarrow Set, (P \text{ true}) \rightarrow (P \text{ false}) \rightarrow \forall b:bool, P(\pi_1(M_{P,x,y,b}))$$

or, comme $\pi_1(M_{P,x,y,b}) \neq b$ ($\pi_1(bool^{ndp} \dots)$ est “bloqué”), ce n’est pas le type attendu.

Re: Mais ...

Donc, on ajoute deux nouvelles règles de réduction :

Re: Mais ...

Donc, on ajoute deux nouvelles règles de réduction :

- ▶ la première est une règle de commutation :

$$\pi_1(\text{bool}^{ndp} (\Sigma z:A. P) x y b) \triangleright \text{bool}^{ndp} A (\pi_1(x)) (\pi_1(y)) b$$

Re: Mais ...

Donc, on ajoute deux nouvelles règles de réduction :

- ▶ la première est une règle de commutation :

$$\pi_1(\mathit{bool}^{ndp} (\Sigma z:A. P) x y b) \triangleright \mathit{bool}^{ndp} A (\pi_1(x)) (\pi_1(y)) b$$

- ▶ la seconde est une règle η :

$$\mathit{bool}^{ndp} \mathit{bool} \mathit{true} \mathit{false} b \triangleright b$$

Ouf !

Avec ces deux règles, on est sauvé :

$$\pi_1(M_{P,x,y,b}) = \pi_1(\text{bool}^{ndp} (\Sigma z:\text{bool}. P z) \langle \text{true}, x \rangle \langle \text{false}, y \rangle b)$$

Ouf !

Avec ces deux règles, on est sauvé :

$$\begin{aligned} \pi_1(M_{P,x,y,b}) &= \pi_1(\text{bool}^{ndp} (\Sigma z:\text{bool}. P z) \langle \text{true}, x \rangle \langle \text{false}, y \rangle b) \\ &\triangleright \text{bool}^{ndp} \text{ bool } (\pi_1 \langle \text{true}, x \rangle) (\pi_1 \langle \text{false}, y \rangle) b \end{aligned}$$

Ouf !

Avec ces deux règles, on est sauvé :

$$\begin{aligned} \pi_1(M_{P,x,y,b}) &= \pi_1(\text{bool}^{ndp} (\Sigma z:\text{bool}. P z) \langle \text{true}, x \rangle \langle \text{false}, y \rangle b) \\ &\triangleright \text{bool}^{ndp} \text{ bool } (\pi_1 \langle \text{true}, x \rangle) (\pi_1 \langle \text{false}, y \rangle) b \\ &\triangleright \text{bool}^{ndp} \text{ bool } \text{ true } \text{ false } b \end{aligned}$$

Ouf !

Avec ces deux règles, on est sauvé :

$$\begin{aligned}
 \pi_1(M_{P,x,y,b}) &= \pi_1(\text{bool}^{ndp} (\Sigma z:\text{bool}. P z) \langle \text{true}, x \rangle \langle \text{false}, y \rangle b) \\
 &\triangleright \text{bool}^{ndp} \text{ bool } (\pi_1 \langle \text{true}, x \rangle) (\pi_1 \langle \text{false}, y \rangle) b \\
 &\triangleright \text{bool}^{ndp} \text{ bool } \text{ true } \text{ false } b \\
 &\triangleright b
 \end{aligned}$$

Ouf !

Avec ces deux règles, on est sauvé :

$$\begin{aligned}
 \pi_1(M_{P,x,y,b}) &= \pi_1(\text{bool}^{ndp} (\sum z:\text{bool}. P z) \langle \text{true}, x \rangle \langle \text{false}, y \rangle b) \\
 &\triangleright \text{bool}^{ndp} \text{ bool } (\pi_1 \langle \text{true}, x \rangle) (\pi_1 \langle \text{false}, y \rangle) b \\
 &\triangleright \text{bool}^{ndp} \text{ bool } \text{ true } \text{ false } b \\
 &\triangleright b
 \end{aligned}$$

Et on vérifie aisément que bool^{dep} à les mêmes réductions que bool^{ndp} .

Ouf !

Avec ces deux règles, on est sauvé :

$$\begin{aligned}
 \pi_1(M_{P,x,y,b}) &= \pi_1(\text{bool}^{ndp} (\sum z:\text{bool}. P z) \langle \text{true}, x \rangle \langle \text{false}, y \rangle b) \\
 &\triangleright \text{bool}^{ndp} \text{ bool } (\pi_1 \langle \text{true}, x \rangle) (\pi_1 \langle \text{false}, y \rangle) b \\
 &\triangleright \text{bool}^{ndp} \text{ bool } \text{ true } \text{ false } b \\
 &\triangleright b
 \end{aligned}$$

Et on vérifie aisément que bool^{dep} à les mêmes réductions que bool^{ndp} :

$$\text{bool}^{dep} P x y \text{ true} = \pi_2(\text{bool}^{ndp} (\sum z:\text{bool}. P z) \langle \text{true}, x \rangle \langle \text{false}, y \rangle \text{ true})$$

Ouf !

Avec ces deux règles, on est sauvé :

$$\begin{aligned} \pi_1(M_{P,x,y,b}) &= \pi_1(\text{bool}^{ndp} (\Sigma z:\text{bool}. P z) \langle \text{true}, x \rangle \langle \text{false}, y \rangle b) \\ &\triangleright \text{bool}^{ndp} \text{ bool } (\pi_1 \langle \text{true}, x \rangle) (\pi_1 \langle \text{false}, y \rangle) b \\ &\triangleright \text{bool}^{ndp} \text{ bool } \text{true} \text{ false } b \\ &\triangleright b \end{aligned}$$

Et on vérifie aisément que bool^{dep} à les mêmes réductions que bool^{ndp} :

$$\begin{aligned} \text{bool}^{dep} P x y \text{true} &= \pi_2(\text{bool}^{ndp} (\Sigma z:\text{bool}. P z) \langle \text{true}, x \rangle \langle \text{false}, y \rangle \text{true}) \\ &\triangleright \pi_2 \langle \text{true}, x \rangle \end{aligned}$$

Ouf !

Avec ces deux règles, on est sauvé :

$$\begin{aligned} \pi_1(M_{P,x,y,b}) &= \pi_1(\text{bool}^{ndp} (\Sigma z:\text{bool}. P z) \langle \text{true}, x \rangle \langle \text{false}, y \rangle b) \\ &\triangleright \text{bool}^{ndp} \text{ bool } (\pi_1 \langle \text{true}, x \rangle) (\pi_1 \langle \text{false}, y \rangle) b \\ &\triangleright \text{bool}^{ndp} \text{ bool } \text{true } \text{false } b \\ &\triangleright b \end{aligned}$$

Et on vérifie aisément que bool^{dep} à les mêmes réductions que bool^{ndp} :

$$\begin{aligned} \text{bool}^{dep} P x y \text{true} &= \pi_2(\text{bool}^{ndp} (\Sigma z:\text{bool}. P z) \langle \text{true}, x \rangle \langle \text{false}, y \rangle \text{true}) \\ &\triangleright \pi_2 \langle \text{true}, x \rangle \\ &\triangleright x \end{aligned}$$

Un premier bilan

- ▶ On sait maintenant coder le schéma d'élimination dépendant de *bool*.

Un premier bilan

- ▶ On sait maintenant coder le schéma d'élimination dépendant de *bool*.
- ▶ Ce codage est “naturel” et assez simple.

Un premier bilan

- ▶ On sait maintenant coder le schéma d'élimination dépendant de *bool*.
- ▶ Ce codage est “naturel” et assez simple.
- ▶ Il nécessite l'addition de deux règles de réduction “raisonnables”.

Un premier bilan

- ▶ On sait maintenant coder le schéma d'élimination dépendant de *bool*.
- ▶ Ce codage est “naturel” et assez simple.
- ▶ Il nécessite l'addition de deux règles de réduction “raisonnables”.

Ensuite ? On va maintenant essayer de faire exactement la même chose pour des inductifs plus compliqués. Et notamment, quid des inductifs “inductifs” ? Essayons donc avec nos amis les entiers naturel.

Les entiers naturels et leur éliminateur non-dépendant

On ajoute d'abord les constantes suivantes :

Les entiers naturels et leur éliminateur non-dépendant

On ajoute d'abord les constantes suivantes :

▶ *nat* : *Set*

Les entiers naturels et leur éliminateur non-dépendant

On ajoute d'abord les constantes suivantes :

- ▶ $nat : Set$
- ▶ $0 : nat$
- ▶ $S : nat \rightarrow nat$

Les entiers naturels et leur éliminateur non-dépendant

On ajoute d'abord les constantes suivantes :

- ▶ $nat : Set$
- ▶ $0 : nat$
- ▶ $S : nat \rightarrow nat$
- ▶ $nat_{rec}^{ndp} : \forall P:Set, P \rightarrow (nat \rightarrow P \rightarrow P) \rightarrow nat \rightarrow P$

Les entiers naturels et leur éliminateur non-dépendant

On ajoute d'abord les constantes suivantes :

- ▶ $nat : Set$
- ▶ $0 : nat$
- ▶ $S : nat \rightarrow nat$
- ▶ $nat_{rec}^{ndp} : \forall P:Set, P \rightarrow (nat \rightarrow P \rightarrow P) \rightarrow nat \rightarrow P$

Ainsi que les règles de réduction du schéma d'élimination non dépendant :

$$nat_{rec}^{ndp} P \times f 0 \triangleright x$$

$$nat_{rec}^{ndp} P \times f (S n) \triangleright f n (nat_{rec}^{ndp} P \times f n)$$

Puis on veut ...

... définir un terme nat_{rec}^{dep} de type

$$\forall P: nat \rightarrow Set, (P\ 0) \rightarrow (\forall n: nat, P\ n \rightarrow P\ (S\ n)) \rightarrow (\forall n: nat, P\ n)$$

et ayant les mêmes réductions que nat_{rec}^{ndp} .

Puis on veut ...

... définir un terme nat_{rec}^{dep} de type

$$\forall P: nat \rightarrow Set, (P\ 0) \rightarrow (\forall n: nat, P\ n \rightarrow P\ (S\ n)) \rightarrow (\forall n: nat, P\ n)$$

et ayant les mêmes réductions que nat_{rec}^{ndp} .

Essayons, pour de rire, de suivre exactement le même cheminement que pour les booléens.

Essai ...

On code donc le schéma d'élimination dépendant par :

$$nat_{rec}^{dep} = \lambda P:nat \rightarrow Set. \lambda x:(P\ 0). \lambda f:(\forall n:nat, P\ n \rightarrow P\ (Sn)). \lambda n:nat. \\ \pi_2(M_{P,x,f,n})$$

Essai ...

On code donc le schéma d'élimination dépendant par :

$$\text{nat}_{rec}^{dep} = \lambda P:\text{nat} \rightarrow \text{Set}. \lambda x:(P\ 0). \lambda f:(\forall n:\text{nat}, P\ n \rightarrow P\ (S\ n)). \lambda n:\text{nat}. \\ \pi_2(M_{P,x,f,n})$$

où

$$M_{P,x,f,n} = \text{nat}_{rec}^{ndp} (\Sigma z:\text{nat}. P\ z) \langle 0, x \rangle \\ (\lambda m:\text{nat}. \lambda c:(\Sigma z:\text{nat}. P\ z). \langle S\ \pi_1(c), f\ \pi_1(c)\ \pi_2(c) \rangle) n$$

... malheureusement pas transformé

Le type de $\pi_2(M_{P,x,f,n})$ est $P(\pi_1(M_{P,x,f,n}))$ et donc, on est confronté au même problème que pour les booléens et on se doit d'introduire de nouvelles règles de réduction.

La première règle est une règle de commutation entre π_1 et nat_{rec}^{ndp} .

... malheureusement pas transformé

Le type de $\pi_2(M_{P,x,f,n})$ est $P(\pi_1(M_{P,x,f,n}))$ et donc, on est confronté au même problème que pour les booléens et on se doit d'introduire de nouvelles règles de réduction.

La première règle est une règle de commutation entre π_1 et nat_{rec}^{ndp} . Mais cette règle devrait réécrire

$$\pi_1(\text{nat}_{rec}^{ndp} (\sum z:\text{nat}. P z) \langle 0, x \rangle) \\ (\lambda m:\text{nat}. \lambda c:(\sum z:\text{nat}. P z). \langle S \pi_1(c), f \pi_1(c) \pi_2(c) \rangle) n$$

... malheureusement pas transformé

Le type de $\pi_2(M_{P,x,f,n})$ est $P(\pi_1(M_{P,x,f,n}))$ et donc, on est confronté au même problème que pour les booléens et on se doit d'introduire de nouvelles règles de réduction.

La première règle est une règle de commutation entre π_1 et nat_{rec}^{ndp} .
Mais cette règle devrait réécrire

$$\pi_1(nat_{rec}^{ndp} (\sum z:nat. P z) \langle 0, x \rangle) \\ (\lambda m:nat. \lambda c:(\sum z:nat. P z). \langle S \pi_1(c), f \pi_1(c) \pi_2(c) \rangle) n$$

en

$$nat_{rec}^{ndp} nat 0 (\lambda m:nat. \lambda c:nat. (S c)) n$$

de façon à ensuite utiliser la règle η de nat_{rec}^{ndp} pour conclure.

... malheureusement pas transformé

Le type de $\pi_2(M_{P,x,f,n})$ est $P(\pi_1(M_{P,x,f,n}))$ et donc, on est confronté au même problème que pour les booléens et on se doit d'introduire de nouvelles règles de réduction.

La première règle est une règle de commutation entre π_1 et nat_{rec}^{ndp} . Mais cette règle devrait réécrire

$$\pi_1(\text{nat}_{rec}^{ndp} (\sum z:\text{nat}. P z) \langle 0, x \rangle (\lambda m:\text{nat}. \lambda c:(\sum z:\text{nat}. P z). \langle S \pi_1(c), f \pi_1(c) \pi_2(c) \rangle) n)$$

en

$$\text{nat}_{rec}^{ndp} \text{ nat } 0 (\lambda m:\text{nat}. \lambda c:\text{nat}. (S c)) n$$

de façon à ensuite utiliser la règle η de nat_{rec}^{ndp} pour conclure.

Est-ce vraiment sérieux ?

Une autre idée

- ▶ La récursivité due au constructeur S semble poser problème.

Une autre idée

- ▶ La récursivité due au constructeur S semble poser problème.
- ▶ Mais on peut utiliser le schéma d'analyse par cas non dépendant afin de définir le schéma d'analyse par cas dépendant.

Une autre idée

- ▶ La récursivité due au constructeur S semble poser problème.
- ▶ Mais on peut utiliser le schéma d'analyse par cas non dépendant afin de définir le schéma d'analyse par cas dépendant.
- ▶ Puis définir le schéma d'élimination dépendant général par point fixe.

Implémentation (1/3)

On se donne (ou on définit à l'aide de nat_{rec}^{ndp}) le schéma d'analyse par cas non dépendant :

$$nat_{case}^{ndp} : \forall P : Set, P \rightarrow (nat \rightarrow P) \rightarrow nat \rightarrow P$$

ayant les règles de réduction :

$$\begin{aligned} nat_{case}^{ndp} P x f 0 &\triangleright x \\ nat_{case}^{ndp} P x f (S n) &\triangleright f n \end{aligned}$$

Implémentation (1/3)

On se donne (ou on définit à l'aide de nat_{rec}^{ndp}) le schéma d'analyse par cas non dépendant :

$$nat_{case}^{ndp} : \forall P : Set, P \rightarrow (nat \rightarrow P) \rightarrow nat \rightarrow P$$

ayant les règles de réduction :

$$\begin{aligned} nat_{case}^{ndp} P x f 0 &\triangleright x \\ nat_{case}^{ndp} P x f (S n) &\triangleright f n \end{aligned}$$

Auxquelles on ajoute les deux règles de réduction suivantes :

$$\pi_1(nat_{case}^{ndp} (\Sigma z:A. P) x (\lambda m:nat. y) n) \triangleright nat_{case}^{ndp} A (\pi_1(x)) (\lambda m:nat. \pi_1(y)) n$$

(Commutation avec π_1)

Implémentation (1/3)

On se donne (ou on définit à l'aide de nat_{rec}^{ndp}) le schéma d'analyse par cas non dépendant :

$$nat_{case}^{ndp} : \forall P : Set, P \rightarrow (nat \rightarrow P) \rightarrow nat \rightarrow P$$

ayant les règles de réduction :

$$\begin{aligned} nat_{case}^{ndp} P x f 0 &\triangleright x \\ nat_{case}^{ndp} P x f (S n) &\triangleright f n \end{aligned}$$

Auxquelles on ajoute les deux règles de réduction suivantes :

$$\begin{aligned} \pi_1(nat_{case}^{ndp} (\Sigma z:A. P) x (\lambda m:nat. y) n) &\triangleright nat_{case}^{ndp} A (\pi_1(x)) (\lambda m:nat. \pi_1(y)) n \\ nat_{case}^{ndp} nat 0 (S) n &\triangleright n \end{aligned}$$

(Commutation avec π_1 et règle η)

Implémentation (2/3)

On définit le schéma d'analyse par cas dépendant par :

$$\mathit{nat}_{\mathit{case}}^{\mathit{dep}} : \forall P : \mathit{nat} \rightarrow \mathit{Set}, (P\ 0) \rightarrow (\forall n : \mathit{nat}, P\ (S\ n)) \rightarrow (\forall n : \mathit{nat}, P\ n)$$

Implémentation (2/3)

On définit le schéma d'analyse par cas dépendant par :

$$\begin{aligned}
 \mathit{nat}_{\mathit{case}}^{\mathit{dep}} : & \quad \forall P:\mathit{nat} \rightarrow \mathit{Set}, (P\ 0) \rightarrow (\forall n:\mathit{nat}, P\ (S\ n)) \rightarrow (\forall n:\mathit{nat}, P\ n) \\
 = & \quad \lambda P:\mathit{nat} \rightarrow \mathit{Set}. \\
 & \quad \lambda x:(P\ 0). \lambda f:(\forall n:\mathit{nat}, P\ (S\ n)). \\
 & \quad \lambda n:\mathit{nat}. \pi_2(M_{P,x,f,n})
 \end{aligned}$$

où

$$M_{P,x,f,n} = \mathit{nat}_{\mathit{case}}^{\mathit{ndp}} (\Sigma z:\mathit{nat}. P\ z) \langle 0, x \rangle (\lambda m:\mathit{nat}. \langle S\ m, f\ m \rangle) n$$

Implémentation (2/3)

On définit le schéma d'analyse par cas dépendant par :

$$\begin{aligned}
 \mathit{nat}_{\mathit{case}}^{\mathit{dep}} &: \forall P:\mathit{nat} \rightarrow \mathit{Set}, (P\ 0) \rightarrow (\forall n:\mathit{nat}, P\ (S\ n)) \rightarrow (\forall n:\mathit{nat}, P\ n) \\
 &= \lambda P:\mathit{nat} \rightarrow \mathit{Set}. \\
 &\quad \lambda x:(P\ 0). \lambda f:(\forall n:\mathit{nat}, P\ (S\ n)). \\
 &\quad \lambda n:\mathit{nat}. \pi_2(M_{P,x,f,n}) \quad : P\ (\pi_1(M_{P,x,f,n}))
 \end{aligned}$$

où

$$M_{P,x,f,n} = \mathit{nat}_{\mathit{case}}^{\mathit{ndp}} (\Sigma z:\mathit{nat}. P\ z) \langle 0, x \rangle (\lambda m:\mathit{nat}. \langle S\ m, f\ m \rangle) n$$

Implémentation (2/3)

On définit le schéma d'analyse par cas dépendant par :

$$\begin{aligned}
 \mathit{nat}_{\mathit{case}}^{\mathit{dep}} &: \forall P:\mathit{nat} \rightarrow \mathit{Set}, (P\ 0) \rightarrow (\forall n:\mathit{nat}, P\ (S\ n)) \rightarrow (\forall n:\mathit{nat}, P\ n) \\
 &= \lambda P:\mathit{nat} \rightarrow \mathit{Set}. \\
 &\quad \lambda x:(P\ 0). \lambda f:(\forall n:\mathit{nat}, P\ (S\ n)). \\
 &\quad \lambda n:\mathit{nat}. \pi_2(M_{P,x,f,n}) \quad : P\ (\pi_1(M_{P,x,f,n}))
 \end{aligned}$$

où

$$\pi_1(M_{P,x,f,n}) = \pi_1(\mathit{nat}_{\mathit{case}}^{\mathit{ndp}} (\Sigma z:\mathit{nat}. P\ z) \langle 0, x \rangle (\lambda m:\mathit{nat}. \langle S\ m, f\ m \rangle) n)$$

Implémentation (2/3)

On définit le schéma d'analyse par cas dépendant par :

$$\begin{aligned}
 \mathit{nat}_{\mathit{case}}^{\mathit{dep}} &: \forall P:\mathit{nat} \rightarrow \mathit{Set}, (P\ 0) \rightarrow (\forall n:\mathit{nat}, P\ (S\ n)) \rightarrow (\forall n:\mathit{nat}, P\ n) \\
 &= \lambda P:\mathit{nat} \rightarrow \mathit{Set}. \\
 &\quad \lambda x:(P\ 0). \lambda f:(\forall n:\mathit{nat}, P\ (S\ n)). \\
 &\quad \lambda n:\mathit{nat}. \pi_2(M_{P,x,f,n}) \quad : P\ (\pi_1(M_{P,x,f,n}))
 \end{aligned}$$

où

$$\begin{aligned}
 \pi_1(M_{P,x,f,n}) &= \pi_1(\mathit{nat}_{\mathit{case}}^{\mathit{ndp}} (\Sigma z:\mathit{nat}. P\ z) \langle 0, x \rangle (\lambda m:\mathit{nat}. \langle S\ m, f\ m \rangle) n) \\
 &\triangleright \mathit{nat}_{\mathit{case}}^{\mathit{ndp}} \mathit{nat} (\pi_1 \langle 0, x \rangle) (\lambda m:\mathit{nat}. \pi_1 \langle S\ m, f\ m \rangle) n
 \end{aligned}$$

Implémentation (2/3)

On définit le schéma d'analyse par cas dépendant par :

$$\begin{aligned}
 \text{nat}_{\text{case}}^{\text{dep}} : & \quad \forall P:\text{nat} \rightarrow \text{Set}, (P\ 0) \rightarrow (\forall n:\text{nat}, P\ (S\ n)) \rightarrow (\forall n:\text{nat}, P\ n) \\
 = & \quad \lambda P:\text{nat} \rightarrow \text{Set}. \\
 & \quad \lambda x:(P\ 0). \lambda f:(\forall n:\text{nat}, P\ (S\ n)). \\
 & \quad \lambda n:\text{nat}. \pi_2(M_{P,x,f,n}) \quad : P\ (\pi_1(M_{P,x,f,n}))
 \end{aligned}$$

où

$$\begin{aligned}
 \pi_1(M_{P,x,f,n}) = & \quad \pi_1(\text{nat}_{\text{case}}^{\text{ndp}} (\Sigma z:\text{nat}. P\ z) \langle 0, x \rangle (\lambda m:\text{nat}. \langle S\ m, f\ m \rangle) n) \\
 \triangleright & \quad \text{nat}_{\text{case}}^{\text{ndp}} \text{ nat} (\pi_1 \langle 0, x \rangle) (\lambda m:\text{nat}. \pi_1 \langle S\ m, f\ m \rangle) n \\
 \triangleright & \quad \text{nat}_{\text{case}}^{\text{ndp}} \text{ nat} 0 (\lambda m:\text{nat}. S\ m) n
 \end{aligned}$$

Implémentation (2/3)

On définit le schéma d'analyse par cas dépendant par :

$$\begin{aligned}
 \mathit{nat}_{\mathit{case}}^{\mathit{dep}} : & \forall P:\mathit{nat} \rightarrow \mathit{Set}, (P\ 0) \rightarrow (\forall n:\mathit{nat}, P\ (S\ n)) \rightarrow (\forall n:\mathit{nat}, P\ n) \\
 = & \lambda P:\mathit{nat} \rightarrow \mathit{Set}. \\
 & \lambda x:(P\ 0). \lambda f:(\forall n:\mathit{nat}, P\ (S\ n)). \\
 & \lambda n:\mathit{nat}. \pi_2(M_{P,x,f,n}) \quad : P\ (\pi_1(M_{P,x,f,n}))
 \end{aligned}$$

où

$$\begin{aligned}
 \pi_1(M_{P,x,f,n}) = & \pi_1(\mathit{nat}_{\mathit{case}}^{\mathit{ndp}} (\Sigma z:\mathit{nat}. P\ z) \langle 0, x \rangle (\lambda m:\mathit{nat}. \langle S\ m, f\ m \rangle) n) \\
 \triangleright & \mathit{nat}_{\mathit{case}}^{\mathit{ndp}} \mathit{nat} (\pi_1 \langle 0, x \rangle) (\lambda m:\mathit{nat}. \pi_1 \langle S\ m, f\ m \rangle) n \\
 \triangleright & \mathit{nat}_{\mathit{case}}^{\mathit{ndp}} \mathit{nat}\ 0 (\lambda m:\mathit{nat}. S\ m) n \\
 \triangleright & \mathit{nat}_{\mathit{case}}^{\mathit{ndp}} \mathit{nat}\ 0\ S\ n
 \end{aligned}$$

Implémentation (2/3)

On définit le schéma d'analyse par cas dépendant par :

$$\begin{aligned}
 \text{nat}_{\text{case}}^{\text{dep}} : & \forall P:\text{nat} \rightarrow \text{Set}, (P\ 0) \rightarrow (\forall n:\text{nat}, P\ (S\ n)) \rightarrow (\forall n:\text{nat}, P\ n) \\
 = & \lambda P:\text{nat} \rightarrow \text{Set}. \\
 & \lambda x:(P\ 0). \lambda f:(\forall n:\text{nat}, P\ (S\ n)). \\
 & \lambda n:\text{nat}. \pi_2(M_{P,x,f,n}) \quad : P\ (\pi_1(M_{P,x,f,n}))
 \end{aligned}$$

où

$$\begin{aligned}
 \pi_1(M_{P,x,f,n}) = & \pi_1(\text{nat}_{\text{case}}^{\text{ndp}} (\Sigma z:\text{nat}. P\ z) \langle 0, x \rangle (\lambda m:\text{nat}. \langle S\ m, f\ m \rangle) n) \\
 \triangleright & \text{nat}_{\text{case}}^{\text{ndp}} \text{nat} (\pi_1 \langle 0, x \rangle) (\lambda m:\text{nat}. \pi_1 \langle S\ m, f\ m \rangle) n \\
 \triangleright & \text{nat}_{\text{case}}^{\text{ndp}} \text{nat}\ 0 (\lambda m:\text{nat}. S\ m) n \\
 \triangleright & \text{nat}_{\text{case}}^{\text{ndp}} \text{nat}\ 0\ S\ n \\
 \triangleright & n
 \end{aligned}$$

Implémentation (3/3)

Il ne reste plus qu'à définir le schéma d'élimination dépendant par point fixe :

$$\begin{aligned}
 \mathit{nat}_{\mathit{rec}}^{\mathit{dep}} = & \lambda P:\mathit{nat} \rightarrow \mathit{Set}. \lambda x:P \ 0. \lambda f:(\forall n:\mathit{nat}. (P \ n) \rightarrow (P \ (S \ n))). \\
 & \mathit{fix}_{\mathit{rec}}(\lambda m:\mathit{nat}. (\mathit{nat}_{\mathit{case}}^{\mathit{dep}} \ \mathit{nat} \ 0 \ (\lambda p. f \ p \ (\mathit{rec} \ p)) \ m))
 \end{aligned}$$

Pour résumer

Pour un inductif donné, on utilise notre codage avec le schéma d'analyse par cas non dépendant. On obtient alors le schéma d'analyse par cas dépendant. Enfin, on obtient le schéma d'élimination général par point fixe.

L'égalité et son schéma d'élimination non dépendant

On ajoute les constantes suivantes :

L'égalité et son schéma d'élimination non dépendant

On ajoute les constantes suivantes :

- ▶ $A : \text{Type}$
- ▶ $x : A$

L'égalité et son schéma d'élimination non dépendant

On ajoute les constantes suivantes :

- ▶ $A : \text{Type}$
- ▶ $x : A$
- ▶ $eq : A \rightarrow \text{Set}$ (et nous noterons $eq\ y$ par $x = y$)

L'égalité et son schéma d'élimination non dépendant

On ajoute les constantes suivantes :

- ▶ $A : \text{Type}$
- ▶ $x : A$
- ▶ $eq : A \rightarrow \text{Set}$ (et nous noterons $eq\ y$ par $x = y$)
- ▶ $refl : x = x$

L'égalité et son schéma d'élimination non dépendant

On ajoute les constantes suivantes :

- ▶ $A : \text{Type}$
- ▶ $x : A$
- ▶ $eq : A \rightarrow \text{Set}$ (et nous noterons $eq\ y$ par $x = y$)
- ▶ $refl : x = x$
- ▶ $eq^{ndp} : \forall P : A \rightarrow \text{Set}, P\ x \rightarrow \forall y : A, (x = y) \rightarrow P\ y$

L'égalité et son schéma d'élimination non dépendant

On ajoute les constantes suivantes :

- ▶ $A : \text{Type}$
- ▶ $x : A$
- ▶ $eq : A \rightarrow \text{Set}$ (et nous noterons $eq\ y$ par $x = y$)
- ▶ $refl : x = x$
- ▶ $eq^{ndp} : \forall P : A \rightarrow \text{Set}, P\ x \rightarrow \forall y : A, (x = y) \rightarrow P\ y$

Ainsi que la règle de réduction du schéma d'élimination :

$$eq^{ndp}\ P\ f\ x\ refl \triangleright f$$

Son schéma d'élimination dépendant

L'égalité n'ayant pas de cas réellement inductif, on peut définir directement son schéma non dépendant :

$$eq^{dep}: \quad \forall P:(\forall a:A, (x = a) \rightarrow Set), (P \times refl) \rightarrow \forall y:A, \forall e:(x = y), P \ y \ e$$

Son schéma d'élimination dépendant

L'égalité n'ayant pas de cas réellement inductif, on peut définir directement son schéma non dépendant :

$$\begin{aligned}
 eq^{dep}: & \quad \forall P:(\forall a:A, (x = a) \rightarrow Set), (P \times refl) \rightarrow \forall y:A, \forall e:(x = y), P \ y \ e \\
 = & \quad \lambda P:(\forall a:A, (x = a) \rightarrow Set). \\
 & \quad \lambda f:(P \times refl). \\
 & \quad \lambda y:A. \lambda e:(x = y). \pi_2(M_{P,f,y,e})
 \end{aligned}$$

Son schéma d'élimination dépendant

L'égalité n'ayant pas de cas réellement inductif, on peut définir directement son schéma non dépendant :

$$\begin{aligned}
 eq^{dep}: & \quad \forall P:(\forall a:A, (x = a) \rightarrow Set), (P \times refl) \rightarrow \forall y:A, \forall e:(x = y), P \ y \ e \\
 = & \quad \lambda P:(\forall a:A, (x = a) \rightarrow Set). \\
 & \quad \lambda f:(P \times refl). \\
 & \quad \lambda y:A. \lambda e:(x = y). \pi_2(M_{P,f,y,e})
 \end{aligned}$$

où

$$M_{P,f,y,e} = eq^{ndp} (\lambda a:A. (\Sigma z:(x = a). P \ a \ z)) \langle refl, f \rangle \ y \ e$$

Les nouvelles règles de réduction

On ajoute les deux nouvelles règles suivantes (commutation) :

$$\pi_1(eq^{ndp} (\lambda a:A. (\sum z:(x = a). P a z)) f y e) \triangleright eq^{ndp} (\lambda a:A. x = a) (\pi_1(f)) y e$$

Les nouvelles règles de réduction

On ajoute les deux nouvelles règles suivantes (commutation et η) :

$$\pi_1(eq^{ndp} (\lambda a:A. (\sum z:(x = a). P a z)) f y e) \triangleright eq^{ndp} (\lambda a:A. x = a) (\pi_1(f)) y e$$

$$eq^{ndp} (\lambda a:A. x = a) refl y e \triangleright e$$

Les nouvelles règles de réduction

On ajoute les deux nouvelles règles suivantes (commutation et η) :

$$\pi_1(eq^{ndp} (\lambda a:A. (\sum z:(x = a). P a z)) f y e) \triangleright eq^{ndp} (\lambda a:A. x = a) (\pi_1(f)) y e$$

$$eq^{ndp} (\lambda a:A. x = a) refl y e \triangleright e$$

et on vérifie bien que eq^{dep} a le type et les réductions attendus.

Les nouvelles règles de réduction

On ajoute les deux nouvelles règles suivantes (commutation et η) :

$$\pi_1(eq^{ndp} (\lambda a:A. (\sum z:(x = a). P a z)) f y e) \triangleright eq^{ndp} (\lambda a:A. x = a) (\pi_1(f)) y e$$

$$eq^{ndp} (\lambda a:A. x = a) refl y e \triangleright e$$

et on vérifie bien que eq^{dep} a le type et les réductions attendus:

$$\pi_1(M_{P,f,y,e}) = \pi_1(eq^{ndp} (\lambda a:A. \sum z:(x = a). P a z) \langle refl, f \rangle y e)$$

Les nouvelles règles de réduction

On ajoute les deux nouvelles règles suivantes (commutation et η) :

$$\begin{aligned} \pi_1(eq^{ndp} (\lambda a:A. (\sum z:(x = a). P a z)) f y e) &\triangleright eq^{ndp} (\lambda a:A. x = a) (\pi_1(f)) y e \\ eq^{ndp} (\lambda a:A. x = a) refl y e &\triangleright e \end{aligned}$$

et on vérifie bien que eq^{dep} a le type et les réductions attendus:

$$\begin{aligned} \pi_1(M_{P,f,y,e}) &= \pi_1(eq^{ndp} (\lambda a:A. \sum z:(x = a). P a z) \langle refl, f \rangle y e) \\ &\triangleright eq^{ndp} (\lambda a:A. (x = a)) (\pi_1 \langle refl, f \rangle) y e \end{aligned}$$

Les nouvelles règles de réduction

On ajoute les deux nouvelles règles suivantes (commutation et η) :

$$\begin{aligned} \pi_1(eq^{ndp} (\lambda a:A. (\sum z:(x = a). P a z)) f y e) &\triangleright eq^{ndp} (\lambda a:A. x = a) (\pi_1(f)) y e \\ eq^{ndp} (\lambda a:A. x = a) refl y e &\triangleright e \end{aligned}$$

et on vérifie bien que eq^{dep} a le type et les réductions attendus:

$$\begin{aligned} \pi_1(M_{P,f,y,e}) &= \pi_1(eq^{ndp} (\lambda a:A. \sum z:(x = a). P a z) \langle refl, f \rangle y e) \\ &\triangleright eq^{ndp} (\lambda a:A. (x = a)) (\pi_1 \langle refl, f \rangle) y e \\ &\triangleright eq^{ndp} (\lambda a:A. (x = a)) refl y e \end{aligned}$$

Les nouvelles règles de réduction

On ajoute les deux nouvelles règles suivantes (commutation et η) :

$$\begin{aligned} \pi_1(eq^{ndp} (\lambda a:A. (\sum z:(x = a). P a z)) f y e) &\triangleright eq^{ndp} (\lambda a:A. x = a) (\pi_1(f)) y e \\ eq^{ndp} (\lambda a:A. x = a) refl y e &\triangleright e \end{aligned}$$

et on vérifie bien que eq^{dep} a le type et les réductions attendus:

$$\begin{aligned} \pi_1(M_{P,f,y,e}) &= \pi_1(eq^{ndp} (\lambda a:A. \sum z:(x = a). P a z) \langle refl, f \rangle y e) \\ &\triangleright eq^{ndp} (\lambda a:A. (x = a)) (\pi_1 \langle refl, f \rangle) y e \\ &\triangleright eq^{ndp} (\lambda a:A. (x = a)) refl y e \\ &\triangleright e \end{aligned}$$

Les nouvelles règles de réduction

On ajoute les deux nouvelles règles suivantes (commutation et η) :

$$\begin{aligned} \pi_1(eq^{ndp} (\lambda a:A. (\sum z:(x = a). P a z)) f y e) &\triangleright eq^{ndp} (\lambda a:A. x = a) (\pi_1(f)) y e \\ eq^{ndp} (\lambda a:A. x = a) refl y e &\triangleright e \end{aligned}$$

et on vérifie bien que eq^{dep} a le type et les réductions attendus:

$$\begin{aligned} \pi_1(M_{P,f,y,e}) &= \pi_1(eq^{ndp} (\lambda a:A. \sum z:(x = a). P a z) \langle refl, f \rangle y e) \\ &\triangleright eq^{ndp} (\lambda a:A. (x = a)) (\pi_1 \langle refl, f \rangle) y e \\ &\triangleright eq^{ndp} (\lambda a:A. (x = a)) refl y e \\ &\triangleright e \end{aligned}$$

Donc, $M_{P,f,y,e}$ ayant pour type $\sum z:(x = y). P y z$,
 $\pi_2(M_{P,f,y,e})$ a bien pour type $P y e$.

Pour aller plus loin ...

Pour les inductifs en général, 2 options :

Pour aller plus loin ...

Pour les inductifs en général, 2 options :

- ▶ Vérifier que le codage marche sur les inductifs définis “à la Coq”.

Pour aller plus loin ...

Pour les inductifs en général, 2 options :

- ▶ Vérifier que le codage marche sur les inductifs définis “à la Coq”.
- ▶ Utiliser le fait que l’on peut recoder les inductifs à partir de l’inductif W (Well-ordering type).

Pour aller plus loin ...

Pour les inductifs en général, 2 options :

- ▶ Vérifier que le codage marche sur les inductifs définis “à la Coq”.
- ▶ Utiliser le fait que l’on peut recoder les inductifs à partir de l’inductif W (Well-ordering type).

Or, le schéma d’élimination dépendant de l’inductif W se code sans plus de problème que celui de nat .

Pour aller plus loin ...

Pour les inductifs en général, 2 options :

- ▶ Vérifier que le codage marche sur les inductifs définis “à la Coq”.
- ▶ Utiliser le fait que l’on peut recoder les inductifs à partir de l’inductif W (Well-ordering type).

Or, le schéma d’élimination dépendant de l’inductif W se code sans plus de problème que celui de nat .

De plus, Goguen et Luo ont montré que pour effectuer ce codage, on devait se donner des règles η (légèrement plus générales que les nôtres, mais les impliquants trivialement).

Pour conclure

- ▶ On arrive à exprimer les schémas d'élimination dépendant en se donnant les schémas non dépendant, les Σ -types et des règles de commutation avec π_1 et d' η -conversion pour chaque inductif.

Pour conclure

- ▶ On arrive à exprimer les schémas d'élimination dépendant en se donnant les schémas non dépendant, les Σ -types et des règles de commutation avec π_1 et d' η -conversion pour chaque inductif.
- ▶ Remarquons qu'on ne sait le faire qu'en présence d'une définition par point fixe.

Pour conclure

- ▶ On arrive à exprimer les schémas d'élimination dépendant en se donnant les schémas non dépendant, les Σ -types et des règles de commutation avec π_1 et d' η -conversion pour chaque inductif.
- ▶ Remarquons qu'on ne sait le faire qu'en présence d'une définition par point fixe.

Maintenant, on pourrait :

Pour conclure

- ▶ On arrive à exprimer les schémas d'élimination dépendant en se donnant les schémas non dépendant, les Σ -types et des règles de commutation avec π_1 et d' η -conversion pour chaque inductif.
- ▶ Remarquons qu'on ne sait le faire qu'en présence d'une définition par point fixe.

Maintenant, on pourrait :

- ▶ S'intéresser à ce problème de commutation dans les cas réellement inductifs.

Pour conclure

- ▶ On arrive à exprimer les schémas d'élimination dépendant en se donnant les schémas non dépendant, les Σ -types et des règles de commutation avec π_1 et d' η -conversion pour chaque inductif.
- ▶ Remarquons qu'on ne sait le faire qu'en présence d'une définition par point fixe.

Maintenant, on pourrait :

- ▶ S'intéresser à ce problème de commutation dans les cas réellement inductifs.
- ▶ S'assurer que les règles de réduction ajoutées ne cassent bien aucunes des propriétés du système.

Pour conclure

- ▶ On arrive à exprimer les schémas d'élimination dépendant en se donnant les schémas non dépendant, les Σ -types et des règles de commutation avec π_1 et d' η -conversion pour chaque inductif.
- ▶ Remarquons qu'on ne sait le faire qu'en présence d'une définition par point fixe.

Maintenant, on pourrait :

- ▶ S'intéresser à ce problème de commutation dans les cas réellement inductifs.
- ▶ S'assurer que les règles de réduction ajoutées ne cassent bien aucunes des propriétés du système.
- ▶ S'intéresser au codage pour tous les inductifs "à la Coq"

MERCI !