

Une nouvelle présentation de l'Arithmétique de Heyting en déduction modulo

Lisa Allali
sous la direction de Gilles Dowek

October 1, 2006



Projet LogiCal



Plan

- Présentation générale
- Quelques exemples
- Objectif
- Comment passer d'axiomes à règles de réécriture
- Applications et perspectives en Coq
- Conclusion

La déduction modulo

Déduction modulo = raisonnement + calcul :

- les règles logique de la déduction naturelle
- une congruence \equiv

Quelques exemples de règles de la déduction modulo :

La déduction modulo

Déduction modulo = raisonnement + calcul :

- les règles logiques de la déduction naturelle
- une congruence \equiv

Quelques exemples de règles de la déduction modulo :

$$\frac{}{\Gamma \vdash_{\equiv} B} \text{Ax si } A \in \Gamma \text{ et } A \equiv B$$

$$\frac{\Gamma \vdash_{\equiv} C \quad \Gamma \vdash_{\equiv} A}{\Gamma \vdash_{\equiv} B} \Rightarrow_e \text{ si } C \equiv A \Rightarrow B$$

$$\frac{\Gamma \vdash_{\equiv} A \quad \Gamma \vdash_{\equiv} B}{\Gamma \vdash_{\equiv} C} \wedge_i \text{ si } C \equiv A \wedge B$$

Théories en déduction modulo

Une **théorie axiomatique** est l'ensemble des théorèmes prouvables à partir d'un ensemble d'axiomes \mathcal{A} .

Théories en déduction modulo

Une **théorie axiomatique** est l'ensemble des théorèmes prouvables à partir d'un ensemble d'axiomes \mathcal{A} .

Une **théorie modulo** est l'ensemble des théorèmes prouvables à partir d'un ensemble d'axiomes \mathcal{A} et d'une congruence \equiv .

Théories en déduction modulo

Une **théorie axiomatique** est l'ensemble des théorèmes prouvables à partir d'un ensemble d'axiomes \mathcal{A} .

Une **théorie modulo** est l'ensemble des théorèmes prouvables à partir d'un ensemble d'axiomes \mathcal{A} et d'une congruence \equiv .

Une **théorie purement calculatoire** est l'ensemble des théorèmes prouvables à partir d'une congruence \equiv .

Théories en déduction modulo

Une **théorie axiomatique** est l'ensemble des théorèmes prouvables à partir d'un ensemble d'axiomes \mathcal{A} .

Une **théorie modulo** est l'ensemble des théorèmes prouvables à partir d'un ensemble d'axiomes \mathcal{A} et d'une congruence \equiv .

Une **théorie purement calculatoire** est l'ensemble des théorèmes prouvables à partir d'une congruence \equiv .

Quels sont les théorèmes d'une théorie qui n'a plus d'axiome ?

Théories en déduction modulo

Une **théorie axiomatique** est l'ensemble des théorèmes prouvables à partir d'un ensemble d'axiomes \mathcal{A} .

Une **théorie modulo** est l'ensemble des théorèmes prouvables à partir d'un ensemble d'axiomes \mathcal{A} et d'une congruence \equiv .

Une **théorie purement calculatoire** est l'ensemble des théorèmes prouvables à partir d'une congruence \equiv .

Quels sont les théorèmes d'une théorie qui n'a plus d'axiome ?

Les propositions congrues à \top ou à une proposition équivalente ($\top \Rightarrow T$, $T \wedge T \dots$ ou des propositions plus compliquées mais toujours équivalentes à \top).

Présentation

Quelques exemples

Objectifs

D'axiomes à congruence

Applications et perspectives Coq

Conclusion

Exemple trivial

$1+1=2$ en arithmétique axiomatique

$1+1=2$ en arithmétique modulo avec Leibniz

$1+1=2$ en arithmétique modulo sans Leibniz

Quelques exemples

Exemple trivial

Soit la théorie purement calculatoire dont la congruence est définie par le système de réécriture suivant :

$$(1) A \longrightarrow B$$

$$(2) B \longrightarrow C$$

Voici une preuve en déduction modulo de $\vdash_{\equiv} A \Rightarrow C$:

Exemple trivial

Soit la théorie purement calculatoire dont la congruence est définie par le système de réécriture suivant :

$$(1) A \longrightarrow B$$

$$(2) B \longrightarrow C$$

Voici une preuve en déduction modulo de $\vdash_{\equiv} A \Rightarrow C$:

$$\frac{\overline{A \vdash_{\equiv} C} \quad Ax, A \equiv C}{\vdash_{\equiv} A \Rightarrow C} \Rightarrow i$$

Exemple trivial

Soit la théorie purement calculatoire dont la congruence est définie par le système de réécriture suivant :

$$(1) A \longrightarrow B$$

$$(2) B \longrightarrow C$$

Voici une preuve en déduction modulo de $\vdash_{\equiv} A \Rightarrow C$:

$$\frac{\overline{A \vdash_{\equiv} C} \quad Ax, A \equiv C}{\vdash_{\equiv} A \Rightarrow C} \Rightarrow i$$

En effet, $A \longrightarrow B$ par la règle (1) et $B \longrightarrow C$ par (2) d'où $A \equiv C$

Exemple trivial

Soit la théorie purement calculatoire dont la congruence est définie par le système de réécriture suivant :

$$(1) A \longrightarrow B$$

$$(2) B \longrightarrow C$$

Voici une preuve en déduction modulo de $\vdash_{\equiv} A \Rightarrow C$:

$$\frac{\overline{A \vdash_{\equiv} C} \quad Ax, A \equiv C}{\vdash_{\equiv} A \Rightarrow C} \Rightarrow i$$

En effet, $A \longrightarrow B$ par la règle (1) et $B \longrightarrow C$ par (2) d'où $A \equiv C$

Remarque : Si on remplace la règle (2) par $C \longrightarrow B$ ça marche aussi

1+1=2 en arithmétique axiomatique

Quelques axiomes de l'Arithmétique de Heyting

(Leib) $\forall x \forall y x = y \Rightarrow P(x) \Rightarrow P(y)$

(Zero_neutre) $\forall y (0 + y = y)$

(Add_Succ) $\forall x \forall y (S(x) + y = S(x + y))$

Voici une preuve en arithmétique de $\vdash_{HA} 1 + 1 = 2$:

1+1=2 en arithmétique axiomatique

Quelques axiomes de l'Arithmétique de Heyting

(Leib) $\forall x \forall y x = y \Rightarrow P(x) \Rightarrow P(y)$

(Zero_neutre) $\forall y (0 + y = y)$

(Add_Succ) $\forall x \forall y (S(x) + y = S(x + y))$

Voici une preuve en arithmétique de $\vdash_{HA} 1 + 1 = 2$:

$$\frac{\frac{}{\vdash_{HA} 0 + y = y} \text{Ax}}{\vdash_{HA} 0 + S(0) = S(0)} \forall e$$

$$\frac{\frac{\frac{\text{Instance du schéma de Leibniz}}{\vdash_{HA} x = y \Rightarrow S(0) + S(0) = S(x) \Rightarrow S(0) + S(0) = S(y)} \text{Ax}}{\vdash_{HA} 0 + S(0) = S(0) \Rightarrow S(0) + S(0) = S(0 + S(0)) \Rightarrow S(0) + S(0) = S(S(0))} \forall e2}{\frac{\vdash_{HA} S(0) + S(0) = S(0 + S(0)) \Rightarrow S(0) + S(0) = S(S(0))}{\vdash_{HA} S(0) + S(0) = S(S(0))} \forall e2} \Rightarrow e$$

1+1=2 en arithmétique modulo avec Leibniz

Soit la théorie purement calculatoire dont la congruence est définie par le système de réécriture suivant :

$$\text{(Leib)} \quad y = z \longrightarrow \forall p (y \in p \Rightarrow z \in p)$$

$$\text{(Zero_neutre)} \quad 0 + y \longrightarrow y$$

$$\text{(Add_Suc)} \quad S(x) + y \longrightarrow S(x + y)$$

Voici une preuve dans ce système de $\vdash_{\equiv} 1 + 1 = 2$:

1+1=2 en arithmétique modulo avec Leibniz

Soit la théorie purement calculatoire dont la congruence est définie par le système de réécriture suivant :

(Leib) $y = z \longrightarrow \forall p (y \in p \Rightarrow z \in p)$

(Zero_neutre) $0 + y \longrightarrow y$

(Add_Suc) $S(x) + y \longrightarrow S(x + y)$

Voici une preuve dans ce système de $\vdash_{\equiv} 1 + 1 = 2$:

$$\frac{\frac{\frac{\frac{}{1 + 1 \in p \vdash_{HA_R} 2 \in p}}{\vdash_{HA_R} 1 + 1 \in p \Rightarrow 2 \in p} \Rightarrow_i}{\vdash_{HA_R} \forall p 1 + 1 \in p \Rightarrow 2 \in p} \forall_i}{\vdash_{HA_R} 1 + 1 = 2} \text{réécriture - Leibniz}}{Ax, 1 + 1 \in p \equiv 2 \in p}$$

1+1=2 en arithmétique modulo sans Leibniz

Soit la théorie purement calculatoire dont la congruence est définie par le système de réécriture suivant :

- (1) $0 = 0 \longrightarrow \top$
- (2) $S(x) = S(y) \longrightarrow x = y$
- (3) $0 + y \longrightarrow y$
- (4) $S(x) + y \longrightarrow S(x + y)$

Voici une preuve en déduction modulo de $\vdash_{\equiv} 1 + 1 = 2$:

1+1=2 en arithmétique modulo sans Leibniz

Soit la théorie purement calculatoire dont la congruence est définie par le système de réécriture suivant :

$$(1) 0 = 0 \longrightarrow \top$$

$$(2) S(x) = S(y) \longrightarrow x = y$$

$$(3) 0 + y \longrightarrow y$$

$$(4) S(x) + y \longrightarrow S(x + y)$$

Voici une preuve en déduction modulo de $\vdash_{\equiv} 1 + 1 = 2$:

$$\frac{}{\vdash_{\equiv} 1 + 1 = 2} \text{Ax, } 1 + 1 = 2 \equiv \top$$

$1+1=2$ en arithmétique modulo sans Leibniz

Soit la théorie purement calculatoire dont la congruence est définie par le système de réécriture suivant :

$$(1) 0 = 0 \longrightarrow \top$$

$$(2) S(x) = S(y) \longrightarrow x = y$$

$$(3) 0 + y \longrightarrow y$$

$$(4) S(x) + y \longrightarrow S(x + y)$$

Voici une preuve en déduction modulo de $\vdash_{\equiv} 1 + 1 = 2$:

$$\frac{}{\vdash_{\equiv} 1 + 1 = 2} \text{Ax, } 1 + 1 = 2 \equiv \top$$

En effet,

$$1 + 1 = 2 \longrightarrow^* 2 = 2 \text{ par les règles (4) et (3)}$$

$$2 = 2 \longrightarrow^* 0 = 0 \text{ par (2)}$$

$$0 = 0 \longrightarrow \top \text{ par (1) d'où } 1 + 1 = 2 \equiv \top$$

Objectifs

Objectif

Le stage a consisté à :

- comprendre la présentation de l'arithmétique en déduction modulo de Gilles Dowek et Benjamin Werner
- intégrer à cette présentation un aspect **essentiel** de l'arithmétique:
la décidabilité de l'égalité

Passer l'exemple modulo avec Leibniz à l'exemple modulo sans Leibniz :

$$x = y \longrightarrow P(x) = P(y).$$

↔ **algorithme** (règles de réécriture) qui décide de l'égalité

L'Arithmétique de Heyting

d'une présentation axiomatique
à
une présentation purement calculatoire

Présentation axiomatique de l'Arithmétique de Heyting

Le langage

Les symboles de fonctions 0 , S , $+$, \times et le symbole de prédicat $=$

Le schéma d'axiomes de la récurrence

$$(P\{x := 0\} \wedge \forall y (P\{x := y\} \Rightarrow P\{x := S(y)\})) \Rightarrow \forall n P\{x := n\}$$

Les axiomes de l'égalité

$$\forall x x = x$$

$$\forall x \forall y x = y \Rightarrow P(x) = P(y)$$

$$\forall x 0 = S(x) \Rightarrow \perp$$

$$\forall x \forall y (S(x) = S(y) \Rightarrow x = y)$$

Les axiomes sur l'addition et la multiplication

$$\forall y (0 + y = y)$$

$$\forall x \forall y (S(x) + y = S(x + y))$$

$$\forall y (0 \times y = 0)$$

$$\forall x \forall y (S(x) \times y = x \times y + y)$$

Définir l'égalité avec des règles de réécriture

sans utiliser Leibniz

Les axiomes sur l'addition et la multiplication

$$\forall y (0 + y = y)$$

$$\forall x \forall y (S(x) + y = S(x + y))$$

$$\forall y (0 \times y = 0)$$

$$\forall x \forall y (S(x) \times y = x \times y + y)$$

Les axiomes sur l'addition et la multiplication

$$\forall y (0 + y = y) \quad \rightsquigarrow \quad 0 + y \longrightarrow y$$

$$\forall x \forall y (S(x) + y = S(x + y))$$

$$\forall y (0 \times y = 0)$$

$$\forall x \forall y (S(x) \times y = x \times y + y)$$

Les axiomes sur l'addition et la multiplication

$$\forall y (0 + y = y) \quad \rightsquigarrow \quad 0 + y \longrightarrow y$$

$$\forall x \forall y (S(x) + y = S(x + y)) \quad \rightsquigarrow \quad S(x) + y \longrightarrow S(x + y)$$

$$\forall y (0 \times y = 0) \quad \rightsquigarrow \quad 0 \times y \longrightarrow 0$$

$$\forall x \forall y (S(x) \times y = x \times y + y) \quad \rightsquigarrow \quad S(x) \times y \longrightarrow x \times y + y$$

Les axiomes de l'égalité - Version précédente

$$\forall x \ x = x$$

$$\forall x \ 0 = S(x) \Rightarrow \perp$$

$$\forall x \ \forall y \ (S(x) = S(y) \Rightarrow x = y)$$

$$\forall x \ \forall y \ x = y \Rightarrow P(x) = P(y)$$

$$\rightsquigarrow \quad 0 = S(x) \longrightarrow \perp$$

$$\rightsquigarrow \quad S(x) = S(y) \longrightarrow x = y$$

$$\rightsquigarrow \quad x = y \longrightarrow P(x) = P(y)$$

Les axiomes de l'égalité - Version précédente

$$\forall x \ x = x$$

$$\forall x \ 0 = S(x) \Rightarrow \perp \quad \rightsquigarrow \quad 0 = S(x) \longrightarrow \perp$$

$$\forall x \ \forall y \ (S(x) = S(y) \Rightarrow x = y) \quad \rightsquigarrow \quad S(x) = S(y) \longrightarrow x = y$$

$$\forall x \ \forall y \ x = y \Rightarrow P(x) = P(y) \quad \rightsquigarrow \quad x = y \longrightarrow P(x) = P(y)$$

La présentation précédente de l'arithmétique de Heyting en déduction modulo **définissait** l'égalité par le principe de **substitution**.

De nouvelles règles pour réécrire l'égalité

$$\forall x \ x = x$$

$$\forall x \ 0 = S(x) \Rightarrow \perp$$

$$\forall x \ \forall y \ (S(x) = S(y) \Rightarrow x = y)$$

$$\forall x \ \forall y \ x = y \Rightarrow P(x) = P(y)$$

De nouvelles règles pour réécrire l'égalité

$$\forall x \ x = x$$

$$\forall x \ 0 = S(x) \Rightarrow \perp$$

$$\forall x \ \forall y \ (S(x) = S(y) \Rightarrow x = y)$$

$$\forall x \ \forall y \ x = y \Rightarrow P(x) = P(y) \quad \rightsquigarrow$$

*NON, on veut un
algorithme qui calcule*

De nouvelles règles pour réécrire l'égalité

$$\forall x \ x = x$$

$$\forall x \ 0 = S(x) \Rightarrow \perp$$

$$\rightsquigarrow \begin{array}{l} 0 = S(x) \longrightarrow \perp \\ S(x) = 0 \longrightarrow \perp \end{array}$$

$$\forall x \ \forall y \ (S(x) = S(y) \Rightarrow x = y)$$

$$\forall x \ \forall y \ x = y \Rightarrow P(x) = P(y)$$

\rightsquigarrow *NON, on veut un
algorithme qui calcule*

De nouvelles règles pour réécrire l'égalité

$$\forall x \ x = x$$

$$\forall x \ 0 = S(x) \Rightarrow \perp$$

$$\rightsquigarrow \begin{array}{l} 0 = S(x) \longrightarrow \perp \\ S(x) = 0 \longrightarrow \perp \end{array}$$

$$\forall x \ \forall y \ (S(x) = S(y) \Rightarrow x = y)$$

$$\rightsquigarrow S(x) = S(y) \longrightarrow x = y$$

$$\forall x \ \forall y \ x = y \Rightarrow P(x) = P(y)$$

$$\rightsquigarrow \text{NON, on veut un} \\ \text{algorithme qui calcule}$$

De nouvelles règles pour réécrire l'égalité

$$\forall x \ x = x$$

$$\rightsquigarrow \quad 0 = 0 \longrightarrow \top$$

$$\forall x \ 0 = S(x) \Rightarrow \perp$$

$$\rightsquigarrow \quad \begin{array}{l} 0 = S(x) \longrightarrow \perp \\ S(x) = 0 \longrightarrow \perp \end{array}$$

$$\forall x \ \forall y \ (S(x) = S(y) \Rightarrow x = y)$$

$$\rightsquigarrow \quad S(x) = S(y) \longrightarrow x = y$$

$$\forall x \ \forall y \ x = y \Rightarrow P(x) = P(y)$$

$$\rightsquigarrow \quad \text{NON, on veut un} \\ \text{algorithme qui calcule}$$

De nouvelles règles pour réécrire l'égalité

$$\forall x \ x = x \quad \rightsquigarrow \quad 0 = 0 \longrightarrow \top$$

$$\forall x \ 0 = S(x) \Rightarrow \perp \quad \rightsquigarrow \quad \begin{array}{l} 0 = S(x) \longrightarrow \perp \\ S(x) = 0 \longrightarrow \perp \end{array}$$

$$\forall x \ \forall y \ (S(x) = S(y) \Rightarrow x = y) \quad \rightsquigarrow \quad S(x) = S(y) \longrightarrow x = y$$

$$\forall x \ \forall y \ x = y \Rightarrow P(x) = P(y) \quad \rightsquigarrow \quad \text{NON, on veut un } \textit{algorithme qui calcule}$$

L'égalité définie par cet **algorithme** suffit. Le principe de **substitution** est **prouvable** avec ces règles de réécriture.

Résultats

- HA_{\rightarrow} est une extension conservative de HA
Le coeur du travail de stage : Prouver que les 4 nouvelles règles qui définissent l'égalité valident Leibniz

Résultats

- HA_{\rightarrow} est une extension conservative de HA
Le coeur du travail de stage : Prouver que les 4 nouvelles règles qui définissent l'égalité valident Leibniz
- HA_{\rightarrow} a la propriété d'élimination des coupures

Applications et perspectives Coq

Définitions en Coq

Les entiers

```
Inductive my_int :Set :=  
| Z : my_int  
| S : my_int -> my_int.
```

L'addition

```
Fixpoint my_plus (n m : my_int) struct n : my_int :=  
match n with  
| Z => m  
| S t => S ( my_plus t m) end.
```

La multiplication

```
Fixpoint my_mult (n m : my_int) struct n : my_int :=  
match n with  
| Z => Z  
| S t => my_plus (my_mult t m) m end.
```

Définitions en Coq

L'égalité

```
Fixpoint my_eq (n m : my_int) struct n : Prop :=  
match n with  
| Z => match m with  
| Z => True  
| _ => False  
end  
| S t => match m with  
| Z => False  
| S u => my_eq t u  
end  
end.
```

Premiers résultats

- **Preuves dans Coq** de toutes nos propriétés sur l'addition, la multiplication et l'égalité
- Premiers tests de **comparaisons sur les termes de preuve** concluants

Comparaison de deux termes de preuves

Avec notre égalité

Lemma lemme1 :

```
not (my_eq (S (S (S (S (S Z))))))  
      (S (S (S (S (S (S (S (S Z)))))) )))) )
```

Avec l'égalité de Coq

Lemma lemme2 :

```
not ( S (S (S (S (S Z)))) =  
      S (S (S (S (S (S (S (S Z)))))))) )
```

Comparaison de deux termes de preuves

Avec notre égalité

```
Lemma lemme1 :  
not (my_eq (S (S (S (S (S Z))))  
  (S (S (S (S (S (S (S (S Z))))))))) )
```

Le terme de preuve

```
lemme1 =  
fun H : False => H  
  : my_eq (S (S (S (S (S Z))))  
    (S (S (S (S (S (S (S (S Z)))))))))
```

Avec l'égalité de Coq

```
Lemma lemme2 :  
not ( S (S (S (S (S (S Z)))) =  
  S (S (S (S (S (S (S (S (S (S (S (S Z))))))))))) )
```

Le terme de preuve

```
lemme2 =  
fun H : S (S (S (S (S Z)))) =  
  S (S (S (S (S (S (S (S (S (S (S (S Z)))))))))) =>  
let H0 :=  
  eq_ind (S (S (S (S (S Z))))  
    (fun ee : my_int =>  
      match ee with  
      | Z => False  
      | S m =>  
        match m with  
        | Z => False  
        | S m0 =>  
          match m0 with  
          | Z => False  
          | S m1 =>  
            match m1 with  
            | Z => False  
            | S m2 =>  
              match m2 with  
              | Z => False  
              | S m3 => match m3 with  
              | Z => True  
              | S _ => False  
              end end end end end end )  
          I (S (S (S (S (S (S (S (S (S (S (S (S Z))))))))))) H in  
          False_ind False H0  
          : S (S (S (S (S Z)))) <>  
          S (S (S (S (S (S (S (S Z))))))))
```

Perspectives à court terme en Coq

- Coder le système de la déduction modulo
- Coder le calcul des prédicats comme un type inductif
- Implémenter la preuve que notre nouvelle théorie valide la propriété de substitution de Leibniz

Conclusion

Conclusion sur le travail accompli

Théorie purement calculatoire de l'arithmétique de Heyting

- qui tire parti de la décidabilité de l'égalité
- qui a la propriété d'élimination des coupures

Les gains par rapport à la version précédente de l'arithmétique de Heyting modulo :

Conclusion sur le travail accompli

Théorie purement calculatoire de l'arithmétique de Heyting

- qui tire parti de la décidabilité de l'égalité
- qui a la propriété d'élimination des coupures

Les gains par rapport à la version précédente de l'arithmétique de Heyting modulo :

- langage plus simple
- réduction de la taille des preuves dans le langage
- la preuve que $HA \longrightarrow$ est une extension conservative de HA est plus simple

Perspectives

- Généraliser nos résultats sur les types inductifs :
trouver automatiquement pour un type inductif donné un
algorithme qui calcule son égalité.
- Utiliser nos recherches dans Coq (tactique `discriminate` ...)