Traduire l'univers des mathématiques en DEDUKTI, sans univers

Amélie LEDEIN et Elliot BUTTE

Vers l'interopérabilité des preuves grâce à DEDUKTI

- Logical framework basé sur le $\lambda\Pi$-CALCUL MODULO THÉORIE
  $\rightarrow$ λ-calcul + types dépendants + règles de réécriture

- Exemple d'un encodage avec des entiers naturels :

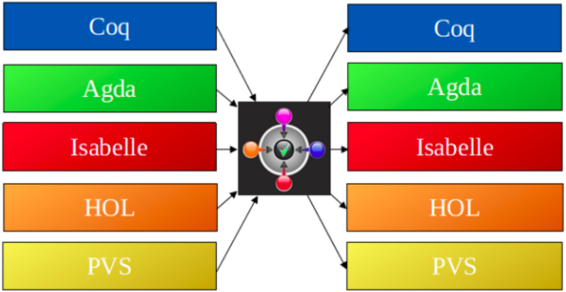
```
constant symbol Nat : TYPE;
constant symbol 0 : Nat;
constant symbol S : Nat → Nat;

symbol + : Nat → Nat → Nat;
rule $m + 0 ↪ $m;
rule $m + (S $n) ↪ S ($m + $n);

symbol add_10 : Π (_ : Nat), Nat := λ x, x + 10;
```

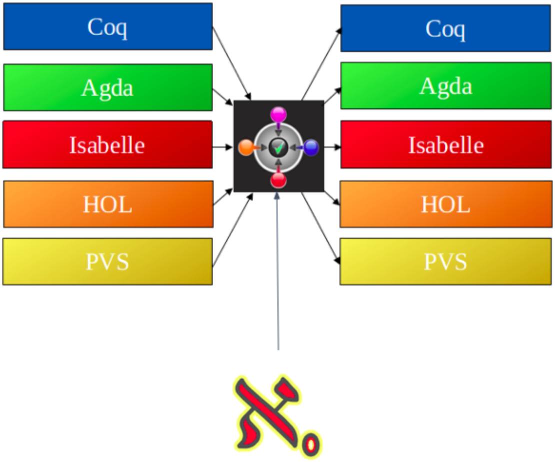
Vers l'interopérabilité des preuves grâce à DEDUKTI

- Logical framework basé sur le $\lambda\Pi$-CALCUL MODULO THÉORIE
  $\rightarrow$ λ-calcul + types dépendants + règles de réécriture

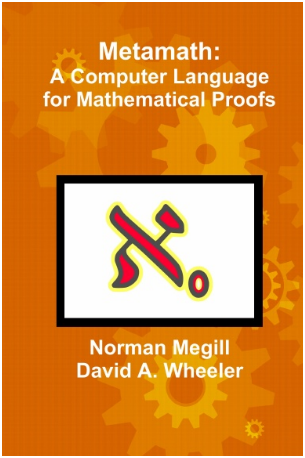


Vers l'interopérabilité des preuves grâce à Dedukti

- Logical framework basé sur le $\lambda\Pi$-calcul modulo théorie
  $\rightarrow$ λ-calcul + types dépendants + règles de réécriture

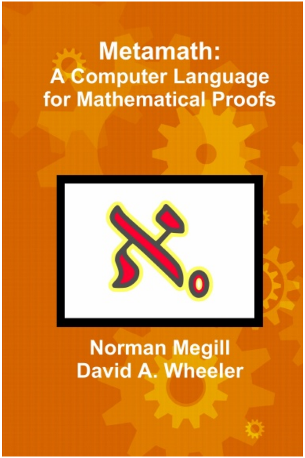


- Logical framework
- Norman Megill
  David A. Wheeler
- États-Unis, début année 90



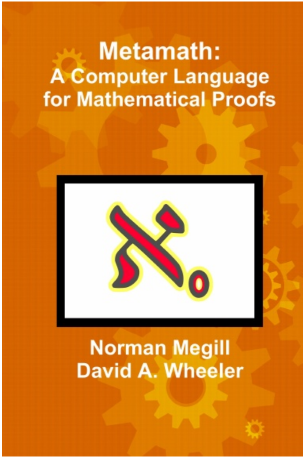
Vue d'ensemble de METAMATH - `metamath.org`

- Logical framework
- Norman Megill
  David A. Wheeler
- États-Unis, début année 90
- Formalisation :
  - 74 sur 100 théorèmes de la liste de Freek Wiedijk



Vue d'ensemble de METAMATH - metamath.org

- Logical framework
- Norman Megill
  David A. Wheeler
- États-Unis, début année 90
- Formalisation :
  - 74 sur 100 théorèmes de la liste de Freek Wiedijk
  - Fondements logiques du framework $\mathbb{K}$



Extrait de la logique propositionnelle à la Hilbert

```
$c ( ) wff -> |-  $.
```

$(Déclaration de 5 constantes (*tokens*) $)

Extrait de la logique propositionnelle à la Hilbert

```
$c ( ) wff -> |-  $.            $( Déclaration de 5 constantes (tokens) $)

${

    wi $a wff ( φ -> ψ ) $. $}    $( Déclaration d'un axiome $)
```

Extrait de la logique propositionnelle à la Hilbert

```
$c ( ) wff -> |- $.                $( Déclaration de 5 constantes (tokens) $)

${ $v φ ψ $.                       $( Déclaration de 2 variables $)


    wi $a wff ( φ -> ψ ) $. $}     $( Déclaration d'un axiome $)
```

Extrait de la logique propositionnelle à la Hilbert

```
$c ( ) wff -> |-  $.                $( Déclaration de 5 constantes (tokens) $)

${  $v φ ψ $.                       $( Déclaration de 2 variables $)
    wph1 $f wff φ $.                $( Typage des variables $)
    wps1 $f wff ψ $.
    wi $a wff ( φ -> ψ ) $. $}      $( Déclaration d'un axiome $)
```

Extrait de la logique propositionnelle à la Hilbert

```
$c ( ) wff -> |-  $.                   $( Déclaration de 5 constantes (tokens) $)

${  $v φ ψ $.                          $( Déclaration de 2 variables $)
    wph1 $f wff φ $.                   $( Typage des variables $)
    wps1 $f wff ψ $.
    wi $a wff ( φ -> ψ ) $. $}         $( Déclaration d'un axiome $)

${  $v φ ψ χ $.
    wph2 $f wff φ $.
    wps2 $f wff ψ $.
    wch2 $f wff χ $.
    a2 $a |- ((φ -> (ψ -> χ)) -> ((φ -> ψ) -> (φ -> χ))) $. $}
```

$$\frac{}{\vdash ((\varphi \texttt{->} (\psi \texttt{->} \chi)) \texttt{->} ((\varphi \texttt{->} \psi) \texttt{->} (\varphi \texttt{->} \chi)))}\ \texttt{a2}$$

Extrait de la logique propositionnelle à la Hilbert

```
$c ( ) wff -> |-  $.                    $( Déclaration de 5 constantes (tokens) $)

${  $v φ ψ $.                           $( Déclaration de 2 variables $)
    wph1 $f wff φ $.                    $( Typage des variables $)
    wps1 $f wff ψ $.
    wi $a wff ( φ -> ψ ) $. $}          $( Déclaration d'un axiome $)

${  $v φ ψ χ $.
    wph2 $f wff φ $.
    wps2 $f wff ψ $.
    wch2 $f wff χ $.
    a2 $a |- ((φ -> (ψ -> χ)) -> ((φ -> ψ) -> (φ -> χ))) $. $}

${  $v φ ψ $.
    wph3 $f wff φ $.
    wps3 $f wff ψ $.
    min $e |- φ $.
    maj $e |- ( φ -> ψ ) $.
    mp  $a |- ψ $.  $}
```

$$\frac{\vdash \varphi \qquad \vdash (\varphi \texttt{ -> } \psi)}{\vdash \psi}\ \texttt{mp}$$

Extrait de la logique propositionnelle à la Hilbert

```
$c ( ) wff -> |-  $.                $( Déclaration de 5 constantes (tokens) $)

$v φ ψ χ $.                         $( Déclaration de 3 variables $)

${  wph1 $f wff φ $.
    wps1 $f wff ψ $.
    wi $a wff ( φ -> ψ ) $.  $}

${  wph2 $f wff φ $.
    wps2 $f wff ψ $.
    wch2 $f wff χ $.
    a2 $a |- ((φ -> (ψ -> χ)) -> ((φ -> ψ) -> (φ -> χ))) $. $}

${  wph3 $f wff φ $.
    wps3 $f wff ψ $.
    min $e |- φ $.
    maj $e |- ( φ -> ψ ) $.
    mp  $a |- ψ $.  $}
```

Extrait de la logique propositionnelle à la Hilbert

```
$c ( ) wff -> |-  $.                $( Déclaration de 5 constantes (tokens) $)

$v φ ψ χ $.                         $( Déclaration de 3 variables $)

wph $f wff φ $.                     $( Typage des variables $)
wps $f wff ψ $.
wch $f wff χ $.

${  wi $a wff ( φ -> ψ ) $.  $}

${  a2 $a |- ((φ -> (ψ -> χ)) -> ((φ -> ψ) -> (φ -> χ))) $. $}

${  min $e |- φ $.
    maj $e |- ( φ -> ψ ) $.
    mp  $a |- ψ $.  $}
```

Extrait de la logique propositionnelle à la Hilbert

```
$c ( ) wff -> |-  $.                 $( Déclaration de 5 constantes (tokens) $)

$v φ ψ χ $.                          $( Déclaration de 3 variables $)

wph $f wff φ $.                      $( Typage des variables $)
wps $f wff ψ $.
wch $f wff χ $.

${  wi $a wff ( φ -> ψ ) $.  $}

${  a2 $a |- ((φ -> (ψ -> χ)) -> ((φ -> ψ) -> (φ -> χ))) $. $}

min $e |- φ $.

${  maj $e |- ( φ -> ψ ) $.
    mp  $a |- ψ $.  $}
```

Extrait de la logique propositionnelle à la Hilbert

```
$c ( ) wff -> |-  $.            $( Déclaration de 5 constantes (tokens) $)

$v φ ψ χ $.                     $( Déclaration de 3 variables $)

wph $f wff φ $.                 $( Typage des variables $)
wps $f wff ψ $.
wch $f wff χ $.

${  wi $a wff ( φ -> ψ ) $.  $}

${  a2 $a |- ((φ -> (ψ -> χ)) -> ((φ -> ψ) -> (φ -> χ))) $.  $}

min $e |- φ $.

${  maj $e |- ( φ -> ψ ) $.
    mp  $a |- ψ $.  $}
```

Comment faire une preuve en Metamath ?

Vérifier une preuve écrite en Metamath

```
$v φ ψ χ $.
wph $f wff φ $.
wps $f wff ψ $.
wch $f wff χ $.
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$= ...preuve...  $.
```

Vérifier une preuve écrite en Metamath

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$= ...preuve...  $.
```

$$\dfrac{\Gamma \vdash (\varphi \to (\psi \to \chi)) \qquad \dfrac{}{\Gamma \vdash (\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))}\,\texttt{a2}}{\Gamma \vdash ((\varphi \to \psi) \to (\varphi \to \chi))}\,\texttt{mp}$$

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

...correspond à
un arbre de preuve
en notation polonaise
inversée...

$$\dfrac{\Gamma \vdash (\varphi \to (\psi \to \chi)) \qquad \dfrac{}{\Gamma \vdash (\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))}\,\texttt{a2}}{\Gamma \vdash ((\varphi \to \psi) \to (\varphi \to \chi))}\,\texttt{mp}$$

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
```

```
wff φ
```

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wps $f wff ψ $.
```

```
wff φ
wff ψ
```

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wch $f wff χ $.
```

wff φ
wff ψ
wff χ

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
wff φ
wff ψ
wff χ
```

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
wff φ
wff ( ψ -> χ )
```

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
wff φ
wff ( ψ -> χ )
```

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
wff ( φ -> ( ψ -> χ ) )
```

Vérifier une preuve écrite en Metamath

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
wff ( φ -> ( ψ -> χ ) )
wff ( φ -> ψ )
```

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

wff (φ -> (ψ -> χ))
wff (φ -> ψ)
wff (φ -> χ)

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
wff ( φ -> ( ψ -> χ ) )
wff ( φ -> ψ )
wff ( φ -> χ )
```

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

wff (φ -> (ψ -> χ))

wff ((φ -> ψ) -> (φ -> χ))

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wff ( φ -> ( ψ -> χ ) )
wff ( ( φ -> ψ ) -> ( φ -> χ ) )
|- ( φ -> (ψ -> χ) )
```

Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wch $f wff χ $.
a2  $a |-
  ( ( φ -> ( ψ -> χ ) ) ->
  ( (φ -> ψ) -> (φ -> χ) ) )
$.
```

```
wff ( φ -> ( ψ -> χ ) )
wff ( ( φ -> ψ ) -> ( φ -> χ ) )
|- ( φ -> (ψ -> χ) )
|- ( ( φ -> (ψ -> χ) ) -> ( (φ -> ψ) -> (φ -> χ) ) )
```

Vérifier une preuve écrite en Metamath

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
min $e |- φ $.
maj $e |- ( φ -> ψ ) $.
mp  $a |- ψ $.
```

```
wff ( φ -> ( ψ -> χ ) )
wff ( ( φ -> ψ ) -> ( φ -> χ ) )
|- ( φ -> (ψ -> χ) )
|- ( ( φ -> (ψ -> χ) ) -> ( (φ -> ψ) -> (φ -> χ) ) )
```

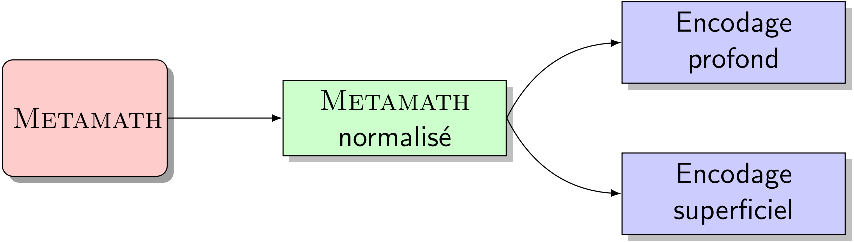
Vérifier une preuve écrite en METAMATH

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

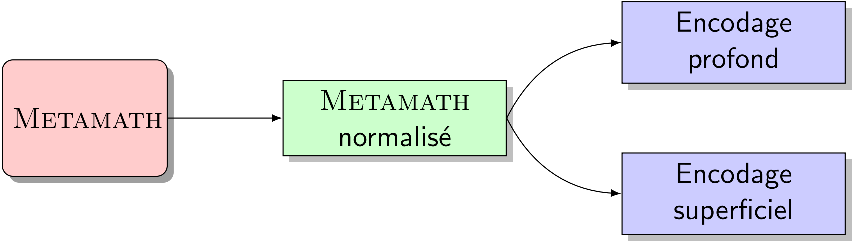
```
wph $f wff φ $.
wps $f wff ψ $.
min $e |- φ $.
maj $e |- ( φ -> ψ ) $.
mp  $a |- ψ $.
```

```
|- ( ( φ -> ψ ) -> ( φ -> χ ) )
```

Contributions

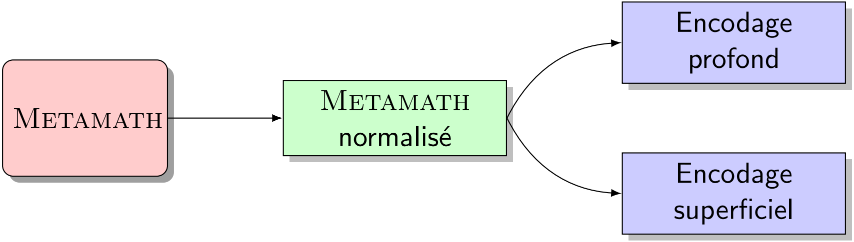


Contributions



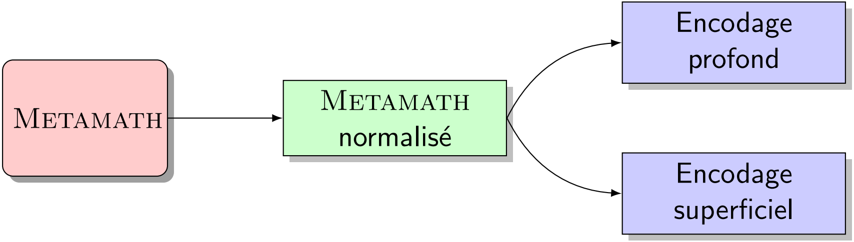
- Grammaire formelle du langage de Metamath

Contributions



- Grammaire formelle du langage de METAMATH
- Formalisation papier de l'étape de normalisation

Contributions



- Grammaire formelle du langage de METAMATH
- Formalisation papier de l'étape de normalisation
- Un encodage profond et un encodage superficiel en DEDUKTI

Encodage profond - Traduire une axiomatisation

- Traduire les constantes :

```
symbol token : TYPE;
```

```
$c ( ) wff -> |- $.
```

```
constant symbol PL : token;
constant symbol PR : token;
constant symbol wff : token;
constant symbol -> : token;
constant symbol taquet : token;
```

Encodage profond - Traduire une axiomatisation

- Traduire les constantes :

```
symbol token : TYPE;
```

- Traduire les énoncés :

```
symbol ++ : token → token → token; // En notation infix
rule ($a ++ $l) ++ $m ↪ $a ++ ($l ++ $m);

injective symbol # : token → token → token; // En notation infix
```

```
${ $v φ ψ $.
   wph1 $f wff φ $.
   wps1 $f wff ψ $.
   wi $a wff ( φ -> ψ ) $.  $}
```

$$\frac{\text{wff} \ \# \ \varphi \qquad \text{wff} \ \# \ \psi}{\text{wff} \ \# \ \text{PL} \ {+}{+} \ \varphi \ {+}{+} \ {-}{>} \ {+}{+} \ \psi \ {+}{+} \ \text{PR}}$$

Encodage profond - Traduire une axiomatisation

- Traduire les constantes :

```
symbol token : TYPE;
```

- Traduire les énoncés :

```
symbol ++ : token → token → token; // En notation infix
rule ($a ++ $l) ++ $m ↪ $a ++ ($l ++ $m);

injective symbol # : token → token → token; // En notation infix
```

- Traduire les méta-opérateurs :

```
symbol ⇒MM : token → token → token ; // En notation infix
```

```
${ $v φ ψ $.
   wph1 $f wff φ $.
   wps1 $f wff ψ $.
   wi $a wff ( φ -> ψ ) $.  $}
```

```
( wff # φ ) ⇒MM ( wff # ψ ) ⇒MM
( wff # PL ++ φ ++ -> ++ ψ ++ PR )
```

Encodage profond - Traduire une axiomatisation

- Traduire les constantes :

```
symbol token : TYPE;
```

- Traduire les énoncés :

```
symbol ++ : token → token → token; // En notation infix
rule ($a ++ $l) ++ $m ↪ $a ++ ($l ++ $m);

injective symbol # : token → token → token; // En notation infix
```

- Traduire les méta-opérateurs :

```
symbol ⇒_MM : token → token → token ; // En notation infix
symbol ∀_MM : (token → token) → token ;
```

```
${ $v φ ψ $.
   wph1 $f wff φ $.
   wps1 $f wff ψ $.
   wi $a wff ( φ -> ψ ) $.  $}
```

```
   (∀_MM(λ φ, ∀_MM(λ ψ,
( wff # φ ) ⇒_MM( wff # ψ ) ⇒_MM
( wff # PL ++ φ ++ -> ++ ψ ++ PR ))))
```

Encodage profond - Traduire une axiomatisation

- Traduire les constantes :

```
symbol token : TYPE;
```

- Traduire les énoncés :

```
symbol ++ : token → token → token; // En notation infix
rule ($a ++ $l) ++ $m ↪ $a ++ ($l ++ $m);

injective symbol # : token → token → token; // En notation infix
```

- Traduire les méta-opérateurs :

```
symbol ⇒MM: token → token → token ; // En notation infix
symbol ∀MM : (token → token) → token ;

injective symbol Prf : token → TYPE ;
```

```
${ $v φ ψ $.
   wph1 $f wff φ $.
   wps1 $f wff ψ $.
   wi $a wff ( φ -> ψ ) $.  $}
```

```
Prf (∀MM(λ φ, ∀MM(λ ψ,
   ( wff # φ ) ⇒MM( wff # ψ ) ⇒MM
   ( wff # PL ++ φ ++ -> ++ ψ ++ PR ))))
```

Encodage profond - Traduire une axiomatisation

- Traduire les constantes :

```
symbol token : TYPE;
```

- Traduire les énoncés :

```
symbol ++ : token → token → token; // En notation infix
rule ($a ++ $l) ++ $m ↪ $a ++ ($l ++ $m);

injective symbol # : token → token → token; // En notation infix
```

- Traduire les méta-opérateurs :

```
symbol ⇒MM: token → token → token ; // En notation infix
symbol ∀MM : (token → token) → token ;

injective symbol Prf : token → TYPE ;
```

```
${ $v φ ψ $.
   wph1 $f wff φ $.
   wps1 $f wff ψ $.
   wi $a wff ( φ -> ψ ) $.  $}
```

```
constant symbol wi :
  Prf (∀MM(λ φ, ∀MM(λ ψ,
    ( wff # φ ) ⇒MM( wff # ψ ) ⇒MM
    ( wff # PL ++ φ ++ -> ++ ψ ++ PR )))) ;
```

Encodage profond - Traduire une axiomatisation

- Traduire les constantes :

```
symbol token : TYPE;
```

- Traduire les énoncés :

```
symbol ++ : token → token → token; // En notation infix
rule ($a ++ $l) ++ $m ↪ $a ++ ($l ++ $m);

injective symbol # : token → token → token; // En notation infix
```

- Traduire les méta-opérateurs :

```
symbol ⇒MM: token → token → token ; // En notation infix
symbol ∀MM : (token → token) → token ;

injective symbol Prf : token → TYPE ;

rule Prf (∀MM(λ a, $b.[a]) ) ↪ Π a, Prf $b.[a];
rule Prf ($a ⇒MM$b) ↪ Prf $a → Prf $b ;
```

```
${ $v φ ψ $.
   wph1 $f wff φ $.
   wps1 $f wff ψ $.
   wi $a wff ( φ -> ψ ) $.  $}
```

```
constant symbol wi :
  Prf (∀MM(λ φ, ∀MM(λ ψ,
    ( wff # φ ) ⇒MM( wff # ψ ) ⇒MM
    ( wff # PL ++ φ ++ -> ++ ψ ++ PR )))) ;
```

Encodage profond - Traduire une axiomatisation

- Traduire les constantes :

```
symbol token : TYPE;
```

- Traduire les énoncés :

```
symbol ++ : token → token → token; // En notation infix
rule ($a ++ $l) ++ $m ↪ $a ++ ($l ++ $m);

injective symbol # : token → token → token; // En notation infix
```

- Traduire les méta-opérateurs :

```
symbol ⇒_MM: token → token → token ; // En notation infix
symbol ∀_MM : (token → token) → token ;

injective symbol Prf : token → TYPE ;

rule Prf (∀_MM(λ a, $b.[a]) ) ↪ Π a, Prf $b.[a];
rule Prf ($a ⇒_MM$b) ↪ Prf $a → Prf $b ;
```

```
${ $v φ ψ $.
   wph1 $f wff φ $.
   wps1 $f wff ψ $.
   wi $a wff ( φ -> ψ ) $.  $}
```

```
constant symbol wi :
  Π φ, Π ψ,
    Prf ( wff # φ ) → Prf ( wff # ψ ) →
    Prf ( wff # PL ++ φ ++ -> ++ ψ ++ PR );
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

Idée clef : Élément de la pile = label

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
```

```
wph
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wps $f wff ψ $.
```

```
wph
wps
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wch $f wff χ $.
```

```
wph
wps
wch
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
wph
wps
wch
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
wph
(wi _ _ wps wch)
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
wph
(wi _ _ wps wch)
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
(wi _ _ wph (wi _ _ wps wch))
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
(wi _ _ wph (wi _ _ wps wch))
(wi _ _ wph wps)
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
(wi _ _ wph (wi _ _ wps wch))
(wi _ _ wph wps)
(wi _ _ wph wch)
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
(wi _ _ wph (wi _ _ wps wch))
(wi _ _ wph wps)
(wi _ _ wph wch)
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
(wi _ _ wph (wi _ _ wps wch))
(wi _ _ (wi _ _ wph wps) (wi _ _ wph wch))
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
(wi _ _ wph (wi _ _ wps wch))
(wi _ _ (wi _ _ wph wps) (wi _ _ wph wch))
hyp
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wch $f wff χ $.
a2  $a |-
  ( ( φ -> ( ψ -> χ ) ) ->
  ( (φ -> ψ) -> (φ -> χ) ) )
$.
```

```
(wi _ _ wph (wi _ _ wps wch))
(wi _ _ (wi _ _ wph wps) (wi _ _ wph wch))
hyp
(a2 _ _ _ wph wps wch)
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
min $e |- φ $.
maj $e |- ( φ -> ψ ) $.
mp  $a |- ψ $.
```

```
(wi _ _ wph (wi _ _ wps wch))
(wi _ _ (wi _ _ wph wps) (wi _ _ wph wch))
hyp
(a2 _ _ _ wph wps wch)
```

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
min $e |- φ $.
maj $e |- ( φ -> ψ ) $.
mp  $a |- ψ $.
```

mp _ _ α β γ ι

Encodage profond - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
min $e |- φ $.
maj $e |- ( φ -> ψ ) $.
mp  $a |- ψ $.
```

```
symbol a2i :
  Prf (∀MM(λ φ, ∀MM(λ ψ, ∀MM(λ χ,
      ( wff # φ ) ⇒MM( wff # ψ ) ⇒MM( wff # χ ) ⇒MM
      (taquet # PL ++ φ ++ -> ++ PL ++ ψ ++ -> ++ χ ++ PR ++ PR) ⇒MM
      (taquet # PL ++ PL ++ φ ++ -> ++ ψ ++ PR ++ -> ++
                        PL ++ φ ++ -> ++ χ ++ PR ++ PR ) ) ) ) ) :=
  λ φ ψ χ wph wps wch hyp,
    mp _ _ (wi _ _ wph (wi _ _ wps wch))
           (wi _ _ (wi _ _ wph wps) (wi _ _ wph wch))
           hyp
           (a2 _ _ _ wph wps wch);
```

Preuves avec variables libres

```
$c ( ) wff -> |-  $.                  $( Déclaration de 5 constantes (tokens) $)

$v φ ψ χ $.                           $( Déclaration de 3 variables $)

wph $f wff φ $.                       $( Typage des variables $)
wps $f wff ψ $.
wch $f wff χ $.

${  wi $a wff ( φ -> ψ ) $.  $}

${  a2 $a |- ((φ -> (ψ -> χ)) -> ((φ -> ψ) -> (φ -> χ))) $.  $}

${  min $e |- φ $.
    maj $e |- ( φ -> ψ ) $.
    mp  $a |- ψ $.  $}
```

Preuves avec variables libres

```
$c ( ) wff -> |- $.                         $( Déclaration de 5 constantes (tokens) $)

$v φ ψ χ $.                                 $( Déclaration de 3 variables $)

wph $f wff φ $.                             $( Typage des variables $)
wps $f wff ψ $.
wch $f wff χ $.

${ wi $a wff ( φ -> ψ ) $.  $}

${ a2 $a |- ((φ -> (ψ -> χ)) -> ((φ -> ψ) -> (φ -> χ))) $.  $}

${ min $e |- φ $.
   maj $e |- ( φ -> ψ ) $.
   mp  $a |- ψ $.  $}
```

- Génération d'un témoin par type : `symbol` wff-witness : Prf (wff # t) ;

Preuves avec variables libres

```
$c ( ) wff -> |-  $.                  $( Déclaration de 5 constantes (tokens) $)

$v φ ψ χ $.                           $( Déclaration de 3 variables $)

wph $f wff φ $.                       $( Typage des variables $)
wps $f wff ψ $.
wch $f wff χ $.

${ wi $a wff ( φ -> ψ ) $.  $}

${ a2 $a |- ((φ -> (ψ -> χ)) -> ((φ -> ψ) -> (φ -> χ))) $.  $}

${ min $e |- φ $.
   maj $e |- ( φ -> ψ ) $.
   mp  $a |- ψ $.  $}
```

- Génération d'un témoin par type : `symbol` wff-witness : Prf (wff # t) ;
- Découverte : Tous les types sont habités.

Traduire la bibliothèque standard de METAMATH

	demo0.mm	miu.mm	peano.mm	big-unifier.mm	hol.mm	ql.mm	nf.mm	iset.mm	set.mm
Taille du fichier	1.4K	4.6K	28K	39K	231K	1.8M	8.3M	11M	273M
Nombre de :									
- déclarations	19	27	116	29	2 768	936	24 967	30 813	196 976
- axiomes	7	10	48	4	77	71	363	467	2711
- preuve(s)	1	1	0	2	1 138	138	5 966	8 990	38 766
Pourcentage traduit avec l'encodage :									

Traduire la bibliothèque standard de Metamath

	`demo0.mm`	`miu.mm`	`peano.mm`	`big-unifier.mm`	`hol.mm`	`ql.mm`	`nf.mm`	`iset.mm`	`set.mm`
Taille du fichier	1.4K	4.6K	28K	39K	231K	1.8M	8.3M	11M	273M
Nombre de :									
- déclarations	19	27	116	29	2 768	936	24 967	30 813	196 976
- axiomes	7	10	48	4	77	71	363	467	2711
- preuve(s)	1	1	0	2	1 138	138	5 966	8 990	38 766
Pourcentage traduit avec l'encodage :									
- profond	100	0	100	100	100	100	100	100	100*

Traduire la bibliothèque standard de Metamath

	demo0.mm	miu.mm	peano.mm	big-unifier.mm	hol.mm	ql.mm	nf.mm	iset.mm	set.mm
Taille du fichier	1.4K	4.6K	28K	39K	231K	1.8M	8.3M	11M	273M
Nombre de :									
- déclarations	19	27	116	29	2 768	936	24 967	30 813	196 976
- axiomes	7	10	48	4	77	71	363	467	2711
- preuve(s)	1	1	0	2	1 138	138	5 966	8 990	38 766
Pourcentage traduit avec l'encodage :									
- sans témoin	100	0	100	100	95,87	100	79,07	80,29	65,26*
- profond	100	0	100	100	100	100	100	100	100*

Et maintenant ?

Nous avons vu que

```
${  $v φ  ψ $.
    wph $f wff  φ $.
    wps $f wff  ψ $.
    min $e |-  φ $.
    maj $e |-  ( φ  ->  ψ  ) $.
    mp  $a |- ψ $.  $}
```

est traduit en DEDUKTI ainsi :

```
constant symbol mp :
 Prf (∀_MM(λ φ, ∀_MM(λ ψ,
   ( wff # φ ) ⇒_MM( wff # ψ ) ⇒_MM
   ( taquet # φ ) ⇒_MM
   ( taquet # PL ++ φ ++ –> ++ ψ ++ PR ) ⇒_MM
   ( taquet # ψ ) ))) ;
```

Et maintenant ?

Nous avons vu que

```
${  $v φ  ψ $.
    wph $f wff  φ $.
    wps $f wff  ψ $.
    min $e |-  φ $.
    maj $e |-  ( φ  -> ψ  ) $.
    mp  $a |- ψ $.  $}
```

est traduit en DEDUKTI ainsi :

```
constant symbol mp :
 Prf (∀MM(λ φ, ∀MM(λ ψ,
   ( wff # φ ) ⇒MM( wff # ψ ) ⇒MM
   ( taquet # φ ) ⇒MM
   ( taquet # PL ++ φ ++ -> ++ ψ ++ PR ) ⇒MM
   ( taquet # ψ ) ))) ;
```

A présent, voyons comment obtenir :

```
constant symbol mp : Π (φ : wff), Π (ψ : wff), |- φ → |- (φ -> ψ) → |- ψ ;
```

Encodage superficiel - Traduire une axiomatisation

```
$c ( ) wff -> |- $.
```

```
${ $v φ ψ $.
   wph1 $f wff φ $.
   wps1 $f wff ψ $.
   wi $a wff ( φ -> ψ ) $.  $}

${ $v φ ψ $.
   wph3 $f wff φ $.
   wps3 $f wff ψ $.
   min $e |- φ $.
   maj $e |- ( φ -> ψ ) $.
   mp  $a |- ψ $.  $}
```

Encodage superficiel - Traduire une axiomatisation

```
$c ( ) wff -> |- $.
```

```
symbol wff : TYPE ;
symbol -> : ? ;
symbol |- : ? ;
```

```
${ $v φ ψ $.
   wph1 $f wff φ $.
   wps1 $f wff ψ $.
   wi $a wff ( φ -> ψ ) $.  $}

${ $v φ ψ $.
   wph3 $f wff φ $.
   wps3 $f wff ψ $.
   min $e |- φ $.
   maj $e |- ( φ -> ψ ) $.
   mp  $a |- ψ $.  $}
```

Encodage superficiel - Traduire une axiomatisation

```
$c ( ) wff -> |- $.
```

symbol wff : TYPE ;
symbol |- : ? ;

```
${  $v φ ψ $.
    wph1 $f wff φ $.
    wps1 $f wff ψ $.
    wi $a wff ( φ -> ψ ) $.  $}
```

symbol -> : wff → wff → wff ;

```
${  $v φ ψ $.
    wph3 $f wff φ $.
    wps3 $f wff ψ $.
    min $e |- φ $.
    maj $e |- ( φ -> ψ ) $.
    mp  $a |- ψ $.  $}
```

Encodage superficiel - Traduire une axiomatisation

```
$c ( ) wff -> |- $.
```

symbol wff : TYPE ;
symbol |- : ? ;

```
${ $v φ ψ $.
   wph1 $f wff φ $.
   wps1 $f wff ψ $.
   wi $a wff ( φ -> ψ ) $.  $}
```

symbol -> : wff → wff → wff ;

```
${ $v φ ψ $.
   wph3 $f wff φ $.
   wps3 $f wff ψ $.
   min $e |- φ $.
   maj $e |- ( φ -> ψ ) $.
   mp  $a |- ψ $.  $}
```

symbol mp : Π (φ : wff), Π (ψ : wff),
|- φ → |- (φ -> ψ) → |- ψ ;

Encodage superficiel - Traduire une axiomatisation

```
$c ( ) wff -> |- $.
```

symbol wff : TYPE ;
symbol |- : ? ;

```
${  $v φ ψ $.
    wph1 $f wff φ $.
    wps1 $f wff ψ $.
    wi $a wff ( φ -> ψ ) $.  $}
```

symbol -> : wff → wff → wff ;

```
${  $v φ ψ $.
    wph3 $f wff φ $.
    wps3 $f wff ψ $.
    min $e |- φ $.
    maj $e |- ( φ -> ψ ) $.
    mp  $a |- ψ $.  $}
```

symbol mp : Π (φ : wff), Π (ψ : wff),
|- φ → |- (φ -> ψ) → |- ψ ;

- **Problème 1** : Comment inférer le type d'un opérateur ?
- **Problème 2** : Comment trouver la précédence d'un opérateur ?

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

Idée clef : Élément de la pile = label ou liste de tokens

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
```

φ

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wps $f wff ψ $.
```

$$\begin{array}{c}\varphi\\ \psi\end{array}$$

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wch $f wff χ $.
```

φ
ψ
χ

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

φ

ψ

χ

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

φ

(ψ -> χ)

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

φ

(ψ -> χ)

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

(φ -> (ψ -> χ))

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
( φ -> ( ψ -> χ ) )
( φ -> ψ )
```

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
( φ -> ( ψ -> χ ) )
( φ -> ψ )
( φ -> χ )
```

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

```
( φ -> ( ψ -> χ ) )
( φ -> ψ )
( φ -> χ )
```

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wi  $a wff ( φ -> ψ ) $.
```

$(\varphi \texttt{ -> } (\psi \texttt{ -> } \chi))$

$((\varphi \texttt{ -> } \psi) \texttt{ -> } (\varphi \texttt{ -> } \chi))$

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

(φ -> (ψ -> χ))
((φ -> ψ) -> (φ -> χ))
hyp

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
wch $f wff χ $.
a2  $a |-
 ( ( φ -> ( ψ -> χ ) ) ->
 ( (φ -> ψ) -> (φ -> χ) ) )
$.
```

(φ -> (ψ -> χ))
((φ -> ψ) -> (φ -> χ))
hyp
a2 φ ψ χ

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
min $e |- φ $.
maj $e |- ( φ -> ψ ) $.
mp  $a |- ψ $.
```

```
( φ -> ( ψ -> χ ) )
( ( φ -> ψ ) -> ( φ -> χ ) )
hyp
a2 φ ψ χ
```

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
min $e |- φ $.
maj $e |- ( φ -> ψ ) $.
mp  $a |- ψ $.
```

mp α β hyp (a2 φ ψ χ)

$$\alpha \triangleq \texttt{(}\ \varphi\ \texttt{->}\ \texttt{(}\ \psi\ \texttt{->}\ \chi\ \texttt{)}\ \texttt{)}$$
$$\beta \triangleq \texttt{(}\ \texttt{(}\ \varphi\ \texttt{->}\ \psi\ \texttt{)}\ \texttt{->}\ \texttt{(}\ \varphi\ \texttt{->}\ \chi\ \texttt{)}\ \texttt{)}$$

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
min $e |- φ $.
maj $e |- ( φ -> ψ ) $.
mp  $a |- ψ $.
```

$$\lambda\ \varphi\ \psi\ \chi\ \texttt{hyp},\ \texttt{mp}\ \alpha\ \beta\ \texttt{hyp}\ (\texttt{a2}\ \varphi\ \psi\ \chi)$$

$$\alpha \triangleq \texttt{(}\ \varphi\ \texttt{->}\ \texttt{(}\ \psi\ \texttt{->}\ \chi\ \texttt{)}\ \texttt{)}$$
$$\beta \triangleq \texttt{(}\ \texttt{(}\ \varphi\ \texttt{->}\ \psi\ \texttt{)}\ \texttt{->}\ \texttt{(}\ \varphi\ \texttt{->}\ \chi\ \texttt{)}\ \texttt{)}$$

Encodage superficiel - Traduire une preuve

```
hyp $e |- (φ -> (ψ -> χ)) $.
a2i $p |- ((φ -> ψ) -> (φ -> χ))
$=
  wph wps wch wi wi
  wph wps wi wph wch wi wi
  hyp
  wph wps wch a2
  mp  $.
```

```
wph $f wff φ $.
wps $f wff ψ $.
min $e |- φ $.
maj $e |- ( φ -> ψ ) $.
mp  $a |- ψ $.
```

```
symbol a2i :  Π (φ :  wff), Π (ψ :  wff), Π (χ :  wff),
    |- (φ -> (ψ -> χ)) → |- ((φ -> ψ) -> (φ -> χ)) :=
    λ φ ψ χ hyp,
        mp ( φ -> ( ψ -> χ ) )
           ( ( φ -> ψ ) -> ( φ -> χ ) )
           hyp
           (a2 φ ψ χ) ;
```

Problèmes actuels

- **Problème 1** : Comment inférer le type d'un opérateur ?
- **Problème 2** : Comment trouver la précédence d'un opérateur ?

Problèmes actuels

- **Problème 1** : Comment inférer le type d'un opérateur ?
- **Problème 2** : Comment trouver la précédence d'un opérateur ?
  - → Solution partielle : Ajout d'annotations (au format JSON)

```
{ name_metamath : "|-",
  name_dedukti : "taquet",
  type : "wff → TYPE",
  mixfix : "prefix",
  precedence : 40 }
```

$\Downarrow$

```
constant symbol taquet : wff → TYPE ;
```

Problèmes actuels

- **Problème 1** : Comment inférer le type d'un opérateur ?
- **Problème 2** : Comment trouver la précédence d'un opérateur ?
  → Solution partielle : Ajout d'annotations (au format JSON)

- **Problème 3** : Comment détecter et traduire
  - le sous-typage ?

```
xnat $f nat x $.
xfloat $a float x $.
```

Problèmes actuels

- **Problème 1** : Comment inférer le type d'un opérateur ?
- **Problème 2** : Comment trouver la précédence d'un opérateur ?

  → Solution partielle : Ajout d'annotations (au format JSON)

- **Problème 3** : Comment détecter et traduire
  - le sous-typage ? → Considérer une injection

```
xnat $f nat x $.
xfloat $a float x $.
```

Problèmes actuels

- **Problème 1** : Comment inférer le type d'un opérateur ?
- **Problème 2** : Comment trouver la précédence d'un opérateur ?

  → Solution partielle : Ajout d'annotations (au format JSON)

- **Problème 3** : Comment détecter et traduire
  - le sous-typage ?
  - la surcharge ?

Problèmes actuels

- **Problème 1** : Comment inférer le type d'un opérateur ?
- **Problème 2** : Comment trouver la précédence d'un opérateur ?

  → Solution partielle : Ajout d'annotations (au format JSON)

- **Problème 3** : Comment détecter et traduire
  - le sous-typage ?
  - la surcharge ?
  - etc.

Traduire la bibliothèque standard de METAMATH

	`demo0.mm`	`miu.mm`	`peano.mm`	`big-unifier.mm`	`hol.mm`	`ql.mm`	`nf.mm`	`iset.mm`	`set.mm`
Taille du fichier	1.4K	4.6K	28K	39K	231K	1.8M	8.3M	11M	273M
Nombre de :									
- déclarations	19	27	116	29	2 768	936	24 967	30 813	196 976
- axiomes	7	10	48	4	77	71	363	467	2711
- preuve(s)	1	1	0	2	1 138	138	5 966	8 990	38 766
Pourcentage traduit avec l'encodage :									
- profond	100	0	100	100	100	100	100	100	100*

Traduire la bibliothèque standard de METAMATH

	`demo0.mm`	`miu.mm`	`peano.mm`	`big-unifier.mm`	`hol.mm`	`ql.mm`	`nf.mm`	`iset.mm`	`set.mm`
Taille du fichier	1.4K	4.6K	28K	39K	231K	1.8M	8.3M	11M	273M
Nombre de :									
- déclarations	19	27	116	29	2 768	936	24 967	30 813	196 976
- axiomes	7	10	48	4	77	71	363	467	2711
- preuve(s)	1	1	0	2	1 138	138	5 966	8 990	38 766
Pourcentage traduit avec l'encodage :									
- profond	100	0	100	100	100	100	100	100	100*
- superficiel	100	0	0	100	0	0+	0+	0+	2,52+

Conclusion

- Caractéristiques de Math$\sim$:
  - Écrit en OCaml
  - Accessible ici :
    `https://gitlab.com/semantiko/mm2dk/translator`

Conclusion

- Caractéristiques de Math$\sim$:
  - Écrit en OCaml
  - Accessible ici :
    `https://gitlab.com/semantiko/mm2dk/translator`

- Problèmes similaires aux autres traducteurs de X vers Dedukti :
  - Pas de formalisation de X
  - Fichiers générés trop longs à vérifier

Conclusion

- Caractéristiques de Math$\sim$:
  - Écrit en OCaml
  - Accessible ici : https://gitlab.com/semantiko/mm2dk/translator
- Problèmes similaires aux autres traducteurs de X vers Dedukti :
  - Pas de formalisation de X
  - Fichiers générés trop longs à vérifier
- Particularités de Math$\sim$:
  - Choix du format d'entrée
  - Deux encodages complètement différents
  - Problème pour détecter certaines fonctionnalités encodées dans Metamath

Conclusion

- Caractéristiques de Math$\sim$:
  - Écrit en OCaml
  - Accessible ici :
    `https://gitlab.com/semantiko/mm2dk/translator`

- Problèmes similaires aux autres traducteurs de X vers Dedukti :
  - Pas de formalisation de X
  - Fichiers générés trop longs à vérifier

- Particularités de Math$\sim$:
  - Choix du format d'entrée
  - Deux encodages complètement différents
  - Problème pour détecter certaines fonctionnalités encodées dans Metamath

- Limite de Math$\sim$:
  - Fonctionnalité non complètement prise en charge : `$d`