Rechecking KProver proof objects into Dedukti

Amélie LEDEIN

in collaboration with Elliot BUTTE

Labex DigiCosme université PARIS-SACLAY

Inria

LMF Laboratoire Méthodes Formelles

Introduction

Goal: Rechecking KProver proof objects into Dedukti

Introduction

Goal: Rechecking KProver proof objects into Dedukti

$\mathbb{K}$

- Semantical framework
  - to define formal semantics of programming languages
  - to automatically generate tools from these semantics

Dedukti

- Logical framework
  - to encode various logics
  - to allow interoperability of proofs between different formal tools

Goal: Rechecking KProver proof objects into Dedukti

$\mathbb{K}$

- Semantical framework
  - to define formal semantics of programming languages
  - to automatically generate tools from these semantics

  Ex: the ATP KProver

Dedukti

- Logical framework
  - to encode various logics
  - to allow interoperability of proofs between different formal tools

Introduction

Goal: Rechecking KProver proof objects into Dedukti

$\mathbb{K}$

- Semantical framework
  - to define formal semantics of programming languages
  - to automatically generate tools from these semantics

  Ex: the ATP KPROVER

- Based on MATCHING LOGIC

  → an untyped 1st order logic with fixpoints and a "next" operator

Dedukti

- Logical framework
  - to encode various logics
  - to allow interoperability of proofs between different formal tools

- Based on $\lambda\Pi$-CALCULUS MODULO THEORY

  → a λ-calculus with dependent types, and extended with rewriting rules

Overview of $\mathbb{K}$

Two steps to define a $\mathbb{K}$ semantics:

- **Syntax**
- **Semantics**

Overview of $\mathbb{K}$

Two steps to define a $\mathbb{K}$ semantics:

- **Syntax**
  - **BNF grammar**
- **Semantics**

Overview of $\mathbb{K}$

Two steps to define a $\mathbb{K}$ semantics:

- **Syntax**
  - **BNF grammar**
- **Semantics**
  - **Configuration** = State of the program
    Example: $\langle\langle\ x + 17\ \rangle_k\ \langle\ x \mapsto 25\ \rangle_{env}\rangle$
  - **Rewriting rule** on configurations ($\sim$ transition system)

Overview of $\mathbb{K}$

$\langle$ `x = 1 ; while 0 < x { x-- } ;` $\rangle_k$
$\langle$ `nil` $\rangle_{env}$

$\langle$ `while 0 < x { x-- } ;` $\rangle_k$
$\langle$ `x` $\mapsto$ `1` $\rangle_{env}$

$\langle$ `while 0 < x { x-- } ;` $\rangle_k$
$\langle$ `x` $\mapsto$ `42` $\rangle_{env}$

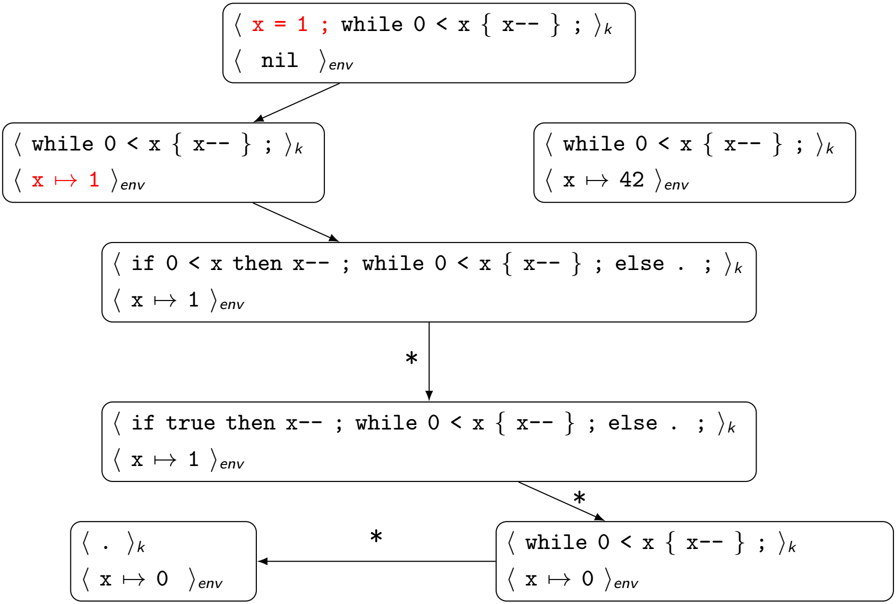
$\langle$ `if 0 < x then x-- ; while 0 < x { x-- } ; else . ;` $\rangle_k$
$\langle$ `x` $\mapsto$ `1` $\rangle_{env}$

$\langle$ `if true then x-- ; while 0 < x { x-- } ; else . ;` $\rangle_k$
$\langle$ `x` $\mapsto$ `1` $\rangle_{env}$

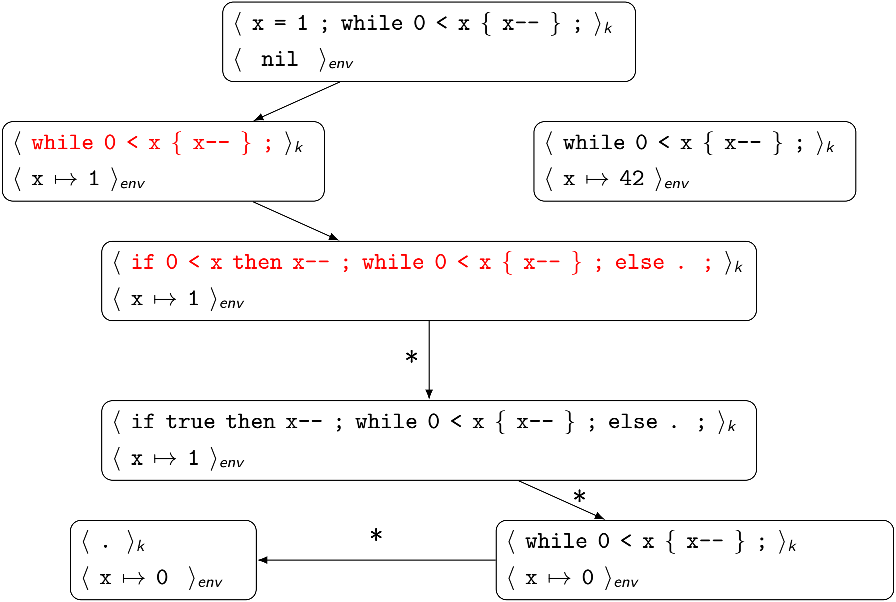
$\langle$ `.` $\rangle_k$
$\langle$ `x` $\mapsto$ `0` $\rangle_{env}$

$\langle$ `while 0 < x { x-- } ;` $\rangle_k$
$\langle$ `x` $\mapsto$ `0` $\rangle_{env}$

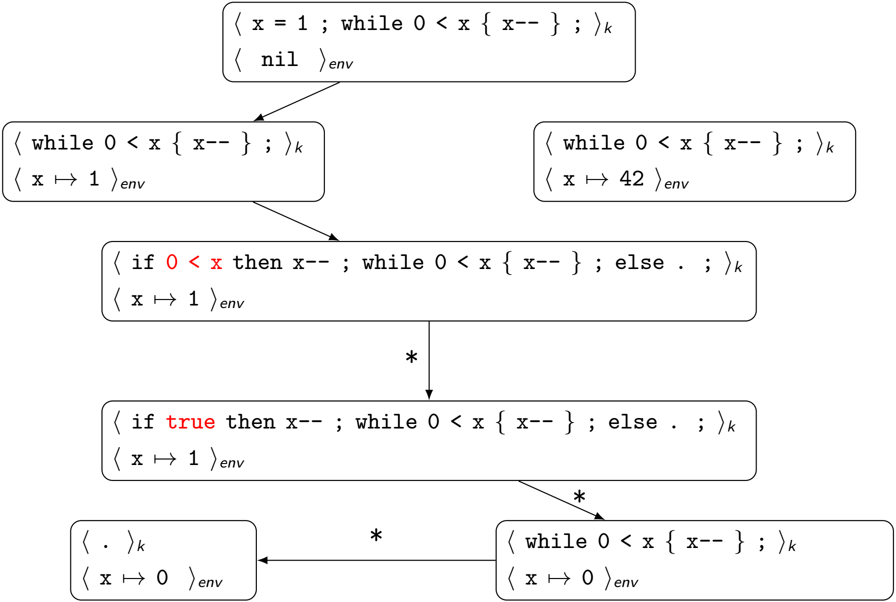
Overview of $\mathbb{K}$



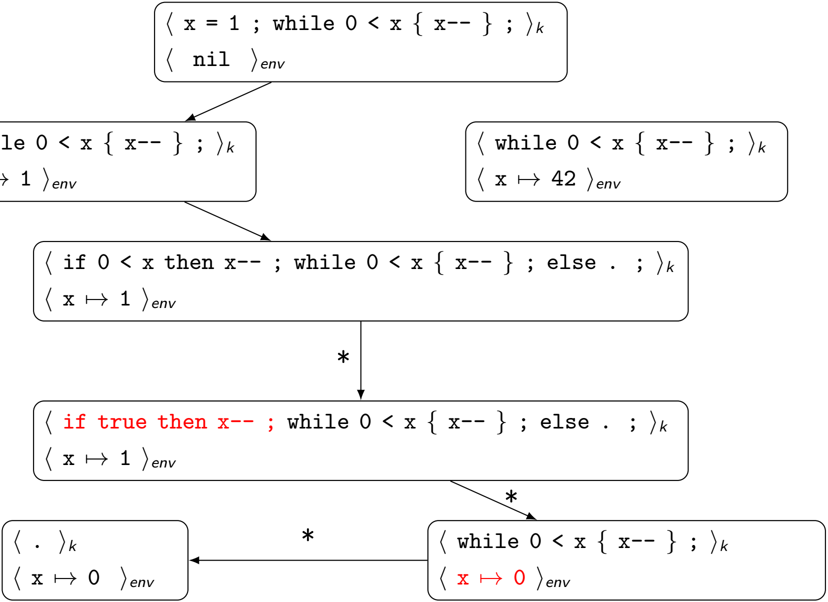
Overview of $\mathbb{K}$



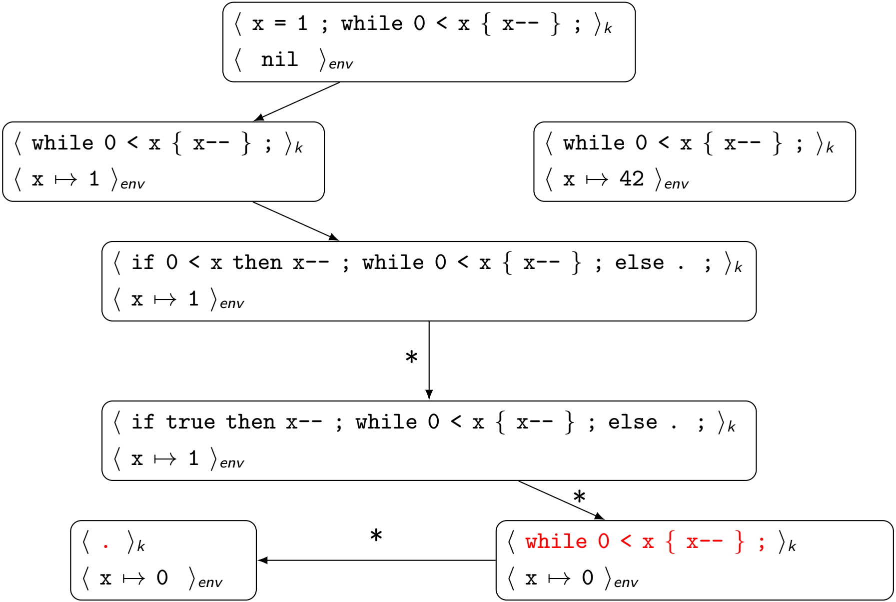
Overview of 𝕂



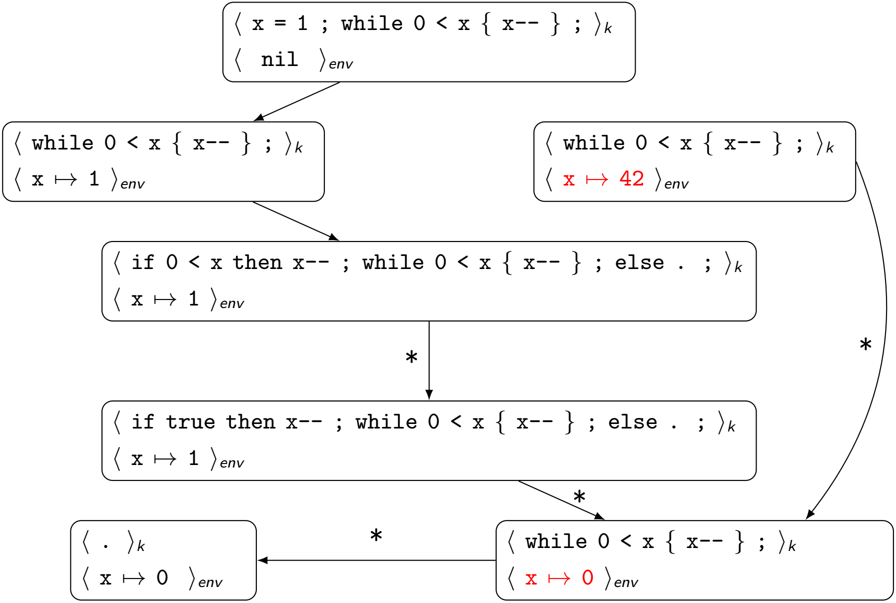
Overview of $\mathbb{K}$



Overview of $\mathbb{K}$



Overview of $\mathbb{K}$



Overview of KProver

- Parametrized by a $\mathbb{K}$ semantics
- Reachability property $\varphi \rightsquigarrow \varphi'$:

  During the execution of a program,
  if φ is **matched**, then φ' will be **matched** later on in a finite number of steps, or there is divergence.

Overview of KProver

- Parametrized by a $\mathbb{K}$ semantics
- Reachability property $\varphi \rightsquigarrow \varphi'$:

  During the execution of a program,
  if φ is **matched**, then φ' will be **matched** later on in a finite number of steps, or there is divergence.

- Example:

  $(N \geq 0) \wedge (S \geq 0) \wedge$
  $\langle\langle$ `while` $0 < \mathsf{n}$ do $\{ \mathsf{s} = \mathsf{s} + \mathsf{n}\ ;\ \mathsf{n} = \mathsf{n} - 1;\ \} \rangle_k \langle\ \mathsf{n} \mapsto \mathsf{N},\ \mathsf{s} \mapsto \mathsf{S}\ \rangle_{env}\rangle$
  $\rightsquigarrow \langle\langle\ \ .\ \ \rangle_k \langle\ \mathsf{n} \mapsto 0,\ \mathsf{s} \mapsto \mathsf{S} + \frac{N*(N+1)}{2}\ \rangle_{env}\rangle$

Overview of KPROVER

- Parametrized by a $\mathbb{K}$ semantics
- Reachability property $\varphi \rightsquigarrow \varphi'$:

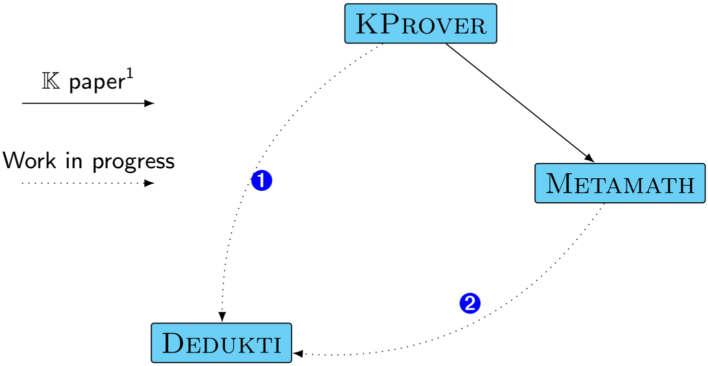
  During the execution of a program,
  if φ is **matched**, then φ' will be **matched** later on in a finite number of steps, or there is divergence.

- Example:

  $(N \geq 0) \wedge (S \geq 0) \wedge$
  $\langle\langle$ `while` $0 < \mathrm{n}$ do $\{ \mathrm{s} = \mathrm{s} + \mathrm{n}\ ;\ \mathrm{n} = \mathrm{n} - 1; \} \rangle_k \langle \mathrm{n} \mapsto \mathrm{N}, \mathrm{s} \mapsto \mathrm{S} \rangle_{env}\rangle$
  $\rightsquigarrow \langle\langle\ .\ \rangle_k \langle \mathrm{n} \mapsto 0, \mathrm{s} \mapsto \mathrm{S} + \frac{N*(N+1)}{2} \rangle_{env}\rangle$

→ **The generation of symbolic trace is not considered in the first version of the KProver with trace.**

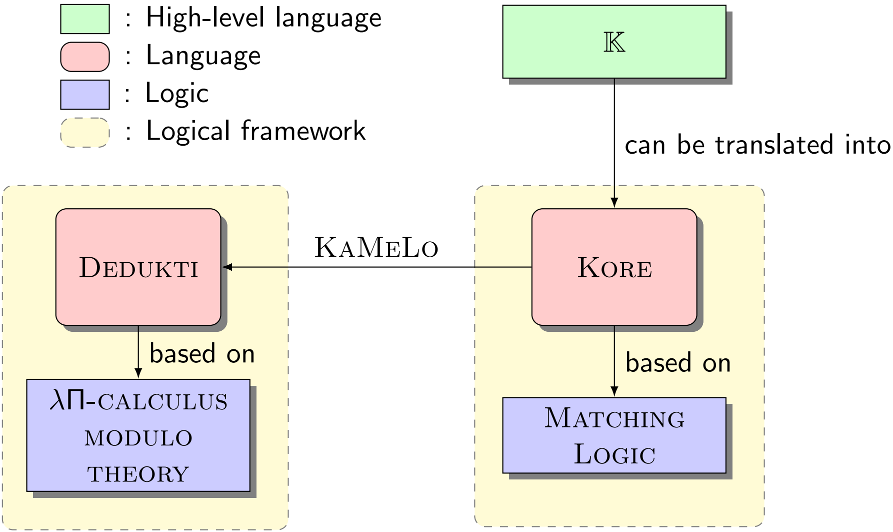
Two ways, two solutions



1. The direct approach
2. The approach via METAMATH

[1] X. Chen, Z. Lin, M.-T. Trinh, and G. Roșu. *Towards a Trustworthy Semantics-Based Language Framework via Proof Generation.* CAV'21.

Overview of the ecosystems



Methodology

Goal recheck: $\Gamma \vdash \varphi \rightsquigarrow \varphi'$ in DEDUKTI

Methodology

Goal recheck: $\Gamma \vdash \varphi \rightsquigarrow \varphi'$ in DEDUKTI

1. Encode MATCHING LOGIC into DEDUKTI. $= DK[ML]$

Methodology

Goal recheck: $\Gamma \vdash \varphi \rightsquigarrow \varphi'$ in DEDUKTI

1. Encode MATCHING LOGIC into DEDUKTI. $= DK[ML]$
   - MATCHING LOGIC patterns[2]
     $\varphi ::= x \mid X \mid \sigma \mid \varphi\ \varphi \mid \bot \mid \varphi \rightarrow \varphi \mid \exists x.\varphi \mid \mu X.\varphi$

[2] A pattern is interpreted as the set of elements that it matches.

Methodology

Goal recheck: $\Gamma \vdash \varphi \rightsquigarrow \varphi'$ in Dedukti

1. Encode Matching Logic into Dedukti. $= DK[ML]$
   - Matching Logic patterns[2]
     $\varphi ::= x \mid X \mid \sigma \mid \varphi\ \varphi \mid \bot \mid \varphi \rightarrow \varphi \mid \exists x.\varphi \mid \mu X.\varphi$
   - Matching Logic proof system

More details at Dedukti school!

[2] A pattern is interpreted as the set of elements that it matches.

Methodology

Goal recheck: $\Gamma \vdash \varphi \rightsquigarrow \varphi'$ in DEDUKTI

1. Encode MATCHING LOGIC into DEDUKTI. $= DK[ML]$
2. Translate $\mathbb{K}$ semantics, the claim and the trace into KORE.

Methodology

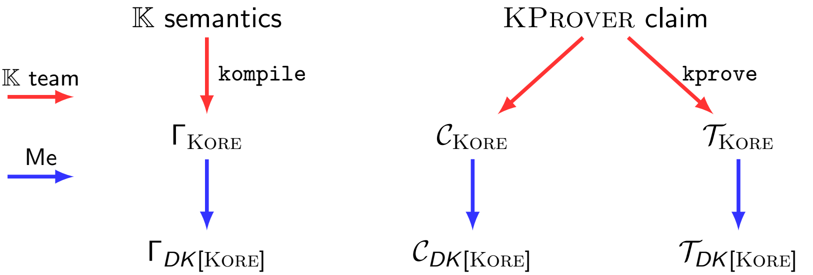
Goal recheck: $\Gamma \vdash \varphi \rightsquigarrow \varphi'$ in Dedukti

1. Encode Matching Logic into Dedukti. $= DK[ML]$
2. Translate $\mathbb{K}$ semantics, the claim and the trace into Kore.
3. Encode Kore into Dedukti. $= DK[\text{Kore}]$
   - → Kore is seen as a language translatable into the pattern of ML.
   - → Rewriting is used to go from $DK[\text{Kore}]$ to $DK[ML]$.

Methodology

Goal recheck: $\Gamma \vdash \varphi \rightsquigarrow \varphi'$ in DEDUKTI

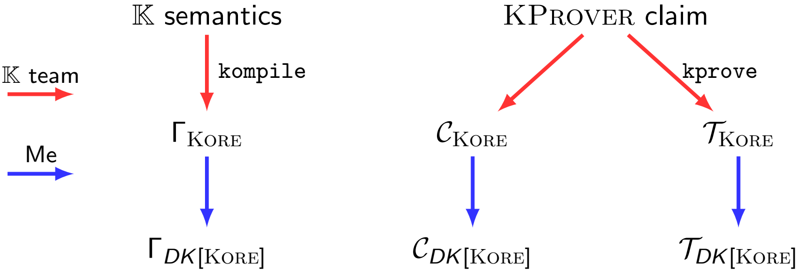
1. Encode MATCHING LOGIC into DEDUKTI. $= DK[ML]$
2. Translate $\mathbb{K}$ semantics, the claim and the trace into KORE.
3. Encode KORE into DEDUKTI. $= DK[\text{KORE}]$
4. Translate $\mathbb{K}$ semantics, the claim and the trace into $DK[\text{KORE}]$.



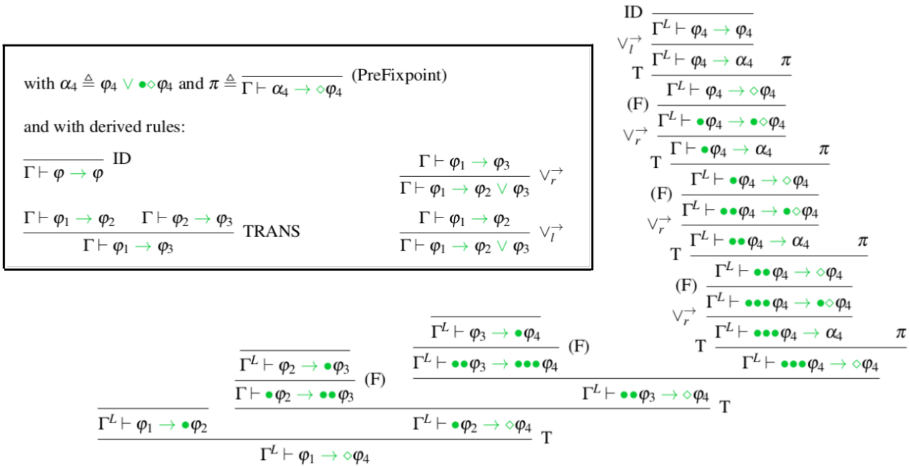
Methodology

Goal recheck: $\Gamma \vdash \varphi \rightsquigarrow \varphi'$ in DEDUKTI

1. Encode MATCHING LOGIC into DEDUKTI. $= DK[ML]$
2. Translate $\mathbb{K}$ semantics, the claim and the trace into KORE.
3. Encode KORE into DEDUKTI. $= DK[\text{KORE}]$
4. Translate $\mathbb{K}$ semantics, the claim and the trace into $DK[\text{KORE}]$.
5. Generate the proof from the KPROVER trace. $= \varphi_{DK[\text{KORE}]}$



Generate the proof from the KPROVER trace



- $\diamond\varphi \equiv \mu X.\varphi \;\vee\; \bullet X$
- KPROVER trace = each applied rule with its substitution
  $\to\ \Gamma^L \vdash \varphi_1 \ \to\ \bullet\varphi_2 + \Gamma^L \vdash \varphi_2 \ \to\ \bullet\varphi_3 + \Gamma^L \vdash \varphi_3 \ \to\ \bullet\varphi_4$

A tribute to Norman Megill

- A mathematician who created METAMATH
- Died on December 9, 2021 at the age of 71

[2]Wink to Youyou Cong. See her invited talk (TYPES 2022 - Nantes): *Composing Music from Types*

- A mathematician who created METAMATH
- Died on December 9, 2021 at the age of 71

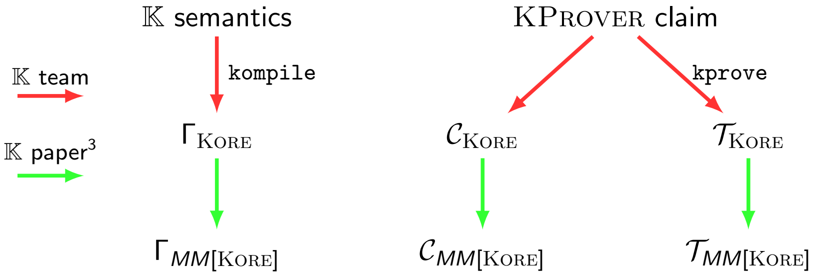
- Let's listen to the proof[2] of Russell's paradox!

[2] Wink to Youyou Cong. See her invited talk (TYPES 2022 - Nantes): *Composing Music from Types*

Methodology

Goal recheck: $\Gamma \vdash \varphi \rightsquigarrow \varphi'$ in DEDUKTI

1. Encode MATCHING LOGIC into METAMATH. $= MM[ML]$
2. Translate $\mathbb{K}$ semantics, the claim and the trace into KORE.
3. Encode KORE into METAMATH. $= MM[\text{KORE}]$
4. Translate $\mathbb{K}$ semantics, the claim and the trace into $MM[\text{KORE}]$.
5. Generate the proof from the KPROVER trace. $= \varphi_{MM[\text{KORE}]}$



[3] X. Chen, Z. Lin, M.-T. Trinh, and G. Roșu. *Towards a Trustworthy Semantics-Based Language Framework via Proof Generation.* CAV'21.

Methodology

Goal recheck: $\Gamma \vdash \varphi \rightsquigarrow \varphi'$ in DEDUKTI

1. Encode MATCHING LOGIC into METAMATH. $= MM[ML]$
2. Translate $\mathbb{K}$ semantics, the claim and the trace into KORE.
3. Encode KORE into METAMATH. $= MM[\text{KORE}]$
4. Translate $\mathbb{K}$ semantics, the claim and the trace into $MM[\text{KORE}]$.
5. Generate the proof from the KPROVER trace. $= \varphi_{MM[\text{KORE}]}$

6. Translate $MM[ML]$ and $MM[\text{KORE}]$.
7. Translate $\Gamma_{MM[\text{KORE}]}$ and $\mathcal{C}_{MM[\text{KORE}]}$.
8. Translate the generated proof $\varphi_{MM[\text{KORE}]}$ into $\varphi_{DK[\text{KORE}]}$.

Methodology

Goal recheck: $\Gamma \vdash \varphi \rightsquigarrow \varphi'$ in DEDUKTI

1. Encode MATCHING LOGIC into METAMATH. $= MM[ML]$
2. Translate $\mathbb{K}$ semantics, the claim and the trace into KORE.
3. Encode KORE into METAMATH. $= MM[\text{KORE}]$
4. Translate $\mathbb{K}$ semantics, the claim and the trace into $MM[\text{KORE}]$.
5. Generate the proof from the KPROVER trace. $= \varphi_{MM[\text{KORE}]}$

6. Translate $MM[ML]$ and $MM[\text{KORE}]$.
7. Translate $\Gamma_{MM[\text{KORE}]}$ and $\mathcal{C}_{MM[\text{KORE}]}$.
8. Translate the generated proof $\varphi_{MM[\text{KORE}]}$ into $\varphi_{DK[\text{KORE}]}$.

→ Translate METAMATH encodings
→ Translate METAMATH proofs

Example: formal number theory (Mendelson)

Constant symbol declaration

```
$c 0 + = -> ( ) term wff |- $.
```

Variable symbol declaration

```
$v t r s P Q $.
```

Typing of variables

```
tt $f term t $.
tr $f term r $.
ts $f term s $.
wp $f wff P $.
wq $f wff Q $.
```

Example: formal number theory (Mendelson)

Syntactical axioms

```
tze $a term 0 $.          tpl $a term ( t + r ) $.
weq $a wff t = r $.       wim $a wff ( P -> Q ) $.
```

Semantical axioms

```
a1 $a |- ( t = r -> ( t = s -> r = s ) ) $.
a2 $a |- ( t + 0 ) = t $.
```

Sections

```
${  min $e |- P $.
    maj $e |- ( P -> Q ) $.
    mp  $a |- Q $.      $}
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
```

term t

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tze $a term 0 $.
```

term t
term 0

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
tpl $a term ( t + r ) $.
```

term t
term 0

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
tpl $a term ( t + r ) $.
```

term (t + 0)

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
```

```
term ( t + 0 )
term t
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
weq $a wff t = r $.
```

```
term ( t + 0 )
term t
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
weq $a wff t = r $.
```

```
wff ( t + 0 ) = t
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
```

wff (t + 0) = t
term t
term t

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
weq $a wff t = r $.
```

```
wff ( t + 0 ) = t
wff t = t
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
a2  $a |- ( t + 0 ) = t $.
```

wff (t + 0) = t
wff t = t
|- (t + 0) = t

Check a Metamath proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tze $a term 0 $.
```

wff (t + 0) = t
wff t = t
|- (t + 0) = t
term t
term 0

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
tpl $a term ( t + r ) $.
```

```
wff ( t + 0 ) = t
wff t = t
|- ( t + 0 ) = t
term ( t + 0 )
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
weq $a wff t = r $.
```

```
wff ( t + 0 ) = t
wff t = t
|- ( t + 0 ) = t
wff ( t + 0 ) = t
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wff ( t + 0 ) = t
wff t = t
|- ( t + 0 ) = t
wff ( t + 0 ) = t
wff ( t + 0 ) = t
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
weq $a wff t = r $.
```

```
wff ( t + 0 ) = t
wff t = t
|- ( t + 0 ) = t
wff ( t + 0 ) = t
wff ( t + 0 ) = t
wff t = t
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wp  $f wff P $.
wq  $f wff Q $.
wim $a wff ( P -> Q ) $.
```

```
wff ( t + 0 ) = t
wff t = t
|- ( t + 0 ) = t
wff ( t + 0 ) = t
wff ( t + 0 ) = t
wff t = t
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wp  $f wff P $.
wq  $f wff Q $.
wim $a wff ( P -> Q ) $.
```

```
wff ( t + 0 ) = t
wff t = t
|- ( t + 0 ) = t
wff ( t + 0 ) = t
wff ( ( t + 0 ) = t -> t = t )
```

Check a Metamath proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
a2  $a |- ( t + 0 ) = t $.
```

```
wff ( t + 0 ) = t
wff t = t
|- ( t + 0 ) = t
wff ( t + 0 ) = t
wff ( ( t + 0 ) = t -> t = t )
|- ( t + 0 ) = t
```

Check a Metamath proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
tpl $a term ( t + r ) $.
```

```
wff ( t + 0 ) = t
wff t = t
|- ( t + 0 ) = t
wff ( t + 0 ) = t
wff ( ( t + 0 ) = t -> t = t )
|- ( t + 0 ) = t
term ( t + 0 )
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
ts  $f term s $.
a1  $a |- ( t = r ->
       ( t = s -> r = s ) ) $.
```

```
wff ( t + 0 ) = t
wff t = t
|- ( t + 0 ) = t
wff ( t + 0 ) = t
wff ( ( t + 0 ) = t -> t = t )
|- ( t + 0 ) = t
|- ( ( t + 0 ) = t -> ( ( t + 0 ) = t -> t = t ) )
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wp  $f wff P $.
wq  $f wff Q $.
${
    min $e |- P $.
    maj $e |- ( P -> Q ) $.
    mp  $a |- Q $.
$}
```

```
wff ( t + 0 ) = t
wff t = t
|- ( t + 0 ) = t
wff ( t + 0 ) = t
wff ( ( t + 0 ) = t -> t = t )
|- ( t + 0 ) = t
|- ( ( t + 0 ) = t -> ( ( t + 0 ) = t -> t = t ) )
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wp  $f wff P $.
wq  $f wff Q $.
${
    min $e |- P $.
    maj $e |- ( P -> Q ) $.
    mp  $a |- Q $.
$}
```

```
wff ( t + 0 ) = t
wff t = t
|- ( t + 0 ) = t
|- ( ( t + 0 ) = t -> t = t )
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wp  $f wff P $.
wq  $f wff Q $.
${
    min $e |- P $.
    maj $e |- ( P -> Q ) $.
    mp  $a |- Q $.
$}
```

```
wff ( t + 0 ) = t
wff t = t
|- ( t + 0 ) = t
|- ( ( t + 0 ) = t -> t = t )
```

Check a METAMATH proof

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wp  $f wff P $.
wq  $f wff Q $.
${
    min $e |- P $.
    maj $e |- ( P -> Q ) $.
    mp  $a |- Q $.
$}
```

```
|- t = t
```

Build a λ-term

Key idea: Use the Metamath proof check mechanism to build a λ-term

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
```

t

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tze $a term 0 $.
```

t
0

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
tpl $a term ( t + r ) $.
```

t
0

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
tpl $a term ( t + r ) $.
```

```
( t + 0 )
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
```

(t + 0)

t

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
weq $a wff t = r $.
```

```
( t + 0 )
t
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
weq $a wff t = r $.
```

```
( t + 0 ) = t
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
```

(t + 0) = t
t
t

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
weq $a wff t = r $.
```

```
( t + 0 ) = t
    t = t
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
a2  $a |- ( t + 0 ) = t $.
```

```
symbol a2 :  Π (t :  term),
    |− ( t + 0 ) = t ;
```

```
( t + 0 ) = t
t = t
a2 t
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tze $a term 0 $.
```

```
( t + 0 ) = t
t = t
a2 t
t
0
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
tpl $a term ( t + r ) $.
```

```
( t + 0 ) = t
    t = t
    a2 t
  ( t + 0 )
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
weq $a wff t = r $.
```

(t + 0) = t
t = t
a2 t
(t + 0) = t

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

(t + 0) = t
t = t
a2 t
(t + 0) = t
(t + 0) = t

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
weq $a wff t = r $.
```

```
( t + 0 ) = t
t = t
a2 t
( t + 0 ) = t
( t + 0 ) = t
t = t
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wp  $f wff P $.
wq  $f wff Q $.
wim $a wff ( P -> Q ) $.
```

```
( t + 0 ) = t
t = t
a2 t
( t + 0 ) = t
( t + 0 ) = t
t = t
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wp  $f wff P $.
wq  $f wff Q $.
wim $a wff ( P -> Q ) $.
```

```
( t + 0 ) = t
t = t
a2 t
( t + 0 ) = t
( ( t + 0 ) = t -> t = t )
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
a2  $a |- ( t + 0 ) = t $.
```

```
symbol a2 :  Π (t :  term),
    |− ( t + 0 ) = t ;
```

```
( t + 0 ) = t
t = t
a2 t
( t + 0 ) = t
( ( t + 0 ) = t -> t = t )
a2 t
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
tpl $a term ( t + r ) $.
```

```
( t + 0 ) = t
t = t
a2 t
( t + 0 ) = t
( ( t + 0 ) = t -> t = t )
a2 t
( t + 0 )
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
tt  $f term t $.
tr  $f term r $.
ts  $f term s $.
a1  $a |- ( t = r ->
       ( t = s -> r = s ) ) $.
```

```
symbol a1 :  Π (t r s :  term),
  |− (t = r -> ( t = s -> r = s)) ;
```

```
( t + 0 ) = t
t = t
a2 t
( t + 0 ) = t
( ( t + 0 ) = t -> t = t )
a2 t
a1 ( t + 0 ) t t
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wp  $f wff P $.
wq  $f wff Q $.
${
    min $e |- P $.
    maj $e |- ( P -> Q ) $.
    mp  $a |- Q $.
$}
symbol mp :  Π (P Q : wff),
  |− P → |− (P -> Q) → |− Q ;
```

```
( t + 0 ) = t
t = t
a2 t
( t + 0 ) = t
( ( t + 0 ) = t -> t = t )
a2 t
a1 ( t + 0 ) t t
```

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wp   $f wff P $.
wq   $f wff Q $.
${
    min $e |- P $.
    maj $e |- ( P -> Q ) $.
    mp  $a |- Q $.
$}
symbol mp :  Π (P Q : wff),
  |− P → |− (P -> Q) → |− Q ;
```

(t + 0) = t

t = t

a2 t

mp α β (a2 t) (a1 (t + 0) t t)

$\alpha \triangleq$ (t + 0) = t $\beta \triangleq \alpha$ -> t = t

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wp  $f wff P $.
wq  $f wff Q $.
${
    min $e |- P $.
    maj $e |- ( P -> Q ) $.
    mp  $a |- Q $.
$}
symbol mp :  Π (P Q : wff),
  |– P → |– (P -> Q) → |– Q ;
```

(t + 0) = t

t = t

a2 t

mp α β (a2 t) (a1 (t + 0) t t)

$\alpha \triangleq$ (t + 0) = t $\qquad$ $\beta \triangleq \alpha$ -> t = t

Build a λ-term

```
th1 $p |- t = t $=
  tt tze tpl tt weq
  tt tt weq tt a2
  tt tze tpl tt weq
  tt tze tpl tt weq
  tt tt weq wim tt a2
  tt tze tpl tt tt a1
  mp mp $.
```

```
wp  $f wff P $.
wq  $f wff Q $.
${
    min $e |- P $.
    maj $e |- ( P -> Q ) $.
    mp  $a |- Q $.
$}
symbol mp : Π (P Q : wff),
  |– P → |– (P -> Q) → |– Q ;
```

mp α γ (a2 t) (mp α β (a2 t) (a1 (t + 0) t t))

$\alpha \triangleq$ (t + 0) = t $\beta \triangleq \alpha$ -> t = t

$\gamma \triangleq$ t = t

1. The direct approach
2. The approach via Metamath
3. Conclusion

Conclusion

- The direct approach
  - ✓ More robust: only one encoding
  - ✗ Many steps and work

- The approach via METAMATH
  - ✗ Less robust: combine two encodings
  - ✓ A way to get METAMATH encodings
  - ✓ A way to get freely METAMATH proofs

Conclusion

- The direct approach → **a part of my thesis**
  - ✓ More robust: only one encoding
  - ✗ Many steps and work

- The approach via METAMATH → **internship supervised by me**
  - ✗ Less robust: combine two encodings
  - ✓ A way to get METAMATH encodings
  - ✓ A way to get freely METAMATH proofs

Conclusion

- The direct approach → **a part of my thesis**
  - ✓ More robust: only one encoding
  - ✗ Many steps and work

- The approach via METAMATH → **internship supervised by me**
  - ✗ Less robust: combine two encodings
  - ✓ A way to get METAMATH encodings
  - ✓ A way to get freely METAMATH proofs

- Open questions:
  - $MM[ML] \overset{?}{\leftrightarrow} DK[ML]$
  - $MM[\textsc{Kore}] \overset{?}{\leftrightarrow} DK[\textsc{Kore}]$
  - $\Gamma_{MM[\textsc{Kore}]} \overset{?}{\leftrightarrow} \Gamma_{DK[\textsc{Kore}]}$
  - $\mathcal{C}_{MM[\textsc{Kore}]} \overset{?}{\leftrightarrow} \mathcal{C}_{DK[\textsc{Kore}]}$

Conclusion

- The direct approach → **a part of my thesis**
  - ✓ More robust: only one encoding
  - ✗ Many steps and work

- The approach via METAMATH → **internship supervised by me**
  - ✗ Less robust: combine two encodings
  - ✓ A way to get METAMATH encodings
  - ✓ A way to get freely METAMATH proofs

- Open questions:
  - $MM[ML] \overset{?}{\leftrightarrow} DK[ML]$
  - $MM[\textsc{Kore}] \overset{?}{\leftrightarrow} DK[\textsc{Kore}]$
  - $\Gamma_{MM[\textsc{Kore}]} \overset{?}{\leftrightarrow} \Gamma_{DK[\textsc{Kore}]}$
  - $\mathcal{C}_{MM[\textsc{Kore}]} \overset{?}{\leftrightarrow} \mathcal{C}_{DK[\textsc{Kore}]}$

A new challenge for interoperability!