EUTypes 2018

Eliminating **reflection** through *reflection*

Théo **Winterhalter**

joint work with

Matthieu **Sozeau**
Nicolas **Tabareau**

Different notions of equality

Conversion

Extends the notion of β-equality

`(λx.t) u ≡ t[x← u]`

Identity types

To handle equalities within type theory

`refl` `u : u = u`

Different notions of equality

Conversion

Extends the notion of β-equality

`(λx.t) u ≡ t[x← u]`

Identity types

To handle equalities within type theory

`refl` `u : u = u`

If `u ≡ v` then **`refl`** `u : u = v`

Reflection

Conversion

Extends the notion of β-equality

Identity types

To handle equalities within type theory

Reflection

Conversion ← **Identity types**

Extends the notion of β-equality | To handle equalities within type theory

$$\frac{p : u = v}{u \equiv v}$$

Example

Example

$vec_A : nat \to \mathbf{Type}$

Example

```
vec_A : nat → Type

[] : vec_A 0
```

Example

```
vec_A : nat → Type

[] : vec_A 0

cons : ∀ n, A → vec_A n → vec_A (S n)
```

Example

```
vecA : nat → Type

[] : vecA 0

cons : ∀ n, A → vecA n → vecA (S n)


rev : ∀ {n m}, vecA n → vecA m → vecA (n + m)
```

Example

```
vecA : nat → Type
[] : vecA 0
cons : ∀ n, A → vecA n → vecA (S n)

rev : ∀ {n m}, vecA n → vecA m → vecA (n + m)
rev [] acc ≐ acc
```

Example

```
vecA : nat → Type

[] : vecA 0

cons : ∀ n, A → vecA n → vecA (S n)

rev : ∀ {n m}, vecA n → vecA m → vecA (n + m)

rev [] acc ≐ acc

rev (cons n a v) acc ≐ rev v (cons m a acc)
```

Example

```
vecA : nat → Type
[] : vecA 0
cons : ∀ n, A → vecA n → vecA (S n)
```

```
rev : ∀ {n m}, vecA n → vecA m → vecA (n + m)
rev [] acc ≐ acc
rev (cons n a v) acc ≐ rev v (cons m a acc)
                                     vecA m
```

3

Example

```
vecA : nat → Type

[] : vecA 0

cons : ∀ n, A → vecA n → vecA (S n)


rev : ∀ {n m}, vecA n → vecA m → vecA (n + m)

rev [] acc ≐ acc

rev (cons n a v) acc ≐ rev v (cons m a acc)

                                vecA (S m)
```

3

Example

```
vecA : nat → Type

[] : vecA 0

cons : ∀ n, A → vecA n → vecA (S n)
```

```
rev : ∀ {n m}, vecA n → vecA m → vecA (n + m)

rev [] acc ≐ acc

rev (cons n a v) acc ≐ rev v (cons m a acc)

                       vecA (n + S m)
```

3

Example

```
vecA : nat → Type

[] : vecA 0

cons : ∀ n, A → vecA n → vecA (S n)


rev : ∀ {n m}, vecA n → vecA m → vecA (n + m)

rev [] acc ≐ acc

rev (cons n a v) acc ≐ rev v (cons m a acc)
```

expected: vecA (S n + m) ≠ vecA (n + S m)

Example

```
vecA : nat → Type

[] : vecA 0

cons : ∀ n, A → vecA n → vecA (S n)

rev : ∀ {n m}, vecA n → vecA m → vecA (n + m)

rev [] acc ≐ acc

rev (cons n a v) acc ≐ rev v (cons m a acc)
```

reflection ⇒ `vecA (S n + m) ≡ vecA (n + S m)`

Intensional VS Extensional

$$\frac{p : u = v}{u \equiv v}$$

ETT = ITT + reflection

Intensional VS Extensional

$$\frac{p : u = v}{u \equiv v}$$

ETT = ITT + reflection

What is the relation between the two?

Intensional VS Extensional

What is the relation between the two?

ETT is *conservative* over **ITT + K + funext**

1995 Martin Hofmann

Intensional VS Extensional

What is the relation between the two?

ETT is *conservative* over **ITT + K + funext**

1995 Martin Hofmann

Intensional VS Extensional

What is the relation between the two?

ETT is *conservative* over **ITT + K + funext**

1995 Martin Hofmann

```
K : ∀ A (x : A) (e : x = x), e = refl x
```

Intensional VS Extensional

What is the relation between the two?

ETT is *conservative* over **ITT + K + funext**

1995 Martin Hofmann

```
K : ∀ A (x : A) (e : x = x), e = refl x

funext : ∀ A B (f g : A → B), (∀ (x : A), f x = g x) → f = g
```

Intensional VS Extensional

What is the relation between the two?

ETT can be *translated* to **ITT + K + funext + ?**

2005 Nicolas Oury

```
K : ∀ A (x : A) (e : x = x), e = refl x

funext : ∀ A B (f g : A → B), (∀ (x : A), f x = g x) → f = g
```

Intensional VS Extensional

What is the relation between the two?

ETT can be *translated* to **ITT + K + funext + ?**

2005 Nicolas Oury

```
K : ∀ A (x : A) (e : x = x), e = refl x

funext : ∀ A B (f g : A → B), (∀ (x : A), f x = g x) → f = g
```

'?' : « heterogenous equality is a congruence for application »

Intensional VS Extensional

What is the relation between the two?

ETT can be *translated* to **ITT + K + funext**

TODAY

```
K : ∀ A (x : A) (e : x = x), e = refl x
funext : ∀ A B (f g : A → B), (∀ (x : A), f x = g x) → f = g
'?' : « heterogenous equality is a congruence for application »
```

Intensional VS Extensional

What is the relation between the two?

ETT can be *translated* to **ITT + K + funext**

TODAY

```
K : ∀ A (x : A) (e : x = x), e = refl x

funext : ∀ A B (f g : A → B), (∀ (x : A), f x = g x) → f = g

~~'?' : « heterogenous equality is a congruence for application »~~
```

Intensional VS Extensional

What is the relation between the two?

ETT can be *translated* to **ITT + K + funext**

TODAY

Intensional VS Extensional

What is the relation between the two?

ETT can be *translated* to **ITT + K + funext**

TODAY

- **Minimal** (axiom-wise)
- **Constructive** (formalised in Coq)
- **Computes** (produces Coq terms)

Fundamental difference

Oury	Hofmann / us

Fundamental difference

Oury

Minimal annotations

```
λ(x : A).t

t u
```

Hofmann / us

Fundamental difference

Oury

Minimal annotations

`λ(x : A).t`

`t u`

Hofmann / us

Fully annotated terms

`λ(x : A).B.t`

$t\ @^{(x:A).B}\ u$

Fundamental difference

Oury

Minimal annotations

$\lambda(x : A).t$

$t\ u$

Hofmann / us

Fully annotated terms

$\lambda(x : A).B.t$

$t\ @^{(x:A).B}\ u$

Blocked β-reduction

$(\lambda(x : A).B.t)\ @^{(x:A).B}\ u$

$\equiv t[x := u]$

Fundamental difference

Oury

Minimal annotations

$\lambda(x : A).t$

$t\ u$

Free β-reduction

$(\lambda(x : A).t)\ u$

$\equiv t[x := u]$

Hofmann / us

Fully annotated terms

$\lambda(x : A).B.t$

$t\ @^{(x:A).B}\ u$

Blocked β-reduction

$(\lambda(x : A).B.t)\ @^{(x:A).B}\ u$

$\equiv t[x := u]$

Fundamental difference

Oury

Free β-reduction

```
(λ(x : A).t) u
  ≡ t[x := u]
```

Hofmann / us

Blocked β-reduction

```
(λ(x : A).B.t) @^(x:A).B u
  ≡ t[x := u]
```

Fundamental difference

Oury

Free β-reduction

$$(\lambda(x : A).x)\ u \equiv x[x := u]$$

Hofmann / us

Blocked β-reduction

$$(\lambda(x : A).B.t)\ @^{(x:A).B}\ u \equiv t[x := u]$$

Fundamental difference

Oury

Free β-reduction

```
(λ(x : A).x) u
  ≡ u
```

Hofmann / us

Blocked β-reduction

```
(λ(x : A).B.t) @(x:A).B u
  ≡ t[x := u]
```

Fundamental difference

Oury

Free β-reduction

$$(\lambda(x : \mathtt{nat}).x)\ 0 \equiv 0$$

Hofmann / us

Blocked β-reduction

$$(\lambda(x : A).B.t)\ @^{(x:A).B}\ u$$
$$\equiv t[x := u]$$

Fundamental difference

Oury

Free β-reduction

$$\frac{(\lambda(x : \mathtt{nat}).x)\ 0 \equiv 0}{\mathtt{nat} \to \mathtt{nat}}$$

Hofmann / us

Blocked β-reduction

$$(\lambda(x : A).B.t)\ @^{(x:A).B}\ u \equiv t[x := u]$$

Fundamental difference

Oury

Free β-reduction

$$\frac{(\lambda(x : \mathtt{nat}).x)\ 0 \equiv 0}{\mathtt{nat} \to \mathtt{nat} \equiv \mathtt{nat} \to \mathtt{bool}}$$

under consistent context

Hofmann / us

Blocked β-reduction

$$(\lambda(x : A).B.t)\ @^{(x:A).B}\ u \equiv t[x := u]$$

Fundamental difference

Oury

Free β-reduction

$$\frac{(\lambda(x : \mathtt{nat}).x)\ 0 \equiv 0}{\mathtt{bool}}$$

Hofmann / us

Blocked β-reduction

$$(\lambda(x : A).B.t)\ @^{(x:A).B}\ u$$
$$\equiv t[x := u]$$

Fundamental difference

Oury

Free β-reduction

$$\frac{(\lambda(x : \mathtt{nat}).x)\ 0}{\mathtt{bool}} \equiv \frac{0}{\mathtt{nat}}$$

Hofmann / us

Blocked β-reduction

$$(\lambda(x : A).B.t)\ @^{(x:A).B}\ u$$
$$\equiv t[x := u]$$

Fundamental difference

Oury

Free β-reduction

$$\frac{(\lambda(x : \mathtt{nat}).x)\ 0}{\mathtt{bool}} \equiv \frac{0}{\mathtt{nat}}$$

bool ≠ nat

under consistent context

Hofmann / us

Blocked β-reduction

$$(\lambda(x : A).B.t)\ @^{(x:A).B}\ u$$

$$\equiv t[x := u]$$

Fundamental difference

Oury

Free β-reduction

$$\underbrace{(\lambda(x : \mathtt{nat}).x)\ 0}_{\mathtt{bool}} \equiv \underbrace{0}_{\mathtt{nat}}$$

bool ≠ nat

No Uniqueness of type

OR

No Subject reduction

Hofmann / us

Blocked β-reduction

$$(\lambda(x : A).B.t)\ @^{(x:A).B}\ u$$

$$\equiv t[x := u]$$

Fundamental difference

Oury

Free β-reduction

$$\underbrace{(\lambda(x : \mathtt{nat}).x)\ 0}_{\mathtt{bool}} \equiv \underbrace{0}_{\mathtt{nat}}$$

bool ≠ nat

No Uniqueness of type

OR

No Subject reduction

Hofmann / us

Blocked β-reduction

$$(\lambda(x : A).B.t)\ @^{(x:A).B}\ u \equiv t[x := u]$$

Uniqueness of type

$\Gamma \vdash t : A$ and $\Gamma \vdash t : B$

$\Rightarrow \Gamma \vdash A \equiv B$

Principle of the translation

ETT	ITT

Principle of the translation

ETT

Typing derivation

ITT

Well typed term

Principle of the translation

ETT

Typing derivation

$$\dfrac{\dfrac{\vdots \qquad \vdots}{\dots \qquad \dots} \qquad \vdots}{\dfrac{\dots \qquad \dots}{\Gamma \vdash_x t : A}}$$

→

ITT

Well typed term

$$\Gamma' \vdash_i t' : A'$$

Principle of the translation

ETT	ITT
Typing derivation	Well typed term

$$\dfrac{\dfrac{\vdots \quad \vdots}{\dots \quad \dots}\ \ \dots \qquad \dfrac{\vdots}{\dots}}{\Gamma \vdash_x t : A} \quad \longrightarrow \quad \Gamma' \vdash_i t' : A'$$

Idea: *Conversion* is translated to *transport*.

Principle of the translation

ETT

Typing derivation

$$\dfrac{\begin{array}{cc} \vdots & \vdots \\ \Gamma \vdash_x t : B & \Gamma \vdash_x B \equiv A \end{array}}{\Gamma \vdash_x t : A}$$

→

ITT

Well typed term

$$\Gamma' \vdash_i t' : B'$$
$$\Gamma' \vdash_i e : B' = A'$$
$$\Downarrow$$
$$\Gamma' \vdash_i \mathbf{transp}(e, t') : A'$$

Idea: *Conversion* is translated to *transport*.

Principle of the translation

ETT

Typing derivation

$$\frac{\begin{array}{cc}\vdots & \vdots \\ \Gamma \vdash_x t : B & \Gamma \vdash_x B \equiv A\end{array}}{\Gamma \vdash_x t : A}$$

⟶

ITT

Well typed term

$$\Gamma' \vdash_i t' : B'$$

$$\Gamma' \vdash_i e : B' = A'$$

$$\Downarrow$$

$$\Gamma' \vdash_i \mathbf{transp}(e, t') : A'$$

Idea: *Conversion* is translated to *transport*.

⇒ Coherence problems

Heterogenous equality

$a \; {}_{A}{=}_{B} \; b$

Heterogenous equality

$$a \;{}_{A}{=}_{B}\; b$$

$$\doteq \sum (p : A = B),\ \mathbf{transp}(p, a) = b$$

Terms up to transport

$$\frac{t \sim t'}{t \sim \mathbf{transp}(e, t')}$$

Terms up to transport

$$\frac{\texttt{t} \sim \texttt{t'}}{\texttt{t} \sim \textbf{transp}\texttt{(e,t')}}
\qquad
\frac{\texttt{t} \sim \texttt{t'} \quad \texttt{A} \sim \texttt{A'} \quad \texttt{B} \sim \texttt{B'} \quad \texttt{u} \sim \texttt{u'}}{\texttt{t } @^{\texttt{(x:A).B}} \texttt{ u} \sim \texttt{t' } @^{\texttt{(x:A').B'}} \texttt{ u'}}$$

Terms up to transport

$$\frac{t \sim t'}{t \sim \mathbf{transp}(e, t')}$$

$$\frac{t \sim t' \quad A \sim A' \quad B \sim B' \quad u \sim u'}{t \; @^{(x:A).B} \; u \sim t' \; @^{(x:A').B'} \; u'}$$

…

Terms up to transport

$$\frac{\texttt{t} \sim \texttt{t'}}{\texttt{t} \sim \textbf{transp}\texttt{(e,t')}} \qquad \frac{\texttt{t} \sim \texttt{t'} \quad \texttt{A} \sim \texttt{A'} \quad \texttt{B} \sim \texttt{B'} \quad \texttt{u} \sim \texttt{u'}}{\texttt{t @}^{\texttt{(x:A).B}}\ \texttt{u} \sim \texttt{t' @}^{\texttt{(x:A').B'}}\ \texttt{u'}} \qquad \dots$$

Invariant

`t` is **translated** to `t'` with `t ~ t'`

Terms up to transport

$$\frac{t \sim t'}{t \sim \mathbf{transp}(e,t')} \qquad \frac{t \sim t' \quad A \sim A' \quad B \sim B' \quad u \sim u'}{t \ @^{(x:A).B}\ u \sim t' \ @^{(x:A').B'}\ u'} \qquad \ldots$$

Invariant

t is **translated** to t' with $t \sim t'$

Fundamental lemma

Given Γ and $t \sim t'$, there exists a term p such that if $\Gamma \vdash_i t : A$ and $\Gamma \vdash_i t' : B$ then $\Gamma \vdash_x p : t\ {}_A{=}_B\ t'$.

Translation

```
if ⊢x Γ  then ∑ Γt ~ Γ, ⊢i Γt
```

Translation

$$\text{if } \vdash_x \Gamma \text{ then } \sum \Gamma^t \sim \Gamma,\ \vdash_i \Gamma^t$$

$$\text{if } \Gamma \vdash_x t : A \text{ then}$$
$$\forall\ \Gamma^t \sim \Gamma,\ \vdash_i \Gamma^t \to \sum\ (t^t \sim t)\ (A^t \sim A),\ \Gamma^t \vdash_i t^t : A^t$$

Translation

if $\vdash_x \Gamma$ then $\sum \Gamma^t \sim \Gamma, \vdash_i \Gamma^t$

if $\Gamma \vdash_x t : A$ then

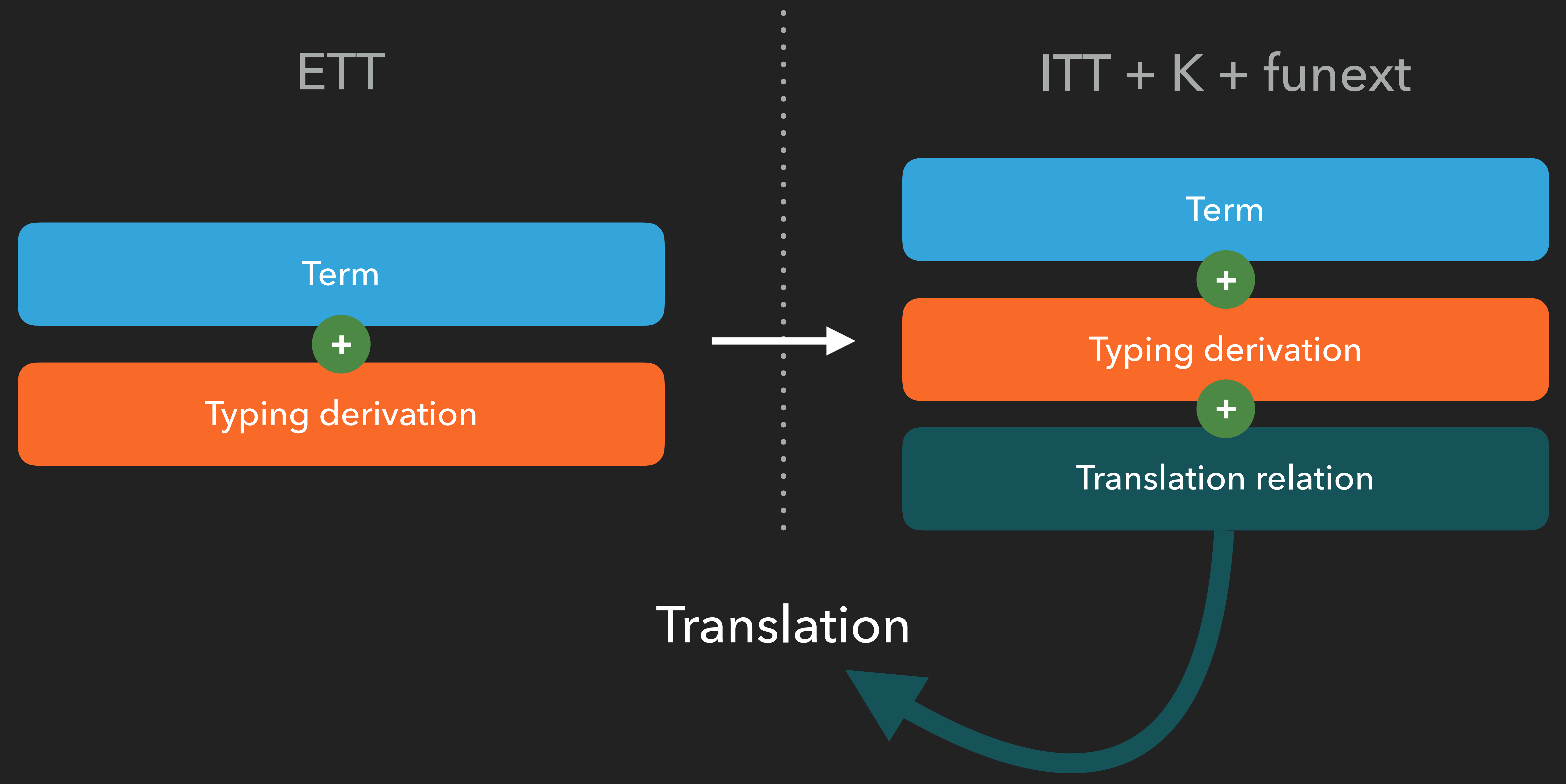
$\forall\ \Gamma^t \sim \Gamma,\ \vdash_i \Gamma^t \to \sum\ (t^t \sim t)\ (A^t \sim A),\ \Gamma^t \vdash_i t^t : A^t$

if $\Gamma \vdash_x t \equiv u : A$ then

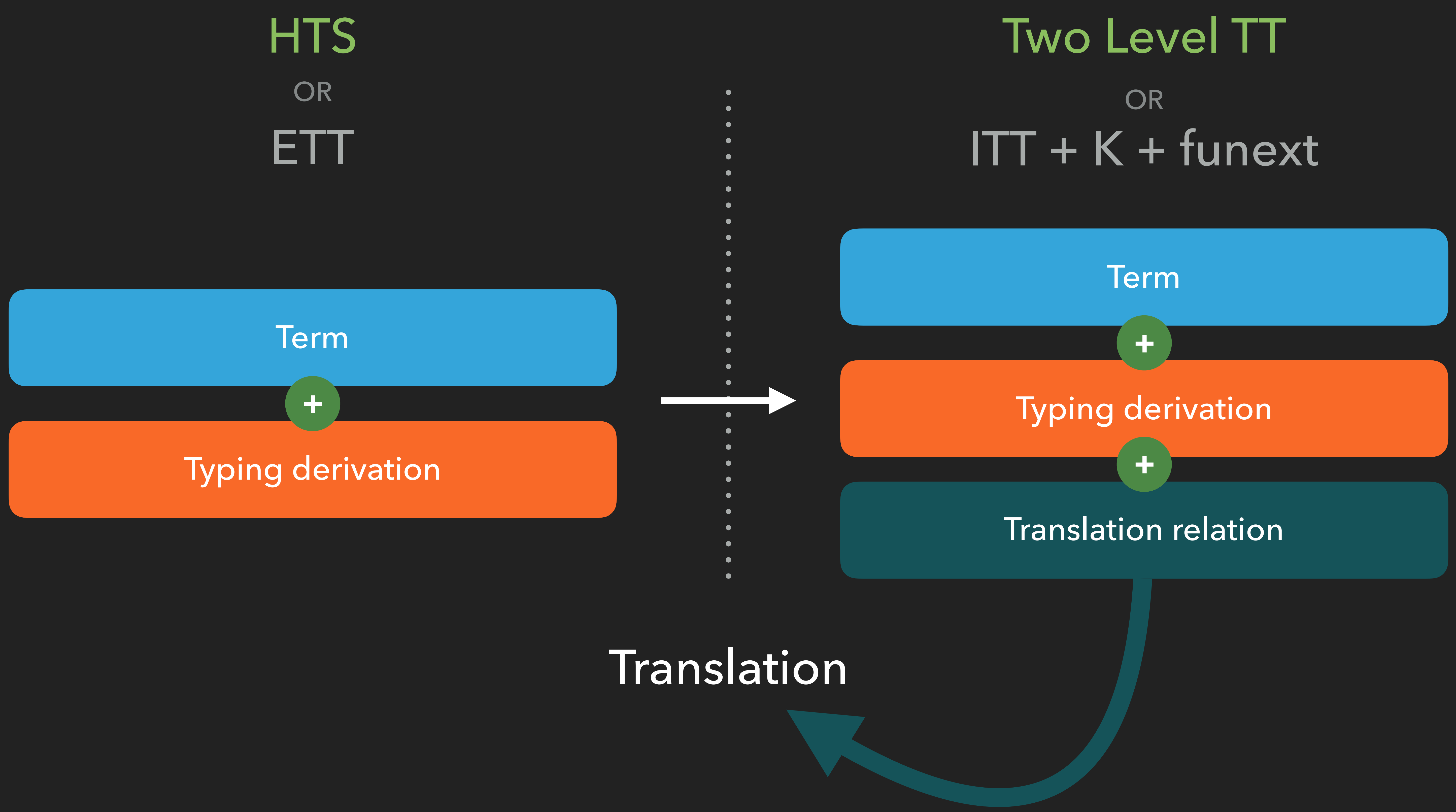
$\forall\ \Gamma^t \sim \Gamma,\ \vdash_i \Gamma^t \to \sum\ (t^t \sim t)\ (A^t \sim A)\ (u^t \sim u)\ (A^s \sim A)\ p,$

$\Gamma^t \vdash_i p : t^t\ {}_{A^t = A^s}\ u^t$

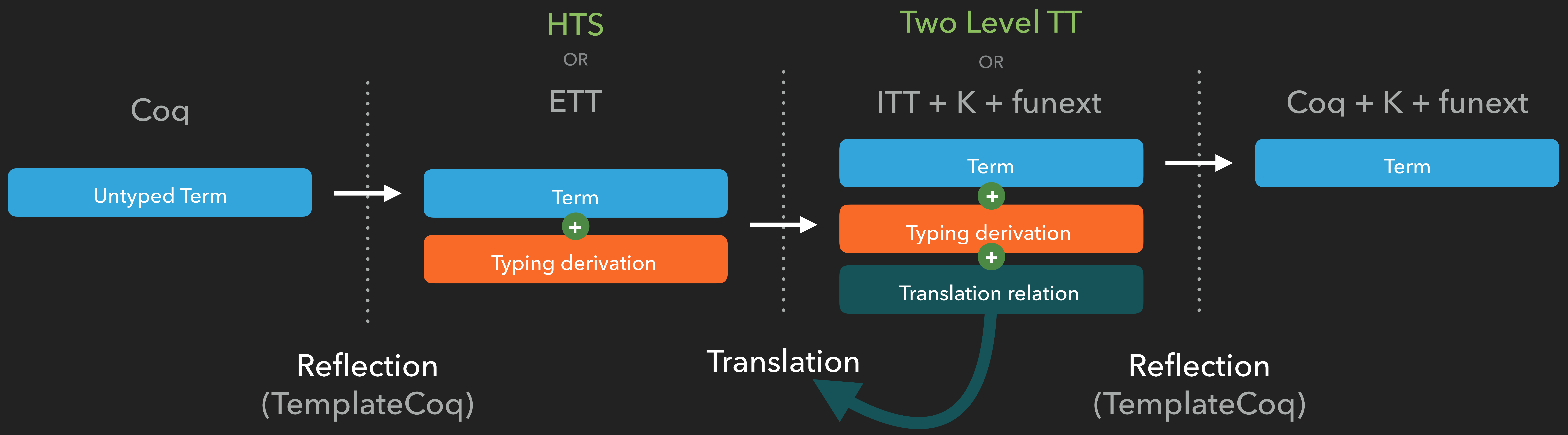
Conclusion


ETT
ITT + K + funext
Term
Typing derivation
Term
Typing derivation
Translation relation
Translation

Conclusion

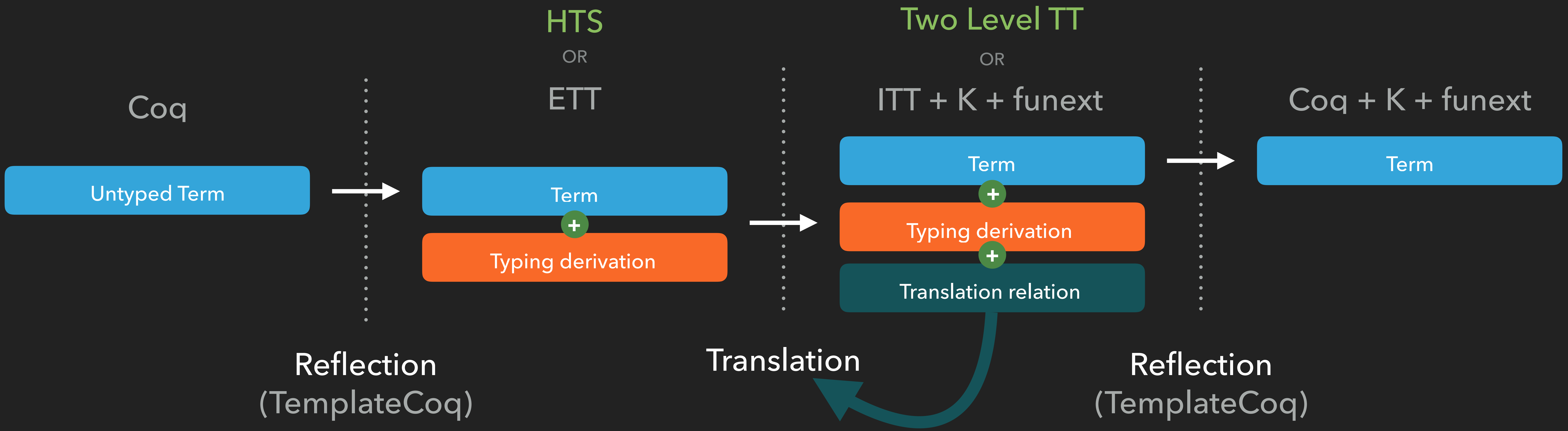

HTS
OR
ETT
Term
+
Typing derivation
Two Level TT
OR
ITT + K + funext
Term
+
Typing derivation
+
Translation relation
Translation

Conclusion



HTS
OR
ETT
Two Level TT
OR
ITT + K + funext
Coq
Coq + K + funext
Untyped Term
Term
+
Typing derivation
Term
+
Typing derivation
+
Translation relation
Term
Reflection
(TemplateCoq)
Translation
Reflection
(TemplateCoq)

Conclusion



https://github.com/TheoWinterhalter/**ett-to-itt**