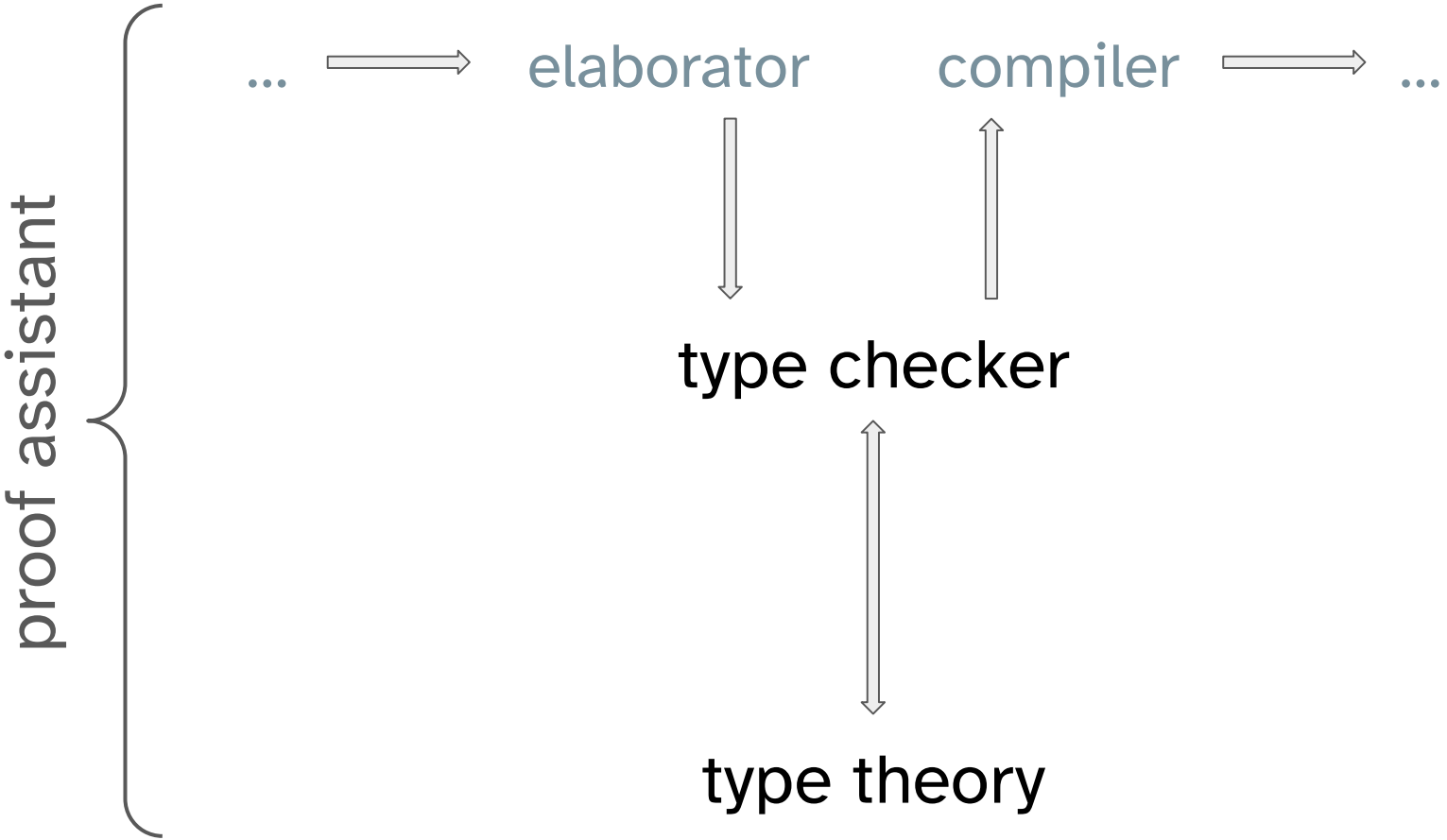
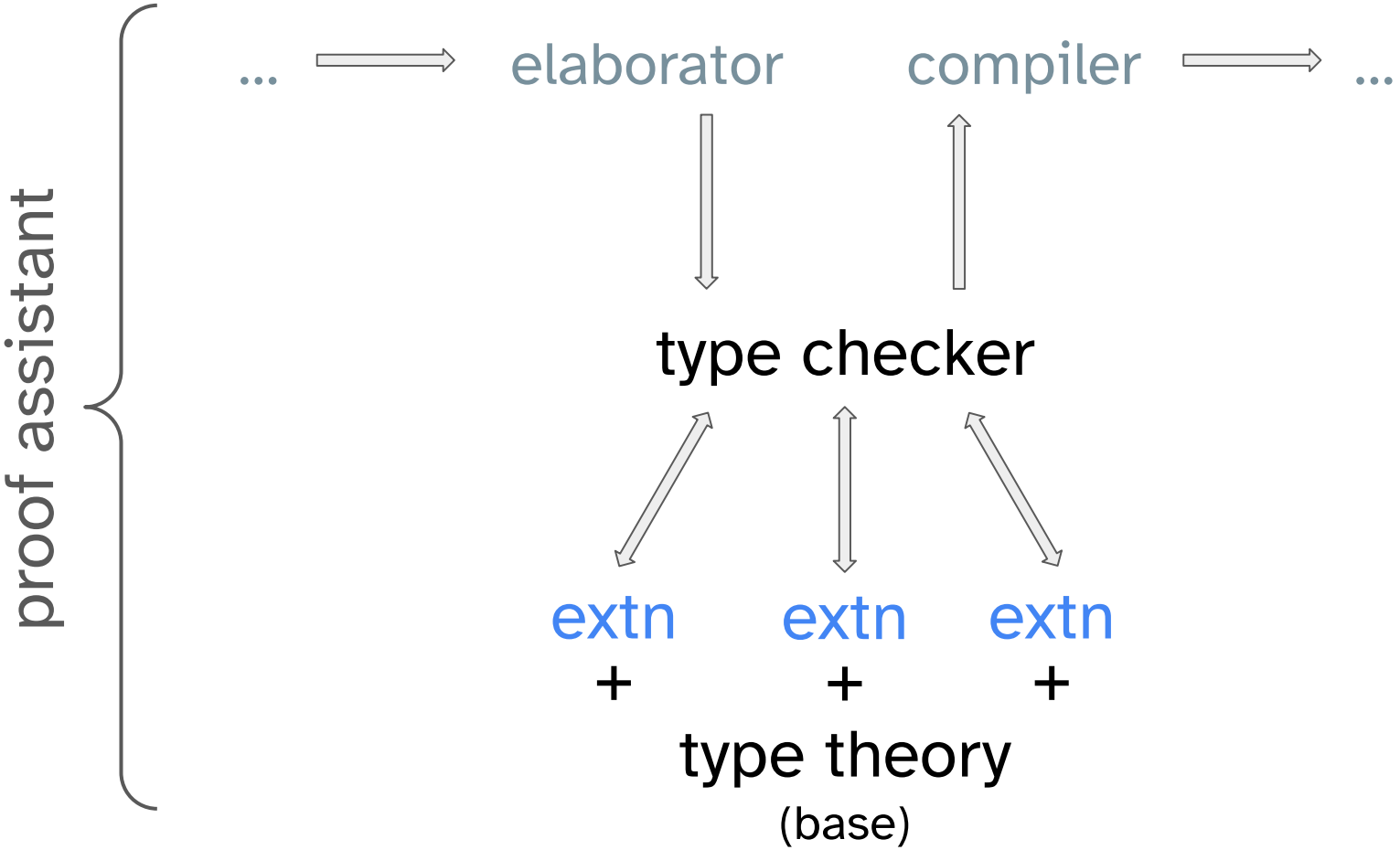
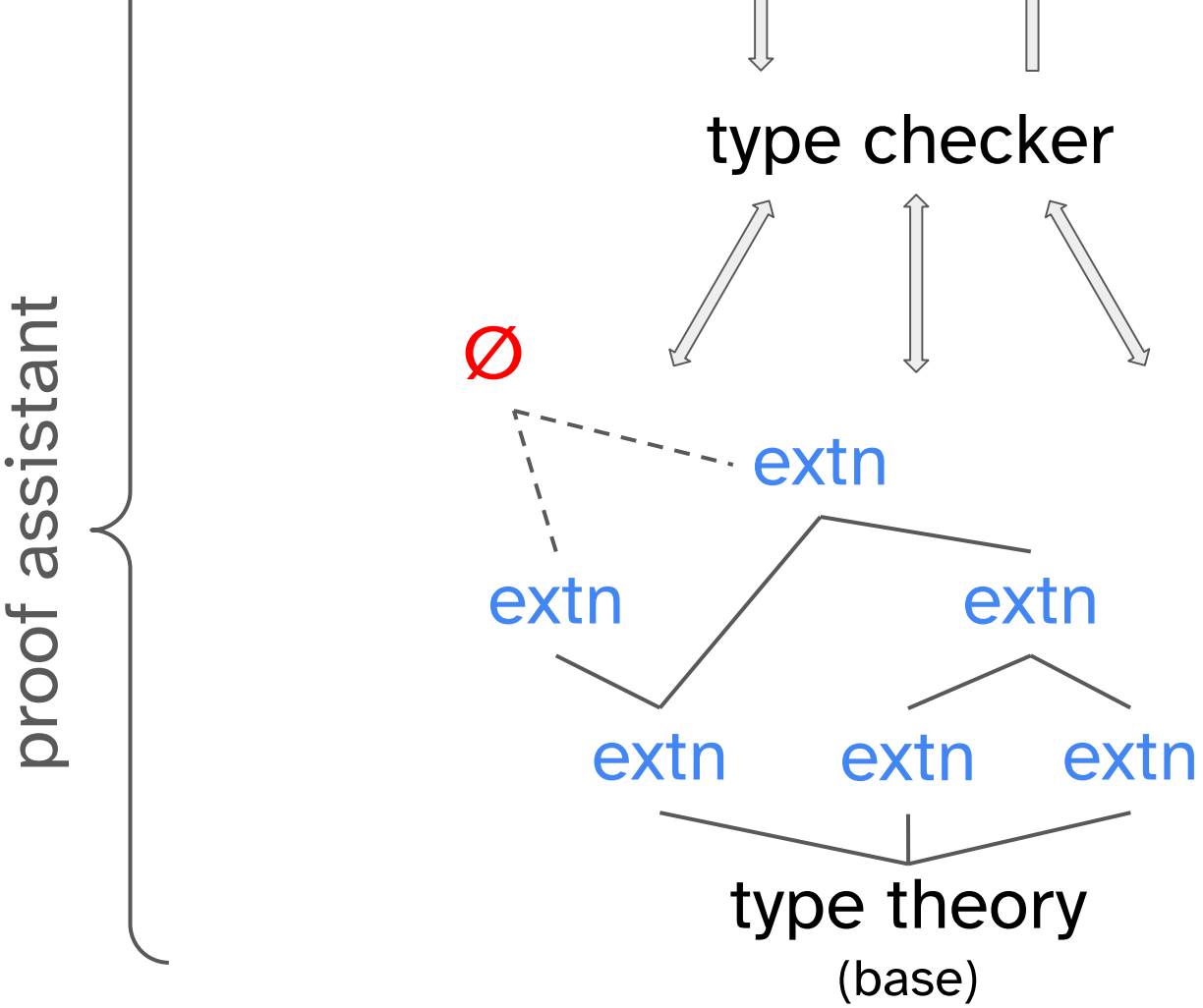
internalizing extensions in lattices of type theories

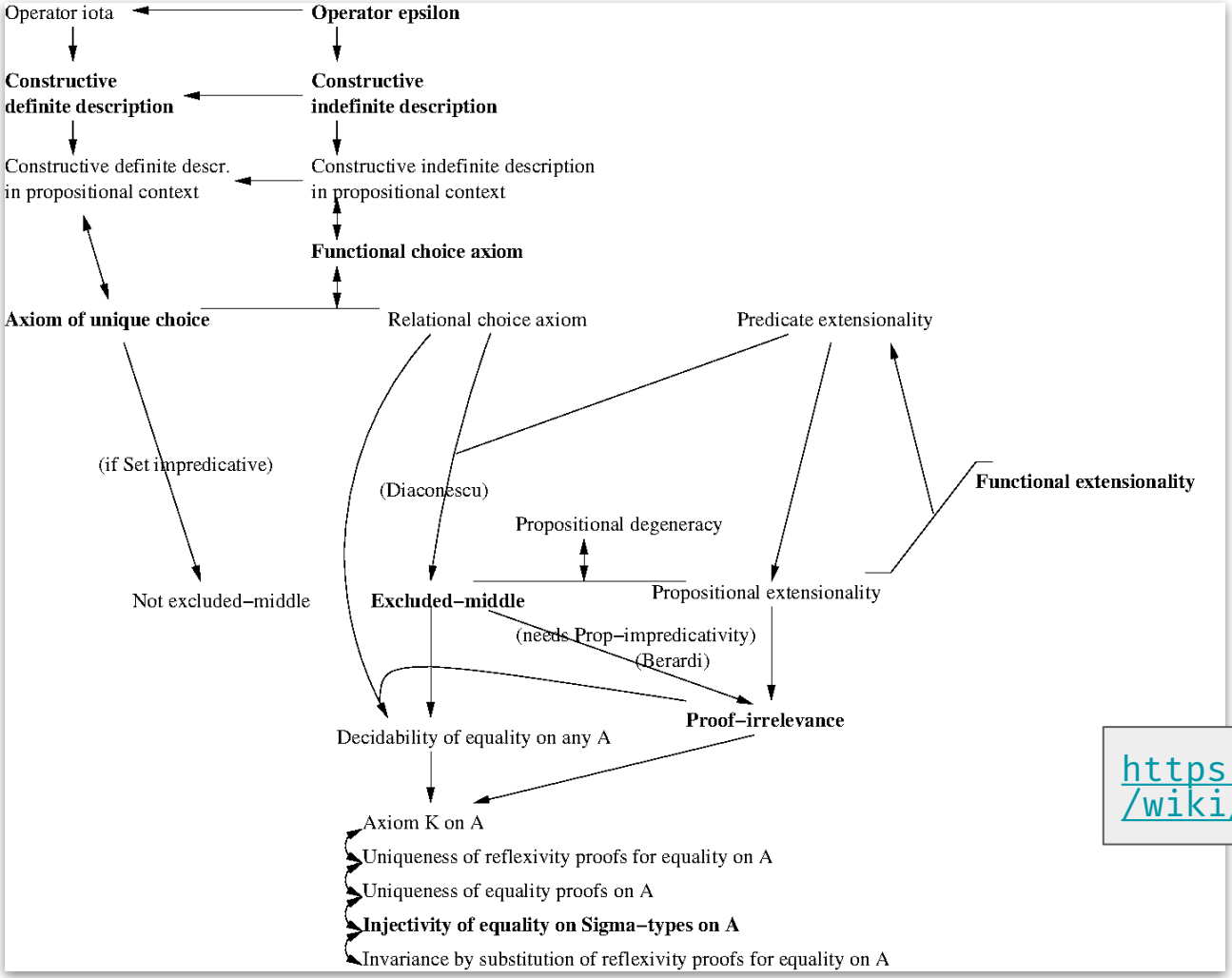
Jonathan Chan

21 November 2024


proof assistant
...
elaborator
compiler
...
type checker
type theory


proof assistant
...
elaborator
compiler
...
type checker
extn
extn
extn
+
+
+
type theory
(base)


proof assistant
type checker
Ø
extn
extn
extn
extn
extn
extn
type theory
(base)



https://github.com/coq/coq/wiki/The-Logic-of-Coq

Checking options for consistency

Agda checks that options used in imported modules are consistent with each other.

An *infective* option is an option that if used in one module, must be used in all modules that depend on this module. The following options are infective:

- `--cohesion`
- `--erased-matches`
- `--erasure`
- `--flat-split`
- `--guarded`
- `--prop`
- `--rewriting`
- `--two-level`

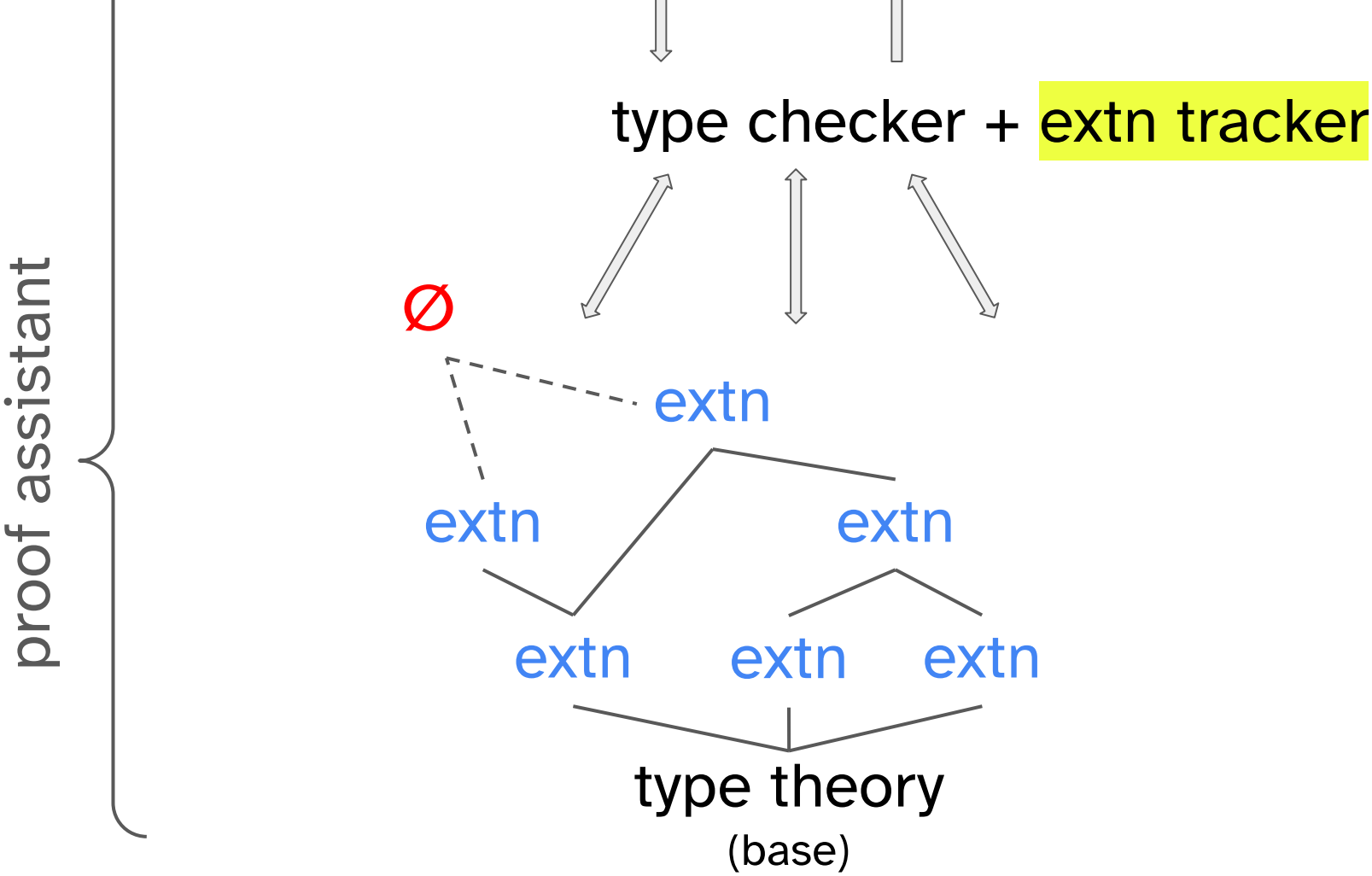
Furthermore `--cubical` and `--erased-cubical` are *jointly infective*: if one of them is used in one module, then one or the other must be used in all modules that depend on this module.

A *coinfective* option is an option that if used in one module, must be used in all modules that this module depends on. The following options are coinfective:

- `--safe`
- `--without-K`
- `--no-universe-polymorphism`
- `--no-sized-types`
- `--no-guardedness`
- `--level-universe`

Furthermore the option `--cubical-compatible` is mostly coinfective. If a module uses `--cubical-`

https://agda.readthedocs.io/en/latest/tools/command-line-options.html#checking-options-for-consistency

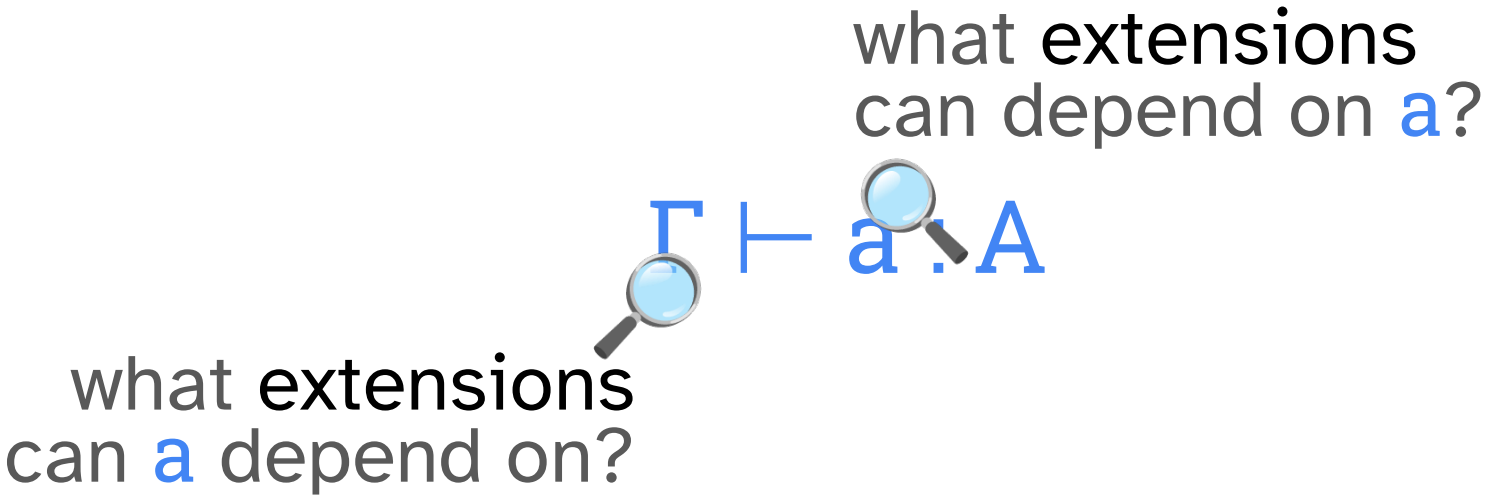

proof assistant
type checker + extn tracker
Ø
extn
extn
extn
extn
extn
extn
type theory
(base)

what's missing: internalization

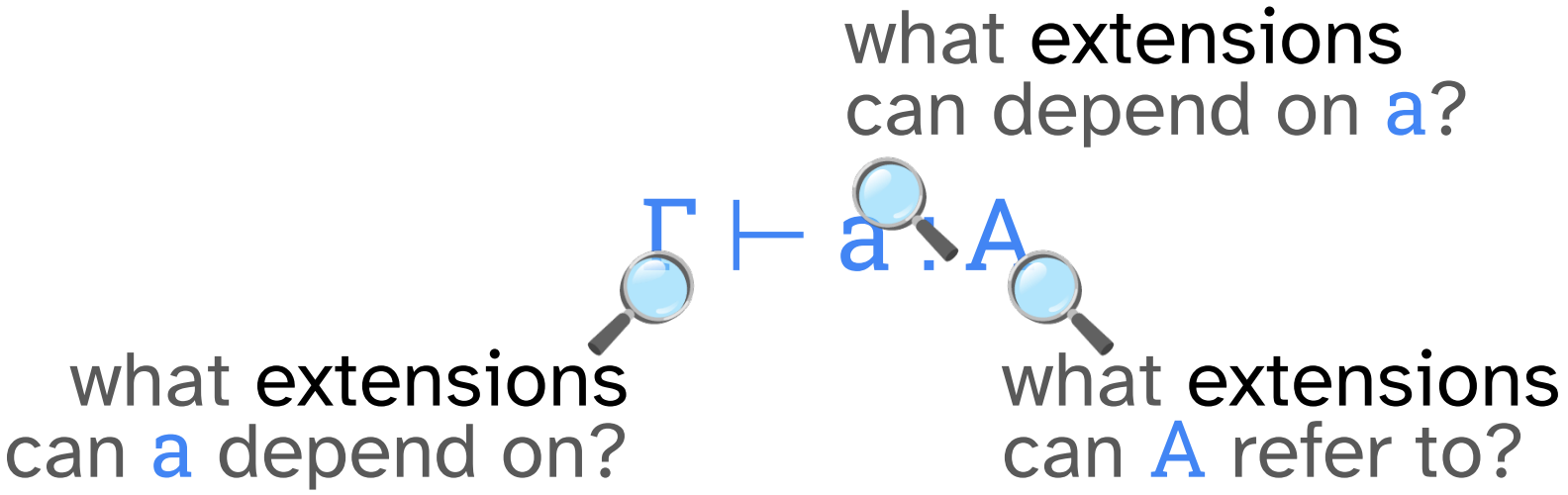
$\Gamma \vdash a : A$

what extensions
can a depend on?

what's missing: internalization



what's missing: internalization



Dependent Calculus of Indistinguishability (Liu, Chan, Shi, Weirich 2024/25)

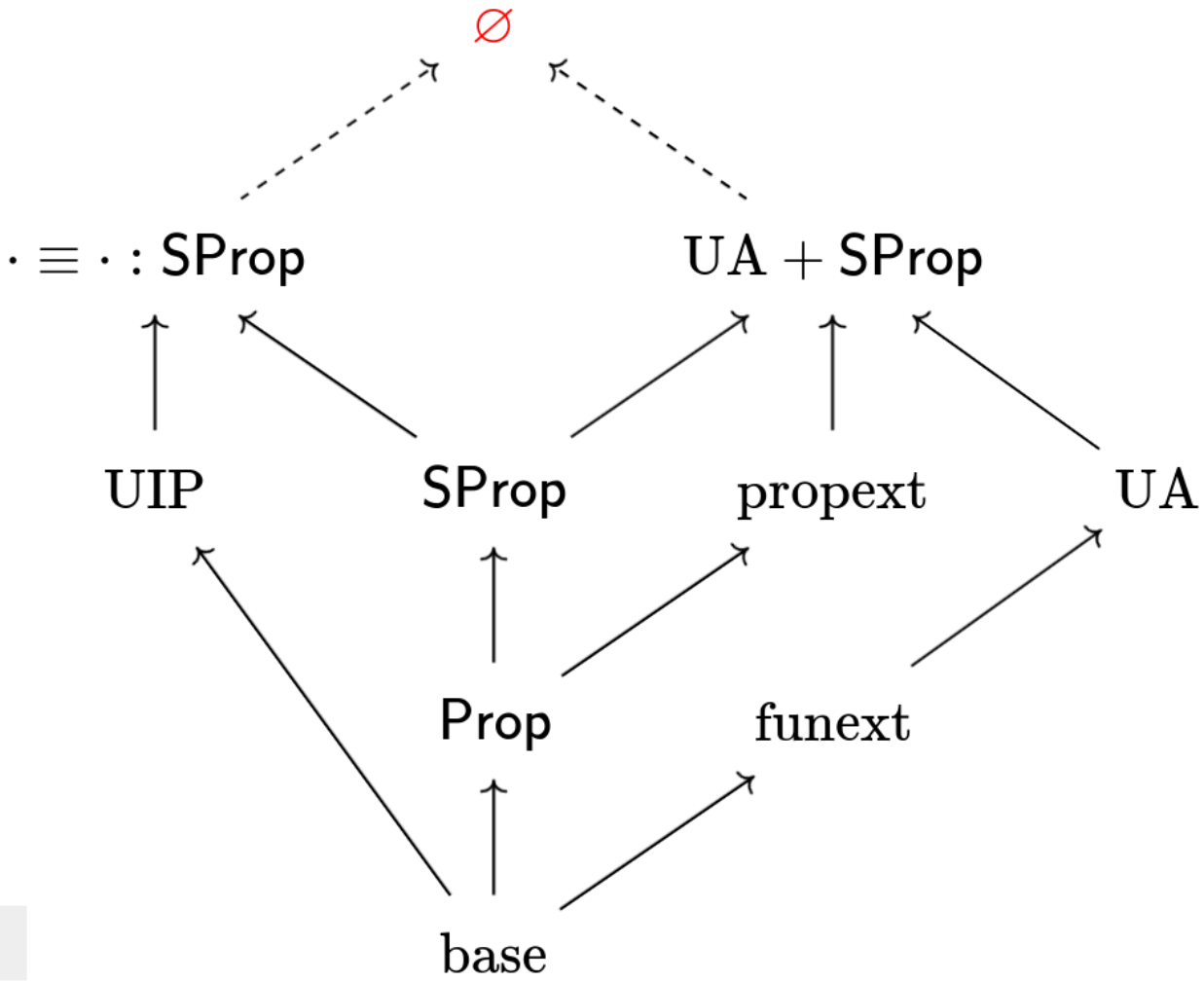
$$\Gamma, x : B \mathbin{@} \ell_1 \vdash a : A \mathbin{@} \ell_2$$

$\ell ::=$ levels from a lattice

outline

of the rest of the talk

1. extensions
2. DCOI
3. lattices of TTs
4. related work



1. extensions

univalence

$$(A \simeq B) \simeq (A \equiv B)$$

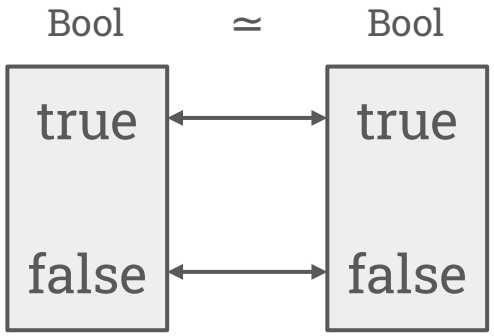
$$\mathrm{ua} : \forall (A\ B : \mathrm{Type}).\ (A \simeq B) \to (A \equiv B)$$

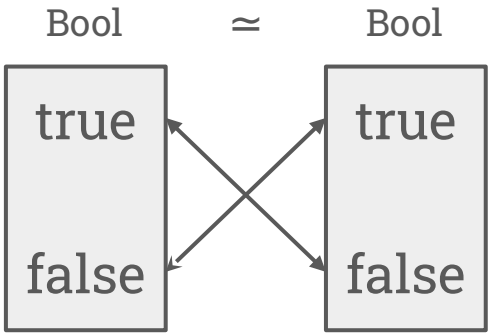
1. extensions

univalence

$$(A \simeq B) \simeq (A \equiv B)$$

$$\mathrm{ua} : \forall (A\ B : \mathrm{Type}).\ (A \simeq B) \to (A \equiv B)$$



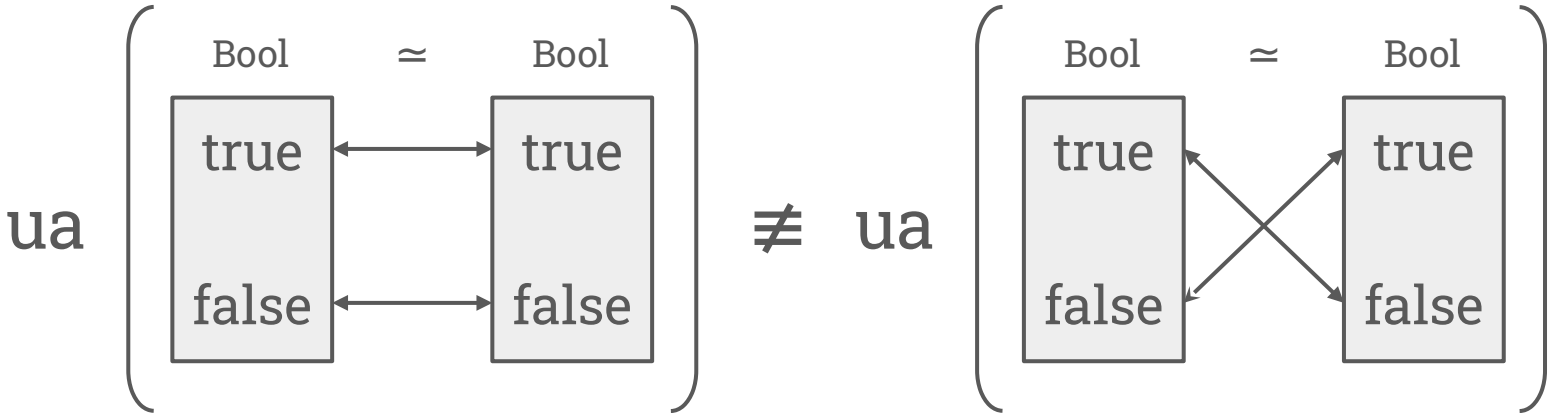


1. extensions

univalence

$$(A \simeq B) \simeq (A \equiv B)$$

$$\mathrm{ua} : \forall (A\ B : \mathrm{Type}).\ (A \simeq B) \to (A \equiv B)$$



1. extensions

uniqueness of identity proofs

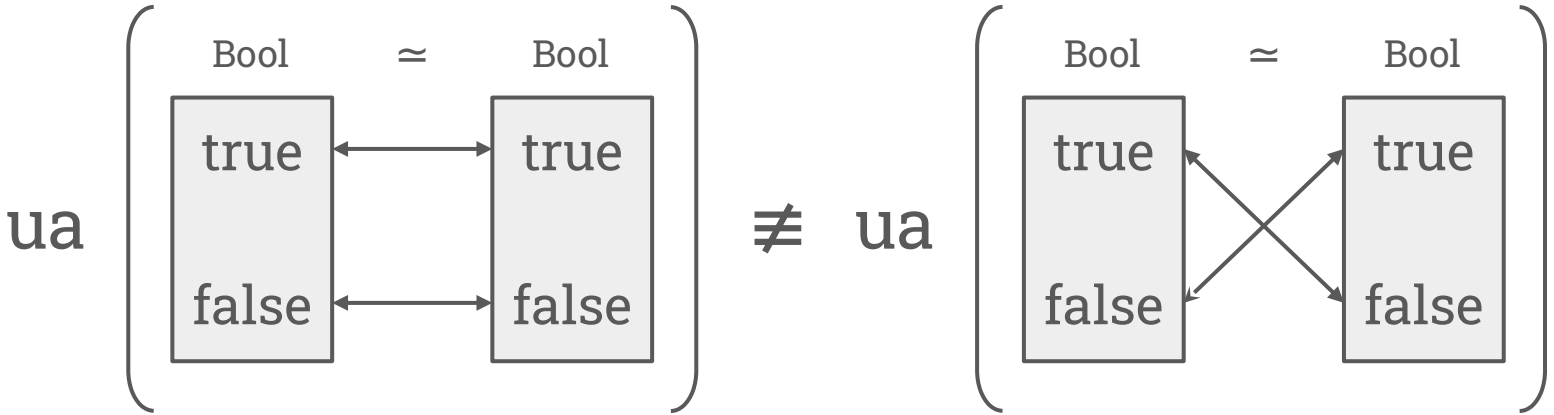
```
UIP : ∀(A : Type) (a b : A) (p q : a ≡ b). p ≡ q
UIP _ _ _ refl refl ⇝ refl

K : ∀ (A : Type) (a : A) (P : a ≡ a → Prop) (p : a ≡ a).
    P refl → P p
K _ _ _ refl d ⇝ d
```

1. extensions

UIP + univalence ⇒ ⊥

UIP Type Bool Bool : ∀(p q : Bool ≡ Bool). p ≡ q



1. extensions

DCOI

Dependent Calculus of Indistinguishability (Liu, Chan, Shi, Weirich 2024/25)

$$\Gamma, x : B \mathbin{@} \ell_1 \vdash a : A \mathbin{@} \ell_2$$

$$\ell ::= L \leq H \leq S$$

DCOI for information flow control

$$\frac{x : A \mathbin{@} \ell \in \Gamma \qquad \ell \leq \ell'}{\Gamma \vdash x : A \mathbin{@} \ell'}$$

$$\lambda x^{L}\, y^{H}.\; x : (\mathrm{Nat} \mathbin{@} L) \to (\mathrm{Nat} \mathbin{@} H) \to \mathrm{Nat} \mathbin{@} L$$

DCOI for information flow control

$$\frac{\Gamma \vdash a : A \mathbin{@} \ell \qquad \ell \leq \ell'}{\Gamma \vdash a : A \mathbin{@} \ell'}$$

$$\lambda x^{L}\, y^{H}.\, x : (\mathrm{Nat} \mathbin{@} L) \to (\mathrm{Nat} \mathbin{@} H) \to \mathrm{Nat} \mathbin{@} L$$

$$\lambda x^{L}\, y^{H}.\, y : (\mathrm{Nat} \mathbin{@} L) \to (\mathrm{Nat} \mathbin{@} H) \to \mathrm{Nat} \mathbin{@} H$$

$$\lambda x^{L}\, y^{H}.\, x + 1 : (\mathrm{Nat} \mathbin{@} L) \to (\mathrm{Nat} \mathbin{@} H) \to \mathrm{Nat} \mathbin{@} H$$

DCOI for run-time erasure of types

$$\lambda A^{H}\ x^{L}\ y^{H}.\ x : \Pi(A : \mathrm{Type}_1 \ @\ H).\ (A\ @\ L) \to (A\ @\ H) \to A\ @\ L$$

DCOI for run-time erasure of types

$$\frac{\Gamma \vdash a : A \mathbin{@} \ell}{\exists \ell', \mathcal{U}.\, \Gamma \vdash A : \mathcal{U} \mathbin{@} \ell'}$$

$$\lambda A^{H}\, x^{L}\, y^{H}.\, x : \Pi(A : \mathrm{Type}_1 \mathbin{@} H).\, (A \mathbin{@} L) \to (A \mathbin{@} H) \to A \mathbin{@} L$$

$$\Pi(A : \mathrm{Type}_1 \mathbin{@} H).\, (A \mathbin{@} L) \to (A \mathbin{@} H) \to A : \mathrm{Type}_2 \mathbin{@} H$$

DCOI for compile-time irrelevance

$$\lambda P^{H}\ p^{L}.\ p : \Pi(P : (\mathrm{Nat} \mathbin{@} S) \to \mathrm{Type} \mathbin{@} H).\ (P\ 0^{S} \mathbin{@} L) \to P\ 1^{S} \mathbin{@} L$$

DCOI for compile-time irrelevance

$$\frac{\Gamma \vdash f = g \mathbin{@} \ell \qquad \ell' \not\leq \ell}{\Gamma \vdash f\ a^{\ell'} = g\ b^{\ell'} \mathbin{@} \ell}$$

$$\vdash P\ 0^{S} = P\ 1^{S} \mathbin{@} H$$

$$\lambda P^{H}\ p^{L}.\ p : \Pi(P : (\mathrm{Nat} \mathbin{@} S) \to \mathrm{Type} \mathbin{@} H).\ (P\ 0^{S} \mathbin{@} L) \to P\ 1^{S} \mathbin{@} L$$

DCOI internalizes indistinguishability

$$\frac{\Gamma \vdash a = b \mathbin{@} \ell}{\Gamma \vdash \mathrm{refl} : a \equiv^{\ell} b \mathbin{@} \ell'}$$

$$\vdash P\ 0^{S} = P\ 1^{S} \mathbin{@} H$$

$$\lambda P^{H}.\ \mathrm{refl} : \Pi(P : (\mathrm{Nat} \mathbin{@} S) \to \mathrm{Type} \mathbin{@} H).\ P\ 0^{S} \equiv^{H} P\ 1^{S} \mathbin{@} L$$

DCOI and eliminating ⊥

$$\mathrm{head} : \forall(l : \mathrm{List}\ A\ @\ \mathrm{L}).\ (0 < \mathrm{len}\ l\ @\ \mathrm{H}) \to A\ @\ \mathrm{L}$$
$$\mathrm{head}\ \mathrm{nil}^{\mathrm{L}} : (0 < 0\ @\ \mathrm{H}) \to A\ @\ \mathrm{L}$$

DCOI and eliminating ⊥

$$\frac{\Gamma \vdash b : \bot \mathbin{@} \ell}{\Gamma \vdash \mathsf{absurd}\ b : A \mathbin{@} \ell'}$$

$$\mathsf{zeroNotGt} : (0 < 0 \mathbin{@} H) \to \bot \mathbin{@} H$$

$$\mathsf{head} : \forall (l : \mathsf{List}\ A \mathbin{@} L).\ (0 < \mathsf{len}\ l \mathbin{@} H) \to A \mathbin{@} L$$

$$\mathsf{head}\ \mathsf{nil}^{L} : (0 < 0 \mathbin{@} H) \to A \mathbin{@} L$$

$$\mathsf{head}\ \mathsf{nil}^{L} \coloneqq \lambda q^{H}.\ \mathsf{absurd}\ (\mathsf{zeroNotGt}\ q)$$

variables

$$\frac{x : A \mathbin{@} \ell \in \Gamma \qquad \ell \leq \ell'}{\Gamma \vdash x : A \mathbin{@} \ell'}$$

subsumption

$$\frac{\Gamma \vdash a : A \mathbin{@} \ell \qquad \ell \leq \ell'}{\Gamma \vdash a : A \mathbin{@} \ell'}$$

regularity

$$\frac{\Gamma \vdash a : A \mathbin{@} \ell}{\exists \ell', \mathcal{U}.\, \Gamma \vdash A : \mathcal{U} \mathbin{@} \ell'}$$

$$\frac{\Gamma \vdash f = g \mathbin{@} \ell \qquad \ell' \not\leq \ell}{\Gamma \vdash f\ a^{\ell'} = g\ b^{\ell'} \mathbin{@} \ell}$$

indistinguishability

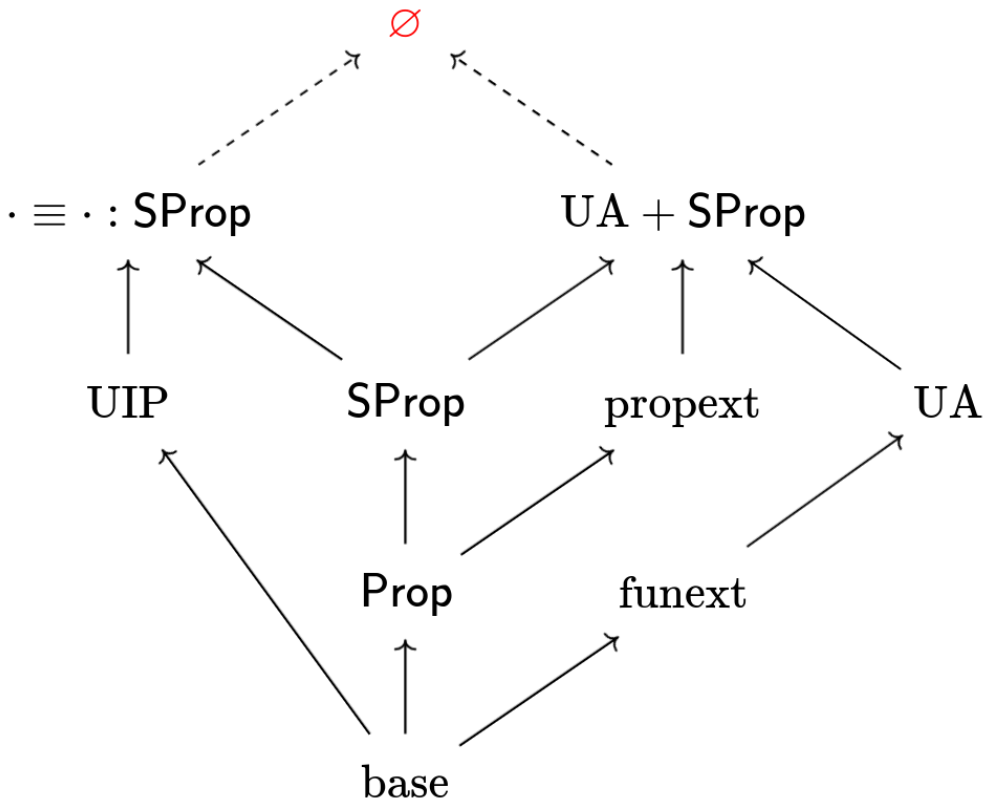
$$\frac{\Gamma \vdash a = b \mathbin{@} \ell}{\Gamma \vdash \mathrm{refl} : a \equiv^{\ell} b \mathbin{@} \ell'}$$

equality

$$\frac{\Gamma \vdash b : \bot \mathbin{@} \ell}{\Gamma \vdash \mathrm{absurd}\ b : A \mathbin{@} \ell'}$$

$\bot$ elimination

lattices of type theories



$\ell ::= \mathsf{base} \mid \mathsf{Prop} \mid \mathsf{SProp} \mid \mathsf{funext} \mid \mathsf{propext} \mid \mathsf{UIP} \mid \mathsf{UA} \mid \ldots$

extensions at levels

$\Gamma \vdash \mathrm{Prop} : \mathrm{Type}_1 \ @\ \mathrm{Prop}$

$\Gamma \vdash \Pi(x : A).\ B : \mathrm{Prop} \ @\ \mathrm{Prop}$

$\Gamma \vdash K\ P\ p\ d : P\ p \ @\ \mathrm{UIP}$

$K\ P\ \mathrm{refl}\ d \rightsquigarrow d$

$\Gamma \vdash \mathrm{propext} : \forall(A\ B : \mathrm{SProp}).\ (A \leftrightarrow B) \to A \equiv B \ @\ \mathrm{propext}$

example: indistinguishability

$$\mathrm{UIP} \not\leq \mathrm{UA}$$

$$\vdash \mathrm{f}\ \mathrm{x}^{\mathrm{UIP}} = \mathrm{f}\ \mathrm{y}^{\mathrm{UIP}} @ \mathrm{UA}$$

example: indistinguishability + equality

$$\mathrm{UIP} \not\leq \mathrm{UA}$$

$$\vdash \mathrm{f}\ \mathrm{x}^{\mathrm{UIP}} = \mathrm{f}\ \mathrm{y}^{\mathrm{UIP}} @ \mathrm{UA}$$

$$\vdash \mathrm{refl} : \mathrm{f}\ \mathrm{x}^{\mathrm{UIP}} \equiv^{\mathrm{UA}} \mathrm{f}\ \mathrm{y}^{\mathrm{UIP}} @ \ell$$

example: regularity

$$(\ldots \text{ua} \ldots, \lambda q. \ldots) :$$
$$\Sigma(p : \text{Bool} \equiv^{\text{UA}} \text{Bool} \mathbin{@} \text{UA}).$$
$$[(p \equiv^{\text{UA}} \text{refl} \mathbin{@} \text{UA}) \to \bot \mathbin{@} \text{UA}] \mathbin{@} \text{base}$$

example: regularity + subsumption

$$(\ldots \text{ua} \ldots, \lambda q. \ldots) :$$
$$\Sigma(p : \text{Bool} \equiv^{\text{UA}} \text{Bool} \ @ \ \text{UA}).$$
$$[(p \equiv^{\text{UA}} \text{refl} \ @ \ \text{UA}) \to \bot \ @ \ \text{UA}] \ @ \ \text{UIP}$$

N.B. $\bot$ elimination: $\exists \ell$. inconsistent $\Rightarrow \forall \ell$. inconsistent

$$\frac{\vdash b : \bot \ @ \ \mathbf{UA + UIP}}{\vdash \mathrm{absurd}\ b : \bot \ @ \ \ell}$$

questions to answer

questions to answer

1. expressivity: what extensions fit?
    - new rules/eliminators/universes/axioms/equalities

questions to answer

1. expressivity: what extensions fit?
    - new rules/eliminators/universes/axioms/equalities
2. expressivity: what more can be proven?
    - UIP facts about UA? constructive facts about classical?

questions to answer

1. expressivity: what extensions fit?
    - new rules/eliminators/universes/axioms/equalities
2. expressivity: what more can be proven?
    - UIP facts about UA? constructive facts about classical?
3. how to prove logical consistency?
    - extensibility to more extensions

part 1: implementation

- many common extensions
- reimplement parts of stdlibs of Rocq/Agda/Lean
- Agda 2LTT lib: UA + UIP
- level inference? level polymorphism?

part 2: formalization

- one (1) extension + base
- DCOI^{ω}: DCOI w/ predicative universes
- syntactic logical relation indexed by universe levels
- UIP/funext likely hold semantically
- impredicativity/typed equality likely difficult

part 2: formalization

- one (1) extension + base

- DCOI$^{\omega}$: DCOI w/ predicative universes
- syntactic logical relation indexed by universe levels

- syntactic modelling:
  type preserving translation to a consistent theory
- SProp [Gilbert et al., 2019], Ghost TT [Winterhalter, 2024]

related work

prior work

DCOI$^{\omega}$ *Consistency of a Dependent Calculus of Indistinguishability*. Yiyun Liu, Jonathan Chan, and Stephanie Weirich. POPL 2025.

DCOI *Internalizing Indistinguishability with Dependent Types*. Yiyun Liu, Jonathan Chan, Jessica Shi, and Stephanie Weirich. POPL 2024.

StraTT *Stratified Type Theory*. Jonathan Chan and Stephanie Weirich. ESOP 2025.

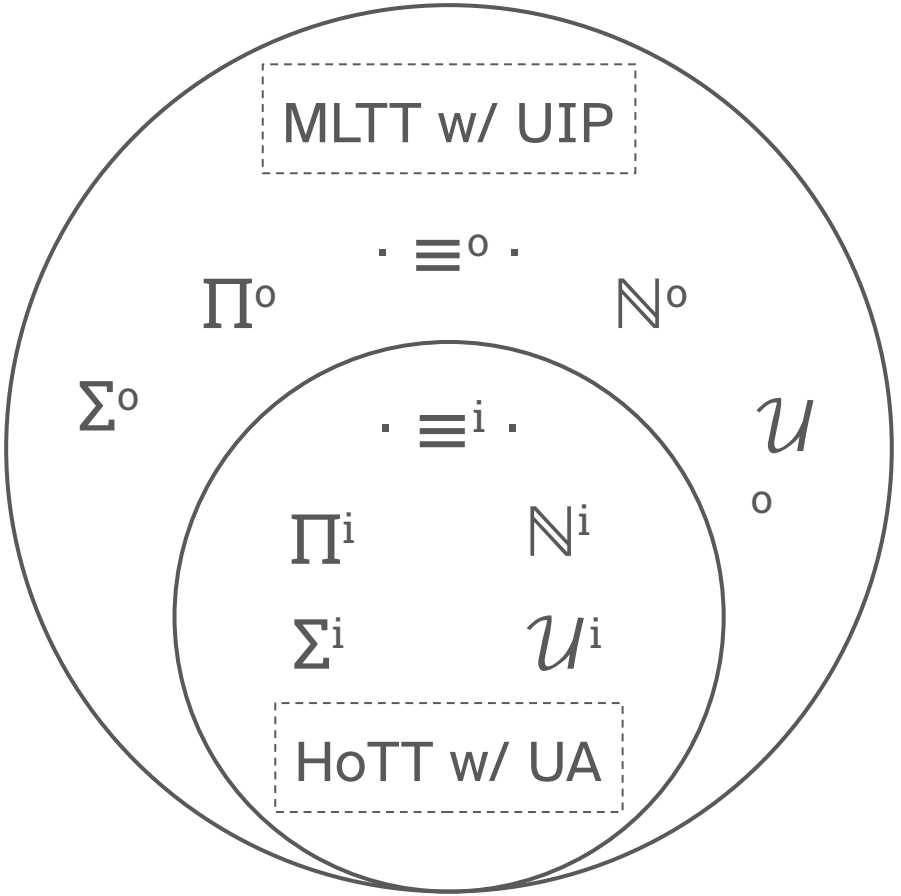
2LTT [Voevodsky; Altenkirch et al., 2016; Annenkov et al., 2023]

2LTT:

$$\forall x^{i}\ y^{i}.\ \uparrow x \equiv^{o} \uparrow y \to x \equiv^{i} y$$

DCOI:

$$\forall x^{L}\ y^{L}.\ x \equiv^{H} y \to x \equiv^{L} y$$



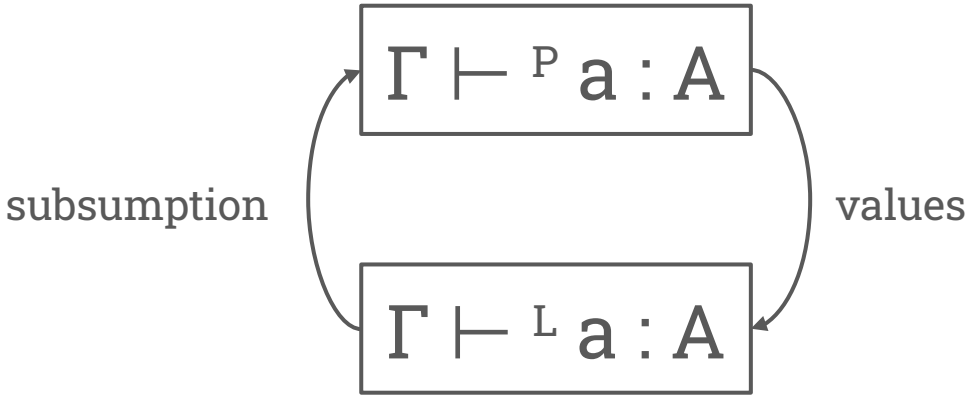
Trellys: *logical proofs + (diverging) programming*

λ^{θ}, LF^{θ}, PCC^{θ} [Casinghino, 2014]
Zombie [Sjöberg, 2015]

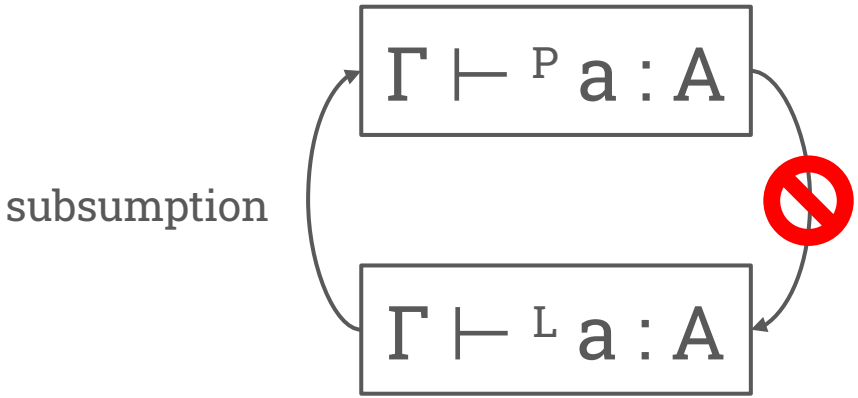
Sep^{3} [Kimmell et al., 2012]

Nax [Ahn, 2014]

λ^{θ}, LF^{θ}, PCC^{θ} [Casinghino, 2014]
Zombie [Sjöberg, 2015]



System DE [Liu and Weirich, 2023]



more proof assistants!

- **F⋆** has monadic ordered effects, including Tot ≤ Dv; Tot types cannot depend on Dv values
- **Idris 2** has pragmas total ≤ covering ≤ partial; partial terms will not reduce in types

Summary

- proof assistants have extensions
  that extend the underlying type theory
- some extensions are mutually incompatible
- type checkers track which extensions are used externally
- can’t refer to extensions internally —
  imprecise and restrictive
- DCOI: dependency tracking system w/ dependent types
- we might be able use DCOI to track extensions