A general definition of dependent type theories

Peter LeFanu Lumsdaine

Stockholm University

Logic Colloquium 2017, impromptu Type Theory session
adapted from Stockholm Logic Seminar, 15 Feb 2017

A general definition of dependent type theories

Peter LeFanu Lumsdaine

Stockholm University

Logic Colloquium 2017, impromptu Type Theory session
adapted from Stockholm Logic Seminar, 15 Feb 2017

Motivation

Some meta-theorems

T_Π: dependent type theory with just Π-types.

Some meta-theorems

T_Π: dependent type theory with just Π-types.

Theorem (Cartmell, Streicher)

The syntax of T_Π *presents the initial contextual category with* Π*-type structure.*

Some meta-theorems

T_Π: dependent type theory with just Π-types.

Theorem (Cartmell, Streicher)

The syntax of T_Π *presents the initial contextual category with* Π*-type structure.*

Theorem (Hofmann)

The logical framework embedding $\mathsf{T}_\Pi \longrightarrow \mathsf{T}_{LF[\Pi]}$ *is conservative.*

Some meta-theorems

T_Π: dependent type theory with just Π-types.

Theorem (Cartmell, Streicher)

The syntax of T_Π presents the initial contextual category with Π-type structure.

Theorem (Hofmann)

The logical framework embedding $\mathsf{T}_\Pi \longrightarrow \mathsf{T}_{LF[\Pi]}$ is conservative.

Theorem (Hofmann, rephrased by Lumsdaine–Warren)

$\mathbf{C}$ *a comprehension category with pseudo-stable Π-type structure. Then the CwA $\mathbf{C}_*$ carries strictly stable Π-type structure.*

Some more meta-theorems

$\mathsf{T}_{\mathrm{ETT}}$: dependent type theory with Π-, Σ-, unit, and extensional Id-types.

Some more meta-theorems

$\mathsf{T}_{\mathrm{ETT}}$: dependent type theory with Π-, Σ-, unit, and extensional Id-types.

Theorem (Hofmann, straightforward extension of Cartmell/Streicher)

The syntax of $\mathsf{T}_{\mathrm{ETT}}$ *presents the initial contextual category with* Π-, Σ-, *unit, and extensional* Id*-type structure.*

Some more meta-theorems

$\mathsf{T}_{\mathrm{ETT}}$: dependent type theory with Π-, Σ-, unit, and extensional Id-types.

Theorem (Hofmann, straightforward extension of Cartmell/Streicher)

The syntax of $\mathsf{T}_{\mathrm{ETT}}$ *presents the initial contextual category with* Π-, Σ-, *unit, and extensional* Id-*type structure.*

Theorem (straightforward extension of Hofmann)

The logical framework embedding $\mathsf{T}_{\mathrm{ETT}} \longrightarrow \mathsf{T}_{LF[\mathrm{ETT}]}$ *is conservative.*

Some more meta-theorems

$\mathsf{T}_{\mathrm{ETT}}$: dependent type theory with Π-, Σ-, unit, and extensional Id-types.

Theorem (Hofmann, straightforward extension of Cartmell/Streicher)

The syntax of $\mathsf{T}_{\mathrm{ETT}}$ *presents the initial contextual category with* Π-, Σ-, *unit, and extensional* Id-*type structure.*

Theorem (straightforward extension of Hofmann)

The logical framework embedding $\mathsf{T}_{\mathrm{ETT}} \longrightarrow \mathsf{T}_{LF[\mathrm{ETT}]}$ *is conservative.*

Theorem (straightforward extension of Hofmann / Lumsdaine–Warren)

$\mathbf{C}$ *a comprehension category with pseudo-stable* Π-, Σ-, *etc. structure. Then the CwA* $\mathbf{C}_*$ *carries strictly stable* Π-, Σ-, *etc. structure.*

Yet more meta-theorems

$\mathsf{T}_{\mathrm{HoTT}}$: dependent type theory with Π-, Σ-, unit, Id-, W-types, finite sums, homotopy-coequalisers, and an ω-hierarchy of univalent universes closed under these.

Yet more meta-theorems

$\mathsf{T}_{\mathrm{HoTT}}$: dependent type theory with Π-, Σ-, unit, Id-, W-types, finite sums, homotopy-coequalisers, and an ω-hierarchy of univalent universes closed under these.

Theorem(?) (straightforward(???) extension of C./S.)

The syntax of $\mathsf{T}_{\mathrm{HoTT}}$ *presents the initial contextual category with suitable logical structure.*

Yet more meta-theorems

$\mathsf{T}_{\mathrm{HoTT}}$: dependent type theory with Π-, Σ-, unit, Id-, W-types, finite sums, homotopy-coequalisers, and an ω-hierarchy of univalent universes closed under these.

Theorem(?) (straightforward(???) extension of C./S.)

The syntax of $\mathsf{T}_{\mathrm{HoTT}}$ *presents the initial contextual category with suitable logical structure.*

Theorem(?) (straightforward(???) extension of H.)

The logical framework embedding $\mathsf{T}_{\mathrm{HoTT}} \longrightarrow \mathsf{T}_{LF[\mathrm{HoTT}]}$ *is conservative.*

Yet more meta-theorems

$\mathsf{T}_{\mathrm{HoTT}}$: dependent type theory with Π-, Σ-, unit, Id-, W-types, finite sums, homotopy-coequalisers, and an ω-hierarchy of univalent universes closed under these.

Theorem(?) (straightforward(???) extension of C./S.)

The syntax of $\mathsf{T}_{\mathrm{HoTT}}$ *presents the initial contextual category with suitable logical structure.*

Theorem(?) (straightforward(???) extension of H.)

The logical framework embedding $\mathsf{T}_{\mathrm{HoTT}} \longrightarrow \mathsf{T}_{LF[\mathrm{HoTT}]}$ *is conservative.*

Theorem(?) (straightforward(???) extension of H./L.–W.)

$\mathbf{C}$ *a comprehension category with pseudo-stable* Π-, Σ-, *etc. structure. Then the CwA* $\mathbf{C}_*$ *carries strictly stable* Π-, Σ-, *etc. structure.*

General project

Hope to generalise all of these (and other theorems/constructions) — to allow statements like:

Dream

For any dependent type theory T*, the syntax of* T *presents the initial contextual category with* T*-structure.*

General project

Hope to generalise all of these (and other theorems/constructions) — to allow statements like:

Dream

For any dependent type theory T*, the syntax of* T *presents the initial contextual category with* T*-structure.*

- Avoid handwaving "straightforward" extensions from toy systems to much larger cases
- Give single formalisation that provides results/constructions off-the-shelf for your new extension of type theory
- Articulate precise assumptions required in results/constructions, e.g. "For any extension of ITT axiomatisable without further judgemental equality rules …"

Hard part: not the proofs, but the definition of general type theories.

Why not LF/PTS?

Why isn't the logical framework a satisfactory solution? Or pure type systems, or other existing setups?

Why not LF/PTS?

Why isn't the logical framework a satisfactory solution? Or pure type systems, or other existing setups?

0. For many purposes, they are totally satisfactory!

Why not LF/PTS?

Why isn't the logical framework a satisfactory solution? Or pure type systems, or other existing setups?

0. For many purposes, they are totally satisfactory! But not all:
1. Don't give exactly the type theories/structures we expected
2. Justification depends in part on Hofmann's conservativity theorem—one of the results we want to generalise!
3. Still quite non-trivial to carve out generality/conditions for the theorems above.
4. Unclear how to give account of *weak* structure with LF (e.g. pseudo-stable, strictly stable).

The syntax–semantics spectrum

Long road from `foo.v` to mathematical models…

The syntax–semantics spectrum

Long road from `foo.v` to mathematical models…

- Concrete syntax: your Coq files
- (…lexing, parsing…) Sugary abstract syntax: Coq's internal representation
- (…desugaring, elaboration…) Fully elaborated, conceptually minimal abstract syntax: as in most theory literature

The syntax–semantics spectrum

Long road from `foo.v` to mathematical models…

- Concrete syntax: your Coq files
- (…lexing, parsing…) Sugary abstract syntax: Coq's internal representation
- (…desugaring, elaboration…) Fully elaborated, conceptually minimal abstract syntax: as in most theory literature
- (…initiality theorems…) Strict algebraic models: contextual categories, categories with attributes…
- (…general coherence theorems…) Weak algebraic models: comprehension categories, …
- (…categorical analysis, ad hoc coherence constructions…) Natural mathematical settings/examples: toposes, Quillen model categories, …

The syntax–semantics spectrum

Long road from `foo.v` to mathematical models…

- Concrete syntax: your Coq files
- (…lexing, parsing…) Sugary abstract syntax: Coq's internal representation
- (…desugaring, elaboration…) Fully elaborated, conceptually minimal abstract syntax: as in most theory literature
- (…initiality theorems…) Strict algebraic models: contextual categories, categories with attributes…
- (…general coherence theorems…) Weak algebraic models: comprehension categories, …
- (…categorical analysis, ad hoc coherence constructions…) Natural mathematical settings/examples: toposes, Quillen model categories, …

Hope all levels should be generalised! — eventually.

Focus for today: where syntax and semantics meet.

The syntax–semantics spectrum

Long road from `foo.v` to mathematical models…

- Concrete syntax: your Coq files
- (…lexing, parsing…) Sugary abstract syntax: Coq's internal representation
- (…desugaring, elaboration…) Fully elaborated, conceptually minimal abstract syntax: as in most theory literature
- (…initiality theorems…) Strict algebraic models: contextual categories, categories with attributes…
- (…general coherence theorems…) Weak algebraic models: comprehension categories, …
- (…categorical analysis, ad hoc coherence constructions…) Natural mathematical settings/examples: toposes, Quillen model categories, …

Hope all levels should be generalised! — eventually.

Focus for today: where syntax and semantics meet.

Today's special

We propose a general definition of syntactic dependent type theories, which:

- includes the specific example type theories above
- suffices for the example theorems/constructions above
- is reasonably natural
- is as conventional as possible, in specific cases

(Joint work with Bauer, Haselwarter, Winterhalter.)

Today's special

We propose a general definition of syntactic dependent type theories, which:

- includes the specific example type theories above
- suffices for the example theorems/constructions above
- is reasonably natural
- is as conventional as possible, in specific cases

(Joint work with Bauer, Haselwarter, Winterhalter.)

One major restriction: consider only Martin-Löf's judgement forms (contexts, types, terms, $\equiv$) and structural rules.

Important systems thus not covered: Pure Type Systems; Calculus of Inductive Constructions; Cubical TT; anything linear…

Today's special

We propose a general definition of syntactic dependent type theories, which:

- includes the specific example type theories above
- suffices for the example theorems/constructions above
- is reasonably natural
- is as conventional as possible, in specific cases

(Joint work with Bauer, Haselwarter, Winterhalter.)

One major restriction: consider only Martin-Löf's judgement forms (contexts, types, terms, $\equiv$) and structural rules.

Important systems thus not covered: Pure Type Systems; Calculus of Inductive Constructions; Cubical TT; anything linear…

Expected semantic counterpart: a corresponding class of essentially algebraic extensions of contextual categories.

(A closely related definition has been given independently by Isaev.)

Syntax

General outline

In practice, a type theory is specified by:

- signature for raw syntax (symbols, arities with binding);
- rules for the typing judgements

Signature: definitions already established (Aczel, Belo; Ahrens, Matthes, Uustalu); quite clean, straightforward.

Interesting part: what are rules?

Rules vs. closure condition

Key distinction: rules vs. closure conditions.

Rule as you write it down:

$$\frac{\vdash \Gamma \ \mathsf{cxt} \qquad \Gamma \vdash A \ \mathsf{type} \qquad \Gamma,\ x{:}A \vdash B \ \mathsf{type}}{\Gamma \vdash \Pi_{x:A} B \ \mathsf{type}}$$

Interpretation of rule as closure condition on judgement relations:

Given any

raw context Γ, s.t. $\vdash \Gamma$ cxt is derivable,
raw type A, s.t. $\Gamma \vdash A$ type is derivable,
raw type B, s.t. $\Gamma,\ x{:}A \vdash B$ type is derivable,

---

then $\Gamma \vdash \Pi_{x:A}$ type is derivable.

Rules vs. closure condition

Key distinction: rules vs. closure conditions.

Rule as you write it down:

$$\frac{\vdash \Gamma \text{ cxt} \qquad \Gamma \vdash A \text{ type} \qquad \Gamma,\ x{:}A \vdash B \text{ type}}{\Gamma \vdash \Pi_{x:A} B \text{ type}}$$

Interpretation of rule as closure condition on judgement relations:

Given any

raw context Γ,	s.t. $\vdash \Gamma$ cxt is derivable,
raw type A,	s.t. $\Gamma \vdash A$ type is derivable,
raw type B,	s.t. $\Gamma,\ x{:}A \vdash B$ type is derivable,

then $\Gamma \vdash \Pi_{x:A}$ type is derivable.

Common explanation: the formal thing is the closure condition; "rule" is just informal notation.

We take the "rule" more seriously: a syntactic object we recognise and typecheck; specification of a type theory is a family of rules.

Any rule gives rise to a closure condition, used to define the typing judgements.

Well-typedness of rules

Can't have arbitrary expressions in rules: must type-check suitably.

$$\frac{\vdash \Gamma\ \mathsf{cxt} \qquad \Gamma \vdash A\ \mathsf{type} \qquad \Gamma,\ x{:}A \vdash B\ \mathsf{type} \qquad \Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \lambda x{:}A.\ b : \Pi_{x:B} A}$$

Well-typedness of rules

Can't have arbitrary expressions in rules: must type-check suitably.

$$\frac{\vdash \Gamma\ \mathsf{cxt} \qquad \Gamma \vdash A\ \mathsf{type} \\ \Gamma,\ x{:}A \vdash B\ \mathsf{type} \\ \Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \lambda x{:}A.\ b : \cancel{\Pi_{x:B} A}\ \ \Pi_{x:A} B}$$

Well-typedness of rules

Can't have arbitrary expressions in rules: must type-check suitably.

$$\frac{\vdash \Gamma\ \mathsf{cxt} \qquad \Gamma \vdash A\ \mathsf{type} \\ \Gamma,\ x{:}A \vdash B\ \mathsf{type} \\ \Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \lambda x{:}A.\, b : \cancel{\Pi_{x:B} A}\ \ \Pi_{x:A} B}$$

Note: to type-check conclusion of Π-ABSTR, need to use Π-FORM.

So: put well-ordering on rules. Type-check later rules over earlier ones.

Ambient contexts

Intend all rules to hold over arbitrary contexts. (E.g. “necessitation” rule not covered.)

$$\frac{\vdash A\ \mathsf{type} \qquad x{:}A \vdash B\ \mathsf{type}}{\vdash \Pi_{x:A} B\ \mathsf{type}}$$

Ambient contexts

Intend all rules to hold over arbitrary contexts. (E.g. "necessitation" rule not covered.)

$$\frac{\vdash \Gamma\ \mathsf{cxt} \qquad \Gamma \vdash A\ \mathsf{type} \qquad \Gamma,\ x{:}A \vdash B\ \mathsf{type}}{\Gamma \vdash \Pi_{x:A} B\ \mathsf{type}}$$

Ambient contexts

Intend all rules to hold over arbitrary contexts. (E.g. "necessitation" rule not covered.)

$$\frac{\vdash \Gamma \ \mathsf{cxt} \qquad \Gamma \vdash A \ \mathsf{type} \qquad \Gamma,\ x{:}A \vdash B \ \mathsf{type}}{\Gamma \vdash \Pi_{x:A} B \ \mathsf{type}}$$

Γ assumed for every rule. Plays no rôle in typechecking the rules.

Conclusion: It's part of implementation as closure condition, not specification of the rule itself.

Ambient contexts

Intend all rules to hold over arbitrary contexts. (E.g. "necessitation" rule not covered.)

$$\frac{\vdash A\ \mathsf{type} \qquad x{:}A \vdash B\ \mathsf{type}}{\vdash \Pi_{x:A} B\ \mathsf{type}}$$

Γ assumed for every rule. Plays no rôle in typechecking the rules.

Conclusion: It's part of implementation as closure condition, not specification of the rule itself.

Omitting ambient context in writing rules—not just abuse of notation!

Metavariables

Closure conditions involve metavariables. What represents these in syntax of rules?

$$\frac{\vdash A\ \mathsf{type} \qquad x{:}A \vdash B \quad \mathsf{type} \qquad \vdash f : \Pi_{x:A}B \qquad \vdash a : A}{\vdash \mathrm{app}_{x:A.B}\ \ (f, a) : B[a/x]}$$

Metavariables

Closure conditions involve metavariables. What represents these in syntax of rules?

$$\frac{\vdash A \text{ type} \qquad x{:}A \vdash B \quad \text{type} \qquad \vdash f : \Pi_{x:A}B \qquad \vdash a : A}{\vdash \mathrm{app}_{x:A.B} \quad (f, a) : B[a/x]}$$

Approach 1: specific syntactic entity.

- then also need explicit substitution, and rules for that;
- essentially duplicating machinery of variable-handling, substitution…

Metavariables

Closure conditions involve metavariables. What represents these in syntax of rules?

$$\frac{\vdash A\ \mathsf{type} \qquad x{:}A \vdash B \quad \mathsf{type} \qquad \vdash f : \Pi_{x:A} B \qquad \vdash a : A}{\vdash \mathrm{app}_{x:A.B}\ (f, a) : B[a/x]}$$

Approach 2: symbols in a temporarily-extended signature.

- re-uses existing machinery
- substitution appears in implementation as closure condition
- requires translating syntax between different signatures (but that'd be needed later anyway)

Metavariables

Closure conditions involve metavariables. What represents these in syntax of rules?

$$\frac{\vdash A \text{ type} \qquad x{:}A \vdash B(x) \text{ type} \qquad \vdash f : \Pi_{x:A} B(x) \qquad \vdash a : A}{\vdash \mathrm{app}_{x:A.B(x)}(f, a) : B(a)}$$

Approach 2: symbols in a temporarily-extended signature.

- re-uses existing machinery
- substitution appears in implementation as closure condition
- requires translating syntax between different signatures (but that'd be needed later anyway)

Definition, stage 1

Ready now for full definition, in several stages.

Definitions

- A (binding) arity: a finite set (“arguments”), each marked with a syntactic class (type/term) and a finite set (“bound variables”).[1]
- A (binding) signature: a set (“symbols”), each with a syntactic class and an arity.

[1] Or use natural numbers instead of finite sets… Treatment of variables suppressed today.

Definition, stage 1

Ready now for full definition, in several stages.

Definitions

- A (binding) arity: a finite set (“arguments”), each marked with a syntactic class (type/term) and a finite set (“bound variables”).[1]
- A (binding) signature: a set (“symbols”), each with a syntactic class and an arity.

Definitions

- raw syntax over a signature
- substitution on raw syntax
- translation of raw syntax along signature morphisms
- basic properties of all these

[1] Or use natural numbers instead of finite sets… Treatment of variables suppressed today.

Definition, stage 2

Definition

A raw rule of arity a, over signature Σ:

- premises: well-ordered collection of raw judgements
- conclusion: single raw judgement
- all over $\Sigma + a$: extension of Σ with symbols for arguments of a
- each argument of a introduced by a unique premise

Definition, stage 2

Definition

A raw rule of arity a, over signature Σ:

- premises: well-ordered collection of raw judgements
- conclusion: single raw judgement
- all over $\Sigma + a$: extension of Σ with symbols for arguments of a
- each argument of a introduced by a unique premise

Definitions

- instantiation of an arity over a signature and a raw context
- given instantiation of a over Σ: translation of raw syntax over $\Sigma + a$ to over Σ (using substitution!)
- implementation of raw rule as closure condition: for every instantiation of its arity, if translations of premises hold, so does translation of conclusion
- derivability over a collection of raw rules

Definition, stage 3

Definition

Raw rule is well-typed relative to a collection T of raw rules if:

- presuppositions of each premise are derivable over T plus earlier premises
- presuppositions of conclusion are derivable over T

(Subtlety: not really over T, but over its translation from Σ to $\Sigma + a$.)

[2] i.e. constructively well-founded partial order

Definition, stage 3

Definition

Raw rule is well-typed relative to a collection T of raw rules if:

- presuppositions of each premise are derivable over T plus earlier premises
- presuppositions of conclusion are derivable over T

(Subtlety: not really over T, but over its translation from Σ to $\Sigma + a$.)

Definition

A fully verbose type theory:

- signature;
- well-ordered[2] collection of raw rules,
- each raw rule well-typed over earlier raw rules of collection
- each symbol introduced by a unique rule of correct arity;
- each symbol also has suitable congruence rule.

[2] i.e. constructively well-founded partial order

Main definition, version 1

Definition

An (economically specified) type theory:

- signature;
- well-ordered collection of raw rules,
- each raw rule well-typed over
  - standard structural core,
  - earlier raw rules of collection,
  - associated congruence rules for all type/term rules;
- 1-1 correspondence: each symbol of signature introduced by a unique type/term rule

Reflections on version 1

Success: have “cut the knot” of dependency between rules, signatures, typing, etc.

Reflections on version 1

Success: have "cut the knot" of dependency between rules, signatures, typing, etc.

Issue: make substantial use of equality reasoning, inconvenient in formalisation:

- Morphisms of signatures, for translating between base and extended syntaxes
- "Each argument introduced by a unique premise" within each rule
- "Each symbol introduced by a unique rule" in overall type theory

Refined approach: cut the knot differently to avoid equality reasoning.

Main definition, version 2

Back up a bit; remove redundant information from raw rules.

Definition

A prepped rule of arity a, over Σ, with conclusion form j:

- an arity a', enumerating the equality premises;
- a well-founded relation on $a + a'$ (the collection of all premises);
- for each premise: a raw judgement boundary, over Σ extended by symbols for earlier type/term premises;
- (heads of type/term premises then taken as the corresponding symbols of $\Sigma + a$, applied to their binding variables);
- for conclusion: a raw judgement boundary of form j, over $\Sigma + a$;
- (once rule included in a TT: conclusion head taken as a corresponding symbol of the signature, applied to the generic arguments of a)

A prepped rule is well-typed if the resulting raw rule is.

Main definition, version 2

Definition

A type theory:

- a family $\mathcal{R}$ of arities, labelled with judgement forms (indexing the rules);
- a well-founded relation on $\mathcal{R}$;
- for each $r : \mathcal{R}$ with arity a and judgement form j:
  - a prepped rule of arity a, with conclusion form j,
  - over the signature $\Sigma_{<r}$: symbols given by the earlier type/term rules of $\mathcal{R}$,
  - well-typed over
    - standard structural core,
    - earlier raw rules of $\mathcal{R}$,
    - associated congruence rules for all type/term rules.

Main definition, version 2

Definition

A type theory:

- a family $\mathcal{R}$ of arities, labelled with judgement forms (indexing the rules);
- a well-founded relation on $\mathcal{R}$;
- for each $r : \mathcal{R}$ with arity a and judgement form j:
  - a prepped rule of arity a, with conclusion form j,
  - over the signature $\Sigma_{<r}$: symbols given by the earlier type/term rules of $\mathcal{R}$,
  - well-typed over
    - standard structural core,
    - earlier raw rules of $\mathcal{R}$,
    - associated congruence rules for all type/term rules.

No use of equality reasoning!

Arguably maybe further from practice than first version.

Summary

- Motivation: give generality for meta-theorems of type theory

Summary

- Motivation: give generality for meta-theorems of type theory
- Have: proposed definition of general (syntactic) type theories

Summary

- Motivation: give generality for meta-theorems of type theory
- Have: proposed definition of general (syntactic) type theories
- Work in progress: formalisation of this definition (Bauer–Haselwarter–Lumsdaine–Winterhalter)

Summary

- Motivation: give generality for meta-theorems of type theory
- Have: proposed definition of general (syntactic) type theories
- Work in progress: formalisation of this definition (Bauer–Haselwarter–Lumsdaine–Winterhalter)
- Have: ≥ 2 proposed definitions of general (semantic/algebraic) type theories (certain ess. alg. extensions of theory of CwA's) (proposals by Isaev, Lumsdaine, …?)

Summary

- Motivation: give generality for meta-theorems of type theory
- Have: proposed definition of general (syntactic) type theories
- Work in progress: formalisation of this definition (Bauer–Haselwarter–Lumsdaine–Winterhalter)
- Have: ≥ 2 proposed definitions of general (semantic/algebraic) type theories (certain ess. alg. extensions of theory of CwA's) (proposals by Isaev, Lumsdaine, …?)
- Work in progress: correspondence between these semantic and syntactic definitions

Summary

- Motivation: give generality for meta-theorems of type theory
- Have: proposed definition of general (syntactic) type theories
- Work in progress: formalisation of this definition (Bauer–Haselwarter–Lumsdaine–Winterhalter)
- Have: ≥ 2 proposed definitions of general (semantic/algebraic) type theories (certain ess. alg. extensions of theory of CwA's) (proposals by Isaev, Lumsdaine, …?)
- Work in progress: correspondence between these semantic and syntactic definitions
- Future work: prove motivating meta-theorems in generality

Summary

- Motivation: give generality for meta-theorems of type theory
- Have: proposed definition of general (syntactic) type theories
- Work in progress: formalisation of this definition (Bauer–Haselwarter–Lumsdaine–Winterhalter)
- Have: ≥ 2 proposed definitions of general (semantic/algebraic) type theories (certain ess. alg. extensions of theory of CwA's) (proposals by Isaev, Lumsdaine, …?)
- Work in progress: correspondence between these semantic and syntactic definitions
- Future work: prove motivating meta-theorems in generality
- Further work: generalise the judgement forms considered

Summary

- Motivation: give generality for meta-theorems of type theory
- Have: proposed definition of general (syntactic) type theories
- Work in progress: formalisation of this definition (Bauer–Haselwarter–Lumsdaine–Winterhalter)
- Have: ≥ 2 proposed definitions of general (semantic/algebraic) type theories (certain ess. alg. extensions of theory of CwA's) (proposals by Isaev, Lumsdaine, …?)
- Work in progress: correspondence between these semantic and syntactic definitions
- Future work: prove motivating meta-theorems in generality
- Further work: generalise the judgement forms considered
- Further work: connect general definitions to other points on syntax–semantics spectrum: more usable syntaxes, more natural semantics