A general definition of dependent type theories

jww Andrej Bauer, Philipp Haselwarter

Peter LeFanu Lumsdaine

Stockholms universitet

EUTypes 2018, Aarhus

Motivation, outline

Goal

- Define a general class of dependent type theories,
- flexible enough to cover MLTT and various extensions,
- structured enough to admit key theorems/constructions.

Motivation, outline

Goal

- Define a general class of dependent type theories,
- flexible enough to cover MLTT and various extensions,
- structured enough to admit key theorems/constructions.

Standard template from other logics, not yet available in dependent type theory.

Not trying to be the final word, cover “all type theories”; just “Martin-Löf-like theories”, including extensional+intensional MLTT, CoC, book HoTT, …

Motivation, outline

Goal

- Define a general class of dependent type theories,
- flexible enough to cover MLTT and various extensions,
- structured enough to admit key theorems/constructions.

Standard template from other logics, not yet available in dependent type theory.

Not trying to be the final word, cover “all type theories”; just “Martin-Löf-like theories”, including extensional+intensional MLTT, CoC, book HoTT, …

Outline

1. Definition
2. Detailed motivation, discussion, payoffs

Much of this formalised in Coq (ongoing work).

Background

Work in a minimal meta-theory:

- constructive type theory with inductive families, one universe;
- agnostic about equality: compatible with UIP, or univalence;
- compatible with reading as (constructive) set theory.

Background

Work in a minimal meta-theory:

- constructive type theory with inductive families, one universe;
- agnostic about equality: compatible with UIP, or univalence;
- compatible with reading as (constructive) set theory.

One consequence: use families instead of (sub)sets throughout.

Definition

A family T of elements of X is a type I, with a function $I \longrightarrow X$.
Notation $\langle x_i \rangle_{i:I}$, and variants.

E.g. classically, a propositional theory just a set of sentences. For us, would be a family of sentences $\langle \varphi_i \rangle_{i:I}$.

Families better in absence of choice/LEM: doesn't require flattening index set, so retains decidable equality, etc. for constructions over index set. Tradeoff: some size issues; but not really problematic ones.

Raw syntax 1: shapes, arities

Treatment of variables: de Bruijn indices, named variables, …?

Raw syntax 1: shapes, arities

Treatment of variables: de Bruijn indices, named variables, …?

We axiomatise a common interface to these: roughly, a category of proto-contexts/shapes, with a coproduct-preserving functor to **Set**. Idea: shape = just the variables of a context, no types yet.

Named variables: shapes are $\mathcal{P}_{\text{fin}}(\text{Ident})$. De Bruijn: shape are $\mathbb{N}$.

For remainder of today: fix a shape-system.

Raw syntax 1: shapes, arities

Treatment of variables: de Bruijn indices, named variables, …?

We axiomatise a common interface to these: roughly, a category of proto-contexts/shapes, with a coproduct-preserving functor to **Set**. Idea: shape = just the variables of a context, no types yet.

Named variables: shapes are $\mathcal{P}_{\text{fin}}(\text{Ident})$. De Bruijn: shape are $\mathbb{N}$.

For remainder of today: fix a shape-system.

Also, fix syntactic classes as just $\{\text{ty}, \text{tm}\}$.

Raw syntax 1: shapes, arities

Treatment of variables: de Bruijn indices, named variables, …?

We axiomatise a common interface to these: roughly, a category of proto-contexts/shapes, with a coproduct-preserving functor to **Set**. Idea: shape = just the variables of a context, no types yet.

Named variables: shapes are $\mathcal{P}_{\text{fin}}(\text{Ident})$. De Bruijn: shape are $\mathbb{N}$.

For remainder of today: fix a shape-system.

Also, fix syntactic classes as just $\{\text{ty}, \text{tm}\}$.

Definition

An arity: a family of pairs of syntactic classes and shapes.

Idea: arguments of a symbol, with classes and bound variables specified.

- Arity of Π: $\langle(\text{ty}, 0), (\text{ty}, 1)\rangle$.
- Arity of λ (fully-annotated): $\langle(\text{ty}, 0), (\text{ty}, 1), (\text{tm}, 1)\rangle$.

Raw syntax 2: signatures, substitution

Definition

A signature Σ: a family of pairs of (arity, syntactic class). Elements of the family symbols.

A signature map $f : \Sigma \longrightarrow \Sigma'$: function sending each symbol of Σ to a symbol of Σ' with same arity.

Raw syntax 2: signatures, substitution

Definition

A signature Σ: a family of pairs of (arity, syntactic class). Elements of the family symbols.

A signature map $f : \Sigma \longrightarrow \Sigma'$: function sending each symbol of Σ to a symbol of Σ' with same arity.

Definition

Given Σ, raw expressions over Σ: sets $\mathrm{expr}^{\mathrm{ty}}_{\Sigma}(\gamma)$, $\mathrm{expr}^{\mathrm{tm}}_{\Sigma}(\gamma)$ for all shapes γ, mutually inductively defined families.

Functorial in both γ and Σ (as are most constructions below).

Raw syntax 2: signatures, substitution

Definition

A signature Σ: a family of pairs of (arity, syntactic class). Elements of the family symbols.

A signature map $f : \Sigma \longrightarrow \Sigma'$: function sending each symbol of Σ to a symbol of Σ' with same arity.

Definition

Given Σ, raw expressions over Σ: sets $\mathrm{expr}^{\mathrm{ty}}_{\Sigma}(\gamma)$, $\mathrm{expr}^{\mathrm{tm}}_{\Sigma}(\gamma)$ for all shapes γ, mutually inductively defined families.

Functorial in both γ and Σ (as are most constructions below).

Definition

Simultaneous capture-free substitution: for $t : \mathrm{expr}^{\mathrm{cl}}_{\Sigma}(\gamma)$, and $s_i \in \mathrm{expr}^{\mathrm{tm}}_{\Sigma}(\delta)$ for each position i of γ, get $t[s_i/x_i] \in \mathrm{expr}^{\mathrm{cl}}_{\Sigma}(\gamma)$.

Natural in δ, Σ; satisfies usual (monad) laws; etc.

Judgements

5 fixed judgement forms:

- the context judgement form, $\vdash \Gamma \ \mathsf{cxt}$;
- for each syntactic class, object and equality judgement forms:

$$\Gamma \vdash A \ \mathsf{type} \qquad \Gamma \vdash A \equiv A' \qquad \Gamma \vdash a : A \qquad \Gamma \vdash a \equiv a' : A$$

Judgements

5 fixed judgement forms:

- the context judgement form, $\vdash \Gamma\ \mathsf{cxt}$;
- for each syntactic class, object and equality judgement forms:

$$\Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash A \equiv A' \qquad \Gamma \vdash a : A \qquad \Gamma \vdash a \equiv a' : A$$

Definition

A judgement over Σ: a judgement form; a shape γ; and suitable expressions over Σ in variables γ, for each slot of the form.

(So "judgement" = "potential judgement", not "derivable judgement".)

Judgements

5 fixed judgement forms:

- the context judgement form, $\vdash \Gamma \ \mathrm{cxt}$;
- for each syntactic class, object and equality judgement forms:

$$\Gamma \vdash A \ \mathrm{type} \qquad \Gamma \vdash A \equiv A' \qquad \Gamma \vdash a : A \qquad \Gamma \vdash a \equiv a' : A$$

Definition

A judgement over Σ: a judgement form; a shape γ; and suitable expressions over Σ in variables γ, for each slot of the form.

(So "judgement" = "potential judgement", not "derivable judgement".)

Definition

A judgement boundary: like a hypothetical judgement, except with no head expression if the form is an object judgement.

$$\Gamma \vdash \_ \ \mathrm{type} \qquad \Gamma \vdash A \equiv A' \qquad \Gamma \vdash \_ : A \qquad \Gamma \vdash a \equiv a' : A$$

Rules

Goal

Derivability: inductive predicate over judgements, specified by rules.

Rules

Goal

Derivability: inductive predicate over judgements, specified by rules.

Specification of rule:

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

Rules

Goal

Derivability: inductive predicate over judgements, specified by rules.

Specification of rule:

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

Interpretation as closure condition on derivability:

$$\frac{\begin{array}{lll}\text{Given any} & & \\ \text{raw context } \Gamma\text{, s.t.} & \vdash \Gamma \text{ cxt} & \text{is derivable,} \\ \text{raw type } A\text{, s.t.} & \Gamma \vdash A \text{ type} & \text{is derivable,} \\ \text{raw type } B\text{, s.t.} & \Gamma,\ x{:}A \vdash B \text{ type} & \text{is derivable,}\end{array}}{\text{then } \Gamma \vdash \Pi_{x:A} \text{ type is derivable.}}$$

Rules: what even are they???

Goal

Derivability: inductive predicate over judgements, specified by rules.

Specification of rule:

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

Interpretation as closure condition on derivability:

$$\frac{\begin{array}{lll}\text{Given any} & & \\ \text{raw context } \Gamma\text{, s.t.} & \vdash \Gamma \text{ cxt} & \text{is derivable,} \\ \text{raw type } A\text{, s.t.} & \Gamma \vdash A \text{ type} & \text{is derivable,} \\ \text{raw type } B\text{, s.t.} & \Gamma,\ x{:}A \vdash B \text{ type} & \text{is derivable,}\end{array}}{\text{then } \Gamma \vdash \Pi_{x:A} \text{ type is derivable.}}$$

Lots to unpack here!

Abstract derivations, closure conditions

Step back briefly from syntax.

Definition

A closure condition on a type X: a family of elements of X (the premises), and another element of X (the conclusion).

Definition

A closure system on X: a family of closure conditions on X.

Definition

Derivations in a closure system T on X, possibly from a family H of hypothesis: inductive family over X, generated by the hypotheses H and closure conditions of T.

Abstract derivations, closure conditions

Step back briefly from syntax.

Definition

A closure condition on a type X: a family of elements of X (the premises), and another element of X (the conclusion).

Definition

A closure system on X: a family of closure conditions on X.

Definition

Derivations in a closure system T on X, possibly from a family H of hypothesis: inductive family over X, generated by the hypotheses H and closure conditions of T.

Typing derivations will be derivations (in this sense) on judgements, in a closure system specified by rules.

Flat rules

Definition

A flat rule, over a signature Σ:

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

Flat rules

Definition

A flat rule, over a signature Σ:

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

- a family of hypothetical judgements (premises),
- another hypothetical judgement (conclusion),

Flat rules

Definition

A flat rule, over a signature Σ:

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

- a family of hypothetical judgements (premises),
- another hypothetical judgement (conclusion),
- all over a metavariable extension $\Sigma + a$ of Σ.

Flat rules

Definition

A flat rule, over a signature Σ:

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

- a family of hypothetical judgements (premises),
- another hypothetical judgement (conclusion),
- all over a metavariable extension $\Sigma + a$ of Σ.

Definition

Metavariable extension of a signature Σ:

- specified by an arity a, e.g. $\langle (0, \text{ty}), (1, \text{ty}) \rangle$;
- adds a new symbol for each argument of a, taking term arguments for the bound variables of that argument, and binding no variables itself.

Flat rules

Definition

A flat rule, over a signature Σ:

$$\frac{\vdash A \text{ type} \qquad x{:}A \vdash B(x) \text{ type}}{\vdash \Pi_{x:A} B(x) \text{ type}}$$

- an arity a,
- a family of hypothetical judgements (premises),
- another hypothetical judgement (conclusion),
- all over the metavariable extension $\Sigma + a$ of Σ.

Definition

Metavariable extension of a signature Σ:

- specified by an arity a, e.g. $\langle (0, \text{ty}), (1, \text{ty}) \rangle$;
- adds a new symbol for each argument of a, taking term arguments for the bound variables of that argument, and binding no variables itself.

Closure conditions from flat rules

Definition

A flat rule over Σ induces a family of closure conditions on judgements over Σ, indexed by instantiations of a over Σ.

Closure conditions from flat rules

Definition

A flat rule over Σ induces a family of closure conditions on judgements over Σ, indexed by instantiations of a over Σ.

Original rule:

$$\frac{\begin{array}{c}\vdash A \text{ type} \\ x{:}A \vdash B(x) \text{ type} \\ \vdash f : \Pi_{x:A} B(x) \\ \vdash a : A\end{array}}{\vdash \mathrm{app}(A, x.B(x), f, a) : B(a)}$$

Closure conditions from flat rules

Definition

A flat rule over Σ induces a family of closure conditions on judgements over Σ, indexed by instantiations of a over Σ.

Original rule:

$$\frac{\begin{array}{c}\vdash A\ \mathrm{type} \\ x{:}A \vdash B(x)\ \mathrm{type} \\ \vdash f : \Pi_{x:A} B(x) \\ \vdash a : A\end{array}}{\vdash \mathrm{app}(A, x.B(x), f, a) : B(a)}$$

Induced closure conditions:

For each raw context Γ, raw types A, B, and raw terms f, a, over the ambient signature,

$$\frac{\begin{array}{c}\Gamma \vdash A\ \mathrm{type} \\ \Gamma, x{:}A \vdash B\ \mathrm{type} \\ \Gamma \vdash f : \Pi_{x:A} B \\ \Gamma \vdash a : A\end{array}}{\Gamma \vdash \mathrm{app}(A, x.B, f, a) : B[a/x]\ \mathrm{type}}$$

Closure conditions from flat rules

Definition

A flat rule over Σ induces a family of closure conditions on judgements over Σ, indexed by instantiations of a over Σ.

Original rule:

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

Induced closure conditions:

For each raw context Γ, raw types A, B, and raw terms f, a, over the ambient signature,

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

- Flat rule: syntactic object; can be type-checked, translated between signatures…
- Closure conditions: as required to define derivations

Flat type theories

Definition

A flat type theory T over Σ: a family of flat rules thereover.

Definition

Any flat TT induces an associated closure system on judgements over Σ:

- the closure conditions associated to all rules of T;
- the structural rules over Σ.

Flat type theories

Definition

A flat type theory T over Σ: a family of flat rules thereover.

Definition

Any flat TT induces an associated closure system on judgements over Σ:

- the closure conditions associated to all rules of T;
- the structural rules over Σ.

Structural rules

- empty context; context extension
- typing of variables
- renaming under shape isos
- substitutions into judgements
- equal substitutions give equal results
- equality judgements equiv rels
- type-conversion under type equality

Typechecking

Definition

Derivations over a flat TT T: derivations over its associated closure system.

Speak of derivable judgments, typing judgements, well-formed expressions etc over T.

Typechecking

Definition

Derivations over a flat TT T: derivations over its associated closure system.

Speak of derivable judgments, typing judgements, well-formed expressions etc over T.

Are we done?

Typechecking

Definition

Derivations over a flat TT T: derivations over its associated closure system.

Speak of derivable judgments, typing judgements, well-formed expressions etc over T.

Are we done?

Not yet: flat rules can still be pathological.

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

Typechecking

Definition

Derivations over a flat TT T: derivations over its associated closure system.

Speak of derivable judgments, typing judgements, well-formed expressions etc over T.

Are we done?

Not yet: flat rules can still be pathological.

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

Need more structure/properties on rules, and on type theory:

- expressions in rules should typecheck,
- over earlier premises of rules;
- each metavariable introduced by some premise.
- all rules should typecheck in this sense,
- over earlier rules of theory;
- each symbol introduced by some rule.

Well-presented rules

Definition

A well-presented rule, over ambient signature Σ:

- a well-ordered family of judgement boundaries, the premises,
- each over the extension of Σ by metavariables for earlier object premises;
- a conclusion judgement boundary, over the extension of Σ by metas for all object premises.

Well-formed over a flat type theory T if all expressions in boundaries are well-formed over T plus earlier premises.

$$\frac{\begin{array}{ll} \vdash \_ \ \mathsf{type} & A \\ x{:}A \vdash \_\_\_\_ \ \mathsf{type} & B \\ x{:}A \vdash \_\_\_\_ : B(x) & t \end{array}}{\vdash \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ : \Pi(A, x.B(x))}$$

Well-presented rules

Definition

A well-presented rule, over ambient signature Σ:

- a well-ordered family of judgement boundaries, the premises,
- each over the extension of Σ by metavariables for earlier object premises;
- a conclusion judgement boundary, over the extension of Σ by metas for all object premises.

Well-formed over a flat type theory T if all expressions in boundaries are well-formed over T plus earlier premises.

$$\frac{\begin{array}{ll} \vdash \underline{A}\ \mathsf{type} & A \\ x{:}A \vdash \underline{B(x)}\ \mathsf{type} & B \\ x{:}A \vdash \underline{t(x)} : B(x) & t \end{array}}{\vdash\ \underline{?\,(A, x.B(x), x.t(x))} : \Pi(A, x.B(x))}$$

Well-presented type theories

Can realise a w-p rule as a flat rule, given (if conclusion is object form) a head symbol.

Any w-p rule with object-form conclusion has associated congruence rule.

Well-presented type theories

Can realise a w-p rule as a flat rule, given (if conclusion is object form) a head symbol.

Any w-p rule with object-form conclusion has associated congruence rule.

Definition

A well-presented type theory:

- a well-ordered family of well-presented rules, the logical rules,
- each over signature with symbols given by earlier object rules.

Well-formed if each rule well-formed over the flat TT given by realisations of earlier rules + their associated congruence rules.

A type theory: a well-formed, well-presented type theory.

Examples: original Martin-Löf extensional and intensional TT; calculus of constructions; book HoTT; …

Roughly: any DTT with just M-L's original judgement forms, presented with fully annotated syntax.

Definition review

Review

- Raw syntax: standard syntax-with-binding, nothing novel.
- Flat type theories: just what's required for defining typing judgements; roughly what one normally wrties; can be pathological.
- Well-presented type theories: more restricted, structured; very robust, well-behaved; many meta-theorems hold off the shelf.

In some sense: nothing novel; just articulating (a subset of) what we already know.

A bit bureaucratic, but not unmanageably.

Definition review

Review

- Raw syntax: standard syntax-with-binding, nothing novel.
- Flat type theories: just what's required for defining typing judgements; roughly what one normally wrties; can be pathological.
- Well-presented type theories: more restricted, structured; very robust, well-behaved; many meta-theorems hold off the shelf.

In some sense: nothing novel; just articulating (a subset of) what we already know.

A bit bureaucratic, but not unmanageably.

Hoped-for payoffs

- Resolve questionable status of results heuristically understood to work for "all reasonable DTT's", especially categorical semantics, initiality.
- Allow rigorous articulation of hypotheses for results that work for some DTT's.

Metatheorems 1

Basic fitness-for-purpose theorems (proved on paper, formalisations in progress):

Theorem

For any type theory T,

- *presuppositions of derivable judgements are derivable:*
  e.g. if $\Gamma \vdash a : A$, then $\Gamma \vdash A$ type, etc.
- *"subsitution-elimination": the structural rules for substitution can be eliminated from derivations.*
- *"unique typing": if $\Gamma \vdash a : A$ and $\Gamma \vdash a : B$, then $\Gamma \vdash A \equiv B$.*
- *T has a "syntactic category with attributes", constructed the traditional way.*

Metatheorems 2

More interesting results/constructions (expected, not yet all proved):

Goals

For any finitary type theory T:

- T has an associated categorical semantics, i.e. a notion of extra algebraic structure on CwA's, such that its syntactic CwA admits this structure and is the initial such.
- T has a logical framework presentation T^{LF}, and this is conservative over T (following Hofmann's approach).
- The categorical semantics of T admit a version of Hofmann's right adjoint strictification (following the analysis given by Lumsdaine–Warren).
- If T has no type equality rules, then the categorical semantics of T admit a version of Lumsdaine–Warren's left adjoint/local universes strictification.

Comparison to logical framework

Objection

Doesn't the logical framework already give a general definition of type theories?

I.e. define a single "universal" type theory, the LF; then present other TT's as contexts within LF, with binding constructors represented as higher-order functionals, etc.

Comparison to logical framework

Objection

Doesn't the logical framework already give a general definition of type theories?

I.e. define a single "universal" type theory, the LF; then present other TT's as contexts within LF, with binding constructors represented as higher-order functionals, etc.

Responses

0. Yes, and it works well for many purposes!

Comparison to logical framework

Objection

Doesn't the logical framework already give a general definition of type theories?

I.e. define a single "universal" type theory, the LF; then present other TT's as contexts within LF, with binding constructors represented as higher-order functionals, etc.

Responses

0. Yes, and it works well for many purposes!
1. (Pragmatic) Unclear how to generalise some metatheorems via LF.
2. (Philosophical) It's not describing the most naïve/natural/widespread/fundamental reading of presentations of type theories. Naïve reading still needed at least for presentation of the LF itself!
3. (Mathematical-philosophical.) Conservativity over naïve reading justifies equivalence. But: to give that in generality, need to define naïve reading.
4. (Pragmatic) How to present DTT's with infinitely many rules?

Further notes

Related proposals:

- Isaev 2016 *Algebraic presentations of dependent type theories*, arXiv:1602.08504: algebraic presentation, comparable to present definition but not exactly equivalent.
- Capriotti 2017, thesis *Models of type theory with strict equality*, arXiv:1702.04912: categorical presentation, not sure of comparison to present approach.
- Brunerie 2018, unpublished work (personal communication): independently gave definition very similar to the present one.

Further notes

Related proposals:

- Isaev 2016 *Algebraic presentations of dependent type theories*, arXiv:1602.08504: algebraic presentation, comparable to present definition but not exactly equivalent.
- Capriotti 2017, thesis *Models of type theory with strict equality*, arXiv:1702.04912: categorical presentation, not sure of comparison to present approach.
- Brunerie 2018, unpublished work (personal communication): independently gave definition very similar to the present one.

Desired generalisations:

- Cover type theories with more general judgement forms.
- Account for computational aspects, not just denotational.

Summary

Desiderata, achieved

- Define and formalise a general class of dependent type theories,
- flexible enough to cover MLTT and various extensions,
- structured enough to admit key theorems/constructions.
- Prove basic fitness-for-purpose metatheorems.

Summary

Desiderata, achieved

- Define and formalise a general class of dependent type theories,
- flexible enough to cover MLTT and various extensions,
- structured enough to admit key theorems/constructions.
- Prove basic fitness-for-purpose metatheorems.

Next goals

- Categorical semantics/initiality!
- General statements/proofs of other less-straightforward metatheorems
- Provide tractable formalisation library for all this in Coq

Summary

Desiderata, achieved

- Define and formalise a general class of dependent type theories,
- flexible enough to cover MLTT and various extensions,
- structured enough to admit key theorems/constructions.
- Prove basic fitness-for-purpose metatheorems.

Next goals

- Categorical semantics/initiality!
- General statements/proofs of other less-straightforward metatheorems
- Provide tractable formalisation library for all this in Coq

Thankyou!