THE ESSENCE OF TYPE-THEORETIC ELABORATION

joint work with Andrej Bauer
QUICKEST INTRO TO TYPE THEORY

Type theories are **deductive systems**:

- mostly used in proof assistants
- we derive terms of types $t : A$
- types are “theorems”, terms are “proofs”
- think of a programming language with fancy typing system and rules how to derive terms
WHAT IS ELABORATION?
Elaboration =

Transformation into mathematical objects?

Figuring out missing (mathematical) context information?
Elaboration =

Type checking + compilation?
We don’t really know.

Confused, frustrated.
IDEA OF ELABORATION:

Adding missing information
IDEA OF ELABORATION:

Adding missing types

\[ \frac{\vdash A \text{ type} \quad \vdash B \text{ type} \quad x : A \vdash e : B}{\vdash \lambda(x.e) : A \rightarrow B} \]

\[ \vdash A \text{ type} \quad \vdash B \text{ type} \quad x : A \vdash e : B \quad \vdash \lambda(A, B, x.e) : A \rightarrow B \]
IDEA OF ELABORATION:

Adding missing evidence

\[
\Gamma \vdash A : Type \\
\Gamma \vdash B : Type \\
\frac{\Gamma \vdash A \rightarrow B : Type}{\Gamma \vdash A \rightarrow B : Type_{\text{max}(i,k)}}
\]

(universe levels, termination checking)
THE ESSENCE OF ELABORATION

standard type theory (kernel)

inter-)

economic type theory

elaboration map

retrogression transformation (conservative)
A signature of symbols.

4 kinds of judgements.

$\text{A type} \quad a : A \quad A \equiv B \quad a \equiv b : A$

Hypothetical judgements

variable context + metavariable context $\Gamma \vdash \mathcal{J}$

*Type theory = Finitary type theory in the sense of Haselwarter and Bauer.
**TYPE THEORY : INGREDIENTS**

- **Structural rules:** Variable rule, reflexivity, symmetry and transitivity of equations etc.

- **Specific rules:**
  - **Object rules**
    - \( \vdash A \text{ type} \quad \vdash B \text{ type} \)
    - \( \vdash A \rightarrow B \text{ type} \)
    - \( \vdash A \text{ type} \quad \vdash B \text{ type} \quad x : A \vdash e : B \)
    - \( \vdash \lambda (x.e) : A \rightarrow B \)
  - **Equality rules**
    - \( \vdash N \equiv \mathbb{N} \)
    - \( \vdash A \text{ type} \quad \vdash B \text{ type} \quad a : A \quad b : B \)
    - \( \vdash \text{fst}(\text{pair}(a,b)) \equiv a : A \)

- **Congurence rules** (for every object rule).
  - Such that the rules are well-formed (presupposition)
SYMBOL RULES

Compare the two rules.

\[- \text{A type} \quad \text{B type} \quad p : A \times B \]
\[\Rightarrow \quad \text{fst}(p) : A\]

\[- \text{A type} \quad \text{B type} \quad p : A \times B \]
\[\Rightarrow \quad \text{fst}(A, B, p) : A \]

Better for user input.

Faithfully records the (proof-relevant parts of) the premises.
A type theory is **standard** if every object rule is a symbol rule and every symbol has exactly one symbol rule.

Standard type theories are **well behaved**:  
- inversion  
- uniqueness of typing
THE ESSENCE OF ELABORATION

standard type theory (kernel)
elaboration map
economic type theory

r
transformation (conservative)

S

T
Transformations form a relative monad for syntax and preserve derivability.
THE ESSENCE OF ELABORATION

S

T

standard type theory
(kernel)

economic type theory

elaboration map

retrogression
transformation
(conservative)

l
Elaboration map takes a derivation! 

... such that $r_* (\Gamma' \vdash \mathcal{J}') = \Gamma \vdash \mathcal{J}$.

$\ell$ is a witness of surjectivity of $r$.

Elaboration map preserves derivability.
THE ESSENCE OF ELABORATION

standard type theory (kernel)

r

transformation (conservative)

S

T

e

elaboration map

economic type theory

S → T

T → S

r

S ← T
Elaboration map is unique up-to judgemental equality.
Retrogressions are essentially the same: they factor through each other by other retrogressions.

\[ r_2 \circ f = r_1 \]

\( f \) is conservative and unique up-to judgemental equality.
AN ELABORATION THEOREM

Every type theory has “an elaboration”.

For every type theory $T$ there exists a standard type theory $S$ with a retrogression $r : S \rightarrow T$ and elaboration map $\ell : T \rightarrow S$. 
ELABORATOR: ALGORITHM
**Elaborator**: an algorithm

- Takes: judgement J
- Outputs: a derivable elaborated judgement J' if it exists, or reports there is none

An elaborator, if it exists, is *computable* for our chosen type theory.
T has an elaborator if and only if T has decidable type and equality checking.

Elaborator is the most general type-checking algorithm for T, if any exists.
The essence of elaboration

- Universal property
- Elaboration theorem (every economic theory can be elaborated)

Diagram:
- Standard type theory (kernel)
- Economic type theory
- Elaboration map
- Retrogression transformation (conservative)