THE ESSENCE OF ELABORATION

joint work with Andrej Bauer
WHAT IS ELABORATION?
Elaboration =

Type checking + compilation?
Elaboration =

Transformation into mathematical objects?

Figuring out missing (mathematical) context information?
Esa Pulkkinen @PulkkinenEsa · 7 Mar
Repyling to @andrejbauer
My understanding is elaboration refers to transition from high level of abstraction to lower one by incrementally adding missing details. But my impression on this could be mistaken, I do still remember I was confused when I first heard of the concept.

Confused, frustrated.

We don’t really know.

andgate (paranormal software detective) @the_andgate · 7 Mar
Reading papers on type theory is very frustrating, because type theorists almost never provide definitions for their esoteric terms and phrases. “Elaboration” being the most damaging example.
IDEA OF ELABORATION:

Adding missing information

- Adding missing types

\[
\vdash \text{A type} \quad \vdash \text{B type} \quad x : \text{A} \quad \vdash e : \text{B}
\]

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

- Adding missing evidence

(termination checker, universe levels)
Elaboration map - a map $\ell : T \to S$ from an economic type theory to a fully annotated type theory

Elaborator - an algorithm performing adding information (related to type-checking)
QUESTIONS TO ANSWER

- What are the properties of the economic syntax/type theory?
- What are the properties of the fully annotated (kernel) syntax/type theory?
- What is an elaboration map?
- What is the input and output of elaborator?
- How does elaborator relate to type checking?
THE ESSENCE OF ELABORATION

- Standard type theory
- Finitary type theory
- Elaboration map
- Retrogression transformation (conservative)
- Economic
A finitary type theory [HB21] is a formal deductive system that consists of:

- A signature of symbols.
- 4 kinds of judgements.
- Boundaries (for every judgement kind).
- Hypothetical judgements and boundaries.
RULES OF FINITARY TYPE THEORY

A finitary type theory [HB21] is a formal deductive system that consists of:

- **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**
    
    \[ \vdash N \equiv \mathbb{N} \]
    
    \[ \vdash A \text{ type} \quad \vdash B \text{ type} \quad a : A \quad b : B \]
    
    \[ \vdash \text{fst}(\text{pair}(a,b)) \equiv a : A \]

- **Congruence rules** (for every object rule).

Such that the rules have well-formed boundaries (presuppositivity).
Compare the two rules.

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

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

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

finitary type theory (economic)

r

retrogression transformation (conservative)

S

T

l
Recall from Andrej Bauer’s talk:

\[ f : \mathcal{U} \longrightarrow \mathcal{T} \]

\[ S \quad \text{a symbol from signature of } \mathcal{U} \]

\[ e \quad \text{a well-formed expression in } \mathcal{T} \]

The syntactic transformation \( f \) acts on expressions \( e' \) in \( \mathcal{U} \) to produce expressions \( f \_e' \) in \( \mathcal{T} \).

Syntactic transformations form a **relative monad** for syntax.
A type-theoretic transformation is a syntactic transformation $f : U \to T$ such that for every specific rule in $U$ there is a derivation of

$$\frac{P_1 \quad \cdots \quad P_n}{\vdash \mathcal{J}}$$

in $U$ there is a derivation of

$$\frac{f^* P_1 \quad \cdots \quad f^* P_n}{\vdash f^* \mathcal{J}}$$

in $T$. Type-theoretic transformations preserve derivability.
The original could also be filled (derivably) if it can be (derivably) filled with a head.
THE ESSENCE OF ELABORATION

S  \( \rightarrow \) r  \( \leftarrow \) T

standard type theory

S  \( \leftarrow \) \( \ell \)  \( \rightarrow \) T

elaboration map

finitary type theory

r  \( \leftarrow \) retrogression transformation (conservative)

(conservative)

(economic)
Side note: elaboration map works uniformly on contexts and boundaries \( \ell(\Gamma', \mathcal{B}', \mathcal{D}) \)

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

\( \ell \) is a section of \( r \).

Elaboration map preserves derivability.
THE ESSENCE OF ELABORATION

standard type theory (kernel)

S

elaboration map

T

finitary type theory (economic)

retrogression transformation (conservative)

L

(conservative)
Retrogression transformation is surjective on derivable judgements.
Elaboration map is unique up-to judgemental equality.
Universal Property

Elaboration map satisfies the following universal property:

\[ r_2 \circ f = r_1 \]

f is conservative and unique up-to judgemental equality.
Every finitary type theory has “an elaboration”.

For every finitary type theory $T$ there exists a standard type theory $S$ with a retrogression transformation $r : S \rightarrow T$ and elaboration map $\ell : T \rightarrow S$. 
PROOF IDEA
(Elaboration theorem)

For every **specific object rule** $R_i = \frac{P_1 \ldots P_n}{\vdash B \in e}$ introduce a symbol $S_{(i, R_i)}$.

Retrogression transformation (syntactic part) $r : S_{(i, R_i)} \rightarrow e$

$i$ is the index of the rule in the signature
Specific rules of standard type theory S:

Specific object rules of T $\rightarrow$ Symbol rules in S

Specific equality rules of T $\rightarrow$ Specific equality rules in S + close under derivability

Do this inductively on the ordering of rules.
PROOF IDEA
(Elaboration theorem)

Define elaboration map inductively.

Prove desired properties of retrogression transformation (conservativity, type-theoretic transformation) and of elaboration map (section, preserves derivability).
AN ELABORATION THEOREM

For every finitary type theory $T$ there exists a standard type theory $S$ with a retrogression transformation $r : S \rightarrow T$ and elaboration map $\ell : T \rightarrow S$.

But $S$ has a looooot of specific equality rules!

Recall: universal property
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.
CHECKING

Type-checking:
Check that a term $a$ has type $A$.
Check that the head $a$ fits the boundary $\Box : A$.

Equality-checking:
Check that $A \equiv B$ (or $a \equiv b : A$).
Check that the head $\bigstar$ fits the boundary $A \equiv B$ by $\Box$ (or $a \equiv b : A$ by $\Box$).

Checking:
Check that the head $e$ fits the boundary $\Box$ to get the judgement $\Box e$. 

Derivable boundary! * in equation-free meta context
T has an elaborator if and only if T has decidable (judgement) checking.

Elaborator is the most general checking algorithm for T, if any exists.
If a standard type theory has decidable equality checking, then it has decidable checking.

If a finitary type theory has decidable (equality) checking, so does its elaborated standard type theory.

Note: the converse does not hold!
THE ESSENCE OF ELABORATION

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

S

∀ elaboration map

r

retrogression transformation (conservative)

T

∀ economic

standard type theory (kernel)

∀ elaboration map

finitary type theory (economic)
We have an example of a type theory such that:

(a) Checking is semidecidable.
(b) Equality checking is decidable.
(c) Checking is not decidable.

\( D \subseteq \mathbb{N} \times \mathbb{N} \) a computable subset, such that

\[ \pi_1(D) = \{ n \in \mathbb{N} \mid \exists m \in \mathbb{N}. (n, m) \in D \} \] is semidecidable, not computable.

A signature is given by \((A_n : \text{Type})_n \in \mathbb{N}\)

Rules: for every \((n, m) \in D\)
\[ R(n, m) = \vdash A_n \text{ type} \]

Derivable boundary \([\square] \vdash \square \text{ type}\)

Check if \([\square] \vdash A_n \text{ type} \) is derivable?