Algebraic simple type theory
A polynomial approach

Nathanael Arkor & Marcelo Fiore
Department of Computer Science and Technology
University of Cambridge

Category Theory 2019
What is a type theory?
What is a type theory?
What is a type theory?

- System of mathematical proof
- Specification for a programming language
What is a type theory?

- system of mathematical proof
- specification for a programming language
- internal language of a category
Algebraic type theory

categorical algebra

c +

typing / sorting
binding
polymorphism
dependency

•••
Algebraic simple type theory

categorical algebra

typing / sorting
binding
polymorphism
dependency

E.g. λ-calculus, computational λ-calculus, predicate logic
Typing judgements

\[ x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau \]

Examples

Monoid action \[ x : M, a : A \vdash x \cdot a : A \]

Integration \[ f : \mathbb{R} \to \mathbb{R} \vdash \int_x f(x) \, dx : \mathbb{R} \]
\( x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau \)
\( x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau \)
Cartesian context structures

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>$\in \text{Set}$</td>
<td>the sorts, or types</td>
</tr>
<tr>
<td>$\mathcal{C}$</td>
<td>$\in \text{Cat}$</td>
<td>the variable contexts</td>
</tr>
<tr>
<td>$\varepsilon$ (terminal)</td>
<td>$\in \mathcal{C}$</td>
<td>the empty context</td>
</tr>
<tr>
<td>$\langle \text{--} \rangle$</td>
<td>: $S \rightarrow \mathcal{C}$</td>
<td>types as contexts</td>
</tr>
<tr>
<td>$- \times \langle = \rangle$</td>
<td>: $\mathcal{C} \times S \rightarrow \mathcal{C}$</td>
<td>context extension</td>
</tr>
</tbody>
</table>

e.g. any cartesian category
Cartesian context structures

\[ S \in \text{Set} \quad \text{the sorts, or types} \]

\[ C \in \text{Cat} \quad \text{the variable contexts} \]

\[ \varepsilon \text{ (terminal)} \in C \quad \text{the empty context} \]

\[ \langle \cdot \rangle : S \to C \quad \text{types as contexts} \]

\[ - \times \langle = \rangle : C \times S \to C \quad \text{context extension} \]

e.g. any cartesian category
Cartesian context structures

\[ S \in \text{Set} \quad \text{the sorts, or types} \]

\[ C \in \text{Cat} \quad \text{the variable contexts} \]

\[ \varepsilon \ (\text{terminal}) \in C \quad \text{the empty context} \]

\[ \langle - \rangle : S \to C \quad \text{types as contexts} \]

\[ - \times \langle = \rangle : C \times S \to C \quad \text{context extension} \]

e.g. any cartesian category
Cartesian context structures

$S \in \text{Set}$ the sorts, or types

$\mathcal{C} \in \text{Cat}$ the variable contexts

$\varepsilon$ (terminal) $\in \mathcal{C}$ the empty context

\[
\langle - \rangle : S \rightarrow \mathcal{C}
\]
types as contexts

$- \times \langle = \rangle : \mathcal{C} \times S \rightarrow \mathcal{C}$ context extension

e.g. any cartesian category
Cartesian context structures

\[ S \in \text{Set} \quad \text{the sorts, or types} \]

\[ \mathcal{C} \in \text{Cat} \quad \text{the variable contexts} \]

\[ \varepsilon \text{ (terminal)} \in \mathcal{C} \quad \text{the empty context} \]

\[ \langle - \rangle : S \rightarrow \mathcal{C} \quad \text{types as contexts} \]

\[ - \times \langle = \rangle : \mathcal{C} \times S \rightarrow \mathcal{C} \quad \text{context extension} \]

e.g. any cartesian category
\[ x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau \]
Term-typing structure

We consider presheaves $\mathcal{C}^{\text{op}} \to \text{Set}$ on a cartesian context structure $({\mathcal{C}, S})$, fibred over $S$.

$$
\begin{align*}
\mathcal{T}(\Gamma) & \quad \text{presheaf of terms (in context $\Gamma$)} \\
\tau_{\Gamma} & \quad \text{assignment of types to terms} \\
\mathcal{S}(\Gamma) & \quad \text{presheaf of types (in context $\Gamma$)}
\end{align*}
$$
Term-typing structure

We consider presheaves $\mathcal{C}^{\text{op}} \to \text{Set}$ on a cartesian context structure $(\mathcal{C}, S)$, fibred over $S$.

$$\tau_{\Gamma} : \mathcal{T}(\Gamma) \to S$$

- $\mathcal{T}(\Gamma)$: presheaf of terms (in context $\Gamma$)
- $\tau_{\Gamma}$: assignment of types to terms
- $S$: constant presheaf of types
Terms with a specified type

NB. The fibre $T_\sigma$ is the set of terms with type $\sigma$. 

\[ T_\sigma \rightarrow T \\
\downarrow \quad \downarrow \tau \\
1 \quad S \]

\[ \begin{array}{c}
pb \\
\sigma \\
\end{array} \]
Presheaf of variables

For any context $\Gamma \in \mathcal{C}$, $V(\Gamma)$ is the set of variables in $\Gamma$.

$$V \overset{\text{def}}{=} \bigsqcup_{\sigma \in S} y^{\langle \sigma \rangle}$$
Models of simple type theory

\[ \Gamma_1 : \tau_1, \ldots, \Gamma_n : \tau_n \vdash t : \tau \]
Models of simple type theory

\[ x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau \]

- contexts
- terms and types
- algebraic structure
Models of simple type theory

\[
\Gamma \vdash \text{op}(t) : \text{Op}(\tau)
\]

- **Term operators**:
  - \((t_1, t_2)\)
  - \(\pi_1 t\)
  - \(t_1 t_2\)
  - \(\lambda(x) t\)
  - ...

- **Type operators**:
  - \(\tau \rightarrow \sigma\)
  - \(\tau \times \sigma\)
  - \(T(\tau)\)
  - ...

Algebraic structure on types

Type structure is as in universal algebra. For instance, the following operators

\[ \tau \rightarrow \sigma \mid \tau \times \sigma \mid T(\tau) \mid U \]

induce a signature endofunctor on Set

\[ \Sigma_{ty} = X \mapsto X^2 + X^2 + X + 1 \]

the algebras for which are sets \( S \) with the appropriate structure

\[ \llbracket \rightarrow \rrbracket, \llbracket \times \rrbracket, \llbracket T \rrbracket, \llbracket U \rrbracket : \Sigma_{ty} S \rightarrow S \]

(NB. These signature functors are polynomial.)
How should we define the algebraic structure on terms?
How should we define the algebraic structure on terms?

Natural deduction rules present algebraic structure
How should we define the algebraic structure on terms?

Natural deduction rules present algebraic structure.

Polynomials present natural deduction rules.
Polynomials & polynomial functors

In a locally cartesian-closed category $\mathcal{E}$, a polynomial is a diagram:

$$A \leftarrow B \rightarrow C \rightarrow D$$

The polynomial functor associated to the polynomial is given by:

$$\Sigma_h \prod_g f^* : \mathcal{E}/A \rightarrow \mathcal{E}/D$$
Polynomials & polynomial functors

We will consider polynomials in $\text{Psh}(\mathcal{C})$, inducing polynomial functors $\text{Psh}(\mathcal{C})/S \to \text{Psh}(\mathcal{C})/S$, where $(\mathcal{C}, S)$ is a cartesian context structure with algebraic structure $\Sigma_{ty} S \to S$.

Let $P$ be a polynomial in $\text{Psh}(\mathcal{C})$. Algebras for the corresponding polynomial functor are bundles $\tau : T \to S$ together with morphisms as in the following.

\[
\begin{array}{ccc}
  \text{F}_P(\tau) & \xrightarrow{[P]} & (\tau) \\
  S & & S
\end{array}
\]
Algebraic structure & natural deduction

\[ \Gamma \vdash a : A \quad \Gamma \vdash b : B \]

\[ \Gamma \vdash \text{pair}(a, b) : \text{Prod}(A, B) \]
Algebraic structure & natural deduction

two type metavariables

\( \Gamma \vdash a : A \quad \Gamma \vdash b : B \)

\[ \frac{}{\Gamma \vdash \text{pair}(a, b) : \text{Prod}(A, B)} \]

\( \times \)

\( S \times S \)

in \( \text{Psh}(C) \)
Algebraic structure & natural deduction

\[
\Gamma \vdash a : A \quad \Gamma \vdash b : B
\]

\[
\Gamma \vdash \text{pair}(a, b) : \text{Prod}(A, B)
\]

in $\text{Psh}(\mathcal{C})$
Algebraic structure & natural deduction

two hypotheses

\[ \Gamma \vdash a : A \quad \Gamma \vdash b : B \]

\[ \Gamma \vdash \text{pair}(a, b) : \text{Prod}(A, B) \]

\[ \text{Prod-INTRO} \]

\[ [\text{Prod}] \]

\[ S \times S + S \times S \rightarrow S \times S \rightarrow S \]

\[ \Gamma \vdash a : A \quad \Gamma \vdash b : B \]

in \( \text{Psh}(\mathcal{C}) \)
Algebraic structure & natural deduction

\[ \Gamma \vdash a : A \quad \Gamma \vdash b : B \]

\[ \frac{}{\Gamma \vdash \text{pair}(a, b) : \text{Prod}(A, B)} \quad \text{Prod-INTRO} \]

in \(\text{Psh}(\mathcal{C})\)
Algebraic structure & natural deduction

Γ ⊢ a : A
Γ ⊢ b : B

Γ ⊢ pair(a, b) : Prod(A, B)

Γ ⊢ a : A
Γ ⊢ b : B

__________________________
Prod-INTRO

Γ ⊢ pair(a, b) : Prod(A, B)

in Psh(ℂ)
Algebraic structure & natural deduction

\[
\Gamma \vdash a : A \quad \Gamma \vdash b : B \\
\frac{}{\Gamma \vdash \text{pair}(a, b) : \text{Prod}(A, B)} \quad \text{Prod-INTRO}
\]

\[
\begin{array}{c}
\text{in } \text{Psh}(\mathcal{C})
\end{array}
\]
Algebraic structure & natural deduction

\[ \Gamma \vdash a : A \quad \Gamma \vdash b : B \]

\[ \Gamma \vdash \text{pair}(a, b) : \text{Prod}(A, B) \]

\[ S \leftrightarrow S \times S + S \times S \nrightarrow S \times S \nrightarrow S \]

\[ [\pi_1, \pi_2] \]

functor algebra

\[ \text{in Psh}(\mathcal{C}) \]
Binding algebraic structure & natural deduction

\[ \Gamma, x : A \vdash t : B \]

\[ \Gamma \vdash \text{abs}(x : A . t) : \text{Fun}(A, B) \]
Binding algebraic structure & natural deduction

two type metavariables

\[ \Gamma, x : A \vdash t : B \]

\[ \Gamma \vdash \text{abs}(x : A . t) : \text{Fun}(A, B) \quad \text{Fun-INTRO} \]

\[ S \times S \]

\[ A \quad B \]

in \( \text{Psh}(\mathcal{C}) \)
Binding algebraic structure & natural deduction

\[ \Gamma, x : A \vdash t : B \]

\[ \Gamma \vdash \text{abs}(x : A \cdot t) : \text{Fun}(A, B) \]

\[ \text{Fun-INTRO} \]

in \( \text{Psh}(\mathcal{C}) \)
Binding algebraic structure & natural deduction

\[ \Gamma, x : A \vdash t : B \]
\[ \Gamma \vdash \text{abs}(x : A . t) : \text{Fun}(A, B) \]

Fun-\text{INTRO}

\[ V \times S \rightarrow S \times S \rightarrow S \]

in \text{Psh}(\mathcal{C})
Binding algebraic structure & natural deduction

\[ \Gamma, x : A \vdash t : B \]

\[ \Gamma \vdash \text{abs}(x : A . t) : \text{Fun}(A, B) \]

Fun-INTRO

\[ \text{types of variables} \]

\[ V \times \sigma \to \sigma \times \sigma \to \sigma \]

in \( \text{Psh}(\mathcal{C}) \)
Binding algebraic structure & natural deduction

\[ \Gamma, x : A \vdash t : B \]

\[ \Gamma \vdash \text{abs}(x : A . t) : \text{Fun}(A, B) \]

\[ \pi_2 \quad \nu \times \text{id} \quad [[\text{Fun}]] \]

\[ S \leftarrow V \times S \rightarrow S \times S \rightarrow S \]

in \( \text{Psh}(\mathcal{C}) \)
Binding algebraic structure & natural deduction

\[ \Gamma, x : A \vdash t : B \]

Fun-INTRO

\[ \Gamma \vdash \text{abs}(x : A \cdot t) : \text{Fun}(A, B) \]

\[
\begin{array}{c}
\pi_2 & \nu \times \text{id} & \llbracket \text{Fun} \rrbracket \\
S & V \times S & S \times S \\
\longrightarrow & \rightarrow & \rightarrow \\
S & S \times S & S
\end{array}
\]

functor algebra

in \( \text{Psh}(\mathcal{C}) \)

\[
\begin{array}{c}
\prod_{A, B \in S} T_B(\Gamma \cdot A) \\
\pi \\
S \times S
\end{array}
\overset{\llbracket \text{abs} \rrbracket_{\Gamma}}{\longrightarrow}
\begin{array}{c}
T(\Gamma) \\
\tau_{\Gamma} \quad \llbracket \text{Fun} \rrbracket_{\Gamma} \\
S
\end{array}
\]

introduction rule

polynomial

polynomial functor algebra
Algebraic structure & natural deduction

The natural deduction rules corresponding to introduction/elimination can be described by a second-order arity (describing the typing and binding data for each argument).

Each second-order arity induces a polynomial in $\text{Psh}(\mathcal{C})$.

The algebras for their associated polynomial functors are presheaves with the corresponding (typed & binding) term structure.

We can collect the arities into a term signature $\Sigma_{\text{tm}}$, which itself induces a polynomial.

(NB. We're using the same notation for a signature and the polynomial functor it induces.)
Models of simple type theory

\[ x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau \]

- **contexts**
- **terms and types**
- **algebraic structure**
Model homomorphisms

\[ (S, \mathcal{C}, \varepsilon, \langle - \rangle, - \times \langle = \rangle, T, \tau, \llbracket - \rrbracket_{ty}, \llbracket - \rrbracket_{tm}) \rightarrow (S', \mathcal{C}', \varepsilon', \langle - \rangle', - \times \langle = \rangle', T', \tau', \llbracket - \rrbracket'_{ty}, \llbracket - \rrbracket'_{tm}) \]

\[ h : S \rightarrow S' \]

\[ H : \mathcal{C} \rightarrow \mathcal{C}' \]

\[ f : T \rightarrow H^*(T') \]

- \( h \) is a \( \Sigma_{ty} \)-algebra homomorphism
- \( H \) is a structure-preserving functor
- \( f \) is a morphism in \( \text{Psh}(\mathcal{C})/\mathcal{S} \) preserving the \( \Sigma_{tm} \)-algebra structure
Model homomorphisms

\[ h : S \rightarrow S' \]
\[ f : T \rightarrow H^*(T') \]

- \( h \) is a \( \Sigma_{ty} \)-algebra homomorphism
- \( f \) is a morphism in \( \text{Psh}(\mathcal{C})/\Sigma \) preserving the \( \Sigma_{tm} \)-algebra structure

Where \( \mathfrak{h} = h^*(- H) \)
For any given term and type signature, we want a model of simple type theory freely generated by the syntax.

The model freely generated by the syntax is exactly the initial model.

Since we have no type dependency, we can construct the initial model piecewise.
Initial models of simple type theory
Initial models of simple type theory

- initial $\Sigma_{ty}$-algebra
  as in universal algebra
Initial models of simple type theory

- $S$: initial $\Sigma_{ty}$-algebra as in universal algebra

- $\mathcal{C}, \varepsilon$: free cartesian category on $S$

concretely, the opposite of the comma category $(\mathcal{F} \to \text{Set}) \downarrow (\mathbb{1} \to S \to \text{Set})$
Initial models of simple type theory

- $S$ initial $\Sigma_{ty}$-algebra as in universal algebra
- $(C, \varepsilon)$ free cartesian category on $S$
  concretely, the opposite of the comma category $(\mathbb{F} \to \text{Set}) \downarrow (1 \to \text{Set})$
- $(T, \tau)$ initial $\Sigma_{tm}$-algebra
  using Adámek's initial algebra construction
There's one last thing...
Substitution

\[ \Gamma \vdash t [u/x] : \tau \]

???
Substitution

\[ \Gamma \vdash t \left[ u/x \right] : \tau \]

an algebraic operation on terms
Substitution

\[ \begin{align*}
\Gamma, \ x : A & \vdash \text{var}(x) : A \\
\Gamma, \ x : A \vdash \text{var}(x) : A
\end{align*} \] 

\[ \begin{align*}
\Gamma, \ x : A \vdash t : B & \quad \Gamma \vdash u : A \\
\Gamma \vdash u : A \\
\Gamma \vdash \text{subst}(x : A . t, u) : B
\end{align*} \]
Substitution

\[ \Gamma, x : A \vdash \text{var}(x) : A \]

\[ S \leftarrow 0 \rightarrow V \rightarrow S \]

\[ \begin{array}{c}
\Gamma, x : A \vdash t : B \\
\Gamma \vdash u : A
\end{array} \]

\[ \Gamma \vdash \text{subst}(x : A . t, u) : B \]

\[ [\pi_2, \pi_1] \\
S \leftarrow V \times S + S \times S \rightarrow S \times S \rightarrow \hat{S} \]
Substitution

\[ \Gamma, x : A \vdash \text{var}(x) : A \]

\[ \Gamma, x : A \vdash t : B \quad \Gamma \vdash u : A \]

\[ \Gamma \vdash \text{subst}(x : A . t, u) : B \]

subject to equational laws...
Initial models of simple type theories

\[ \Sigma_{ty} + \Sigma_{tm} \rightarrow \text{syntactic model} \]
Initial models of simple type theories with substitution

\[ \Sigma_{ty} + \Sigma_{tm} \rightarrow \Sigma_{subst} \]

syntactic model with substitution
A partial answer

Q. What is a simple type theory?
A partial answer

Q. What is a simple type theory?
A. An initial model

\[ (C, T \rightarrow S, \llbracket \cdot \rrbracket_{ty}, \llbracket \cdot \rrbracket_{tm + \text{subst}}) \]
A partial answer

Q. What is a simple type theory?

A. An initial model

\[(C, T \to S, \llbracket \cdot \rrbracket_{ty}, \llbracket \cdot \rrbracket_{tm + subst})\]

We can now construct the classifying category and equational logic...
Conclusion

- Models of simple type theory consist of structures for contexts, typed terms and algebraic structure.

- Natural deduction rules that present simple type theories can themselves be presented by polynomials.

- The initial model of simple type theory is the syntactic model of the type theory, and can be constructed explicitly with a free algebra construction.

- We can construct the syntactic model with substitution, from which we can derive a classifying category, demonstrating that the type theory is its internal language.