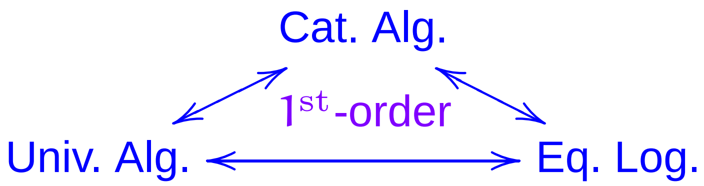
Algebraic Foundations for Type Theories

Marcelo Fiore

Computer Laboratory
University of Cambridge

Types 2011
8.IX.2011

The known



Cat. Alg.

1^{st}-order

Univ. Alg.

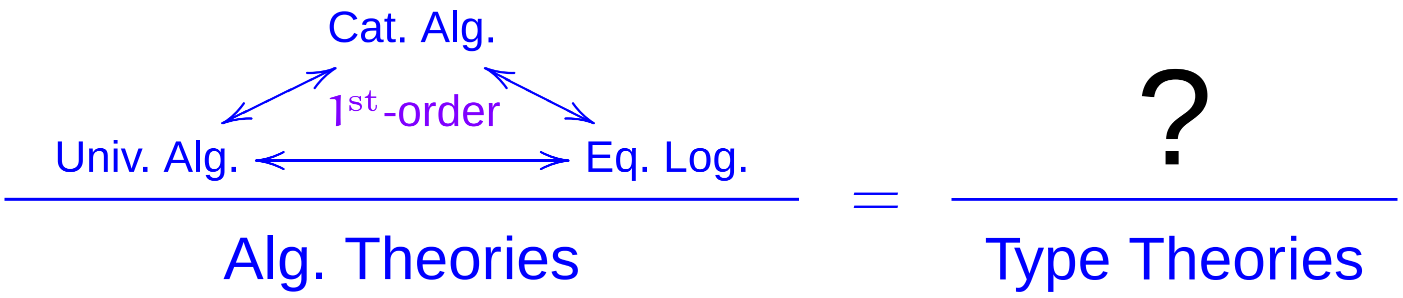
Eq. Log.

Alg. Theories

(mon, grp,
ring, mod, . . .)

▶ [1, 5, 6]

The Unknown



Cat. Alg.

1st-order

Univ. Alg. ⟷ Eq. Log.

Alg. Theories

=

?

Type Theories

(mon, grp, ring, mod, . . .)

(simply typed, dep. typed, polymorphic, linear, . . .)

▶ [1, 5, 6], [29].

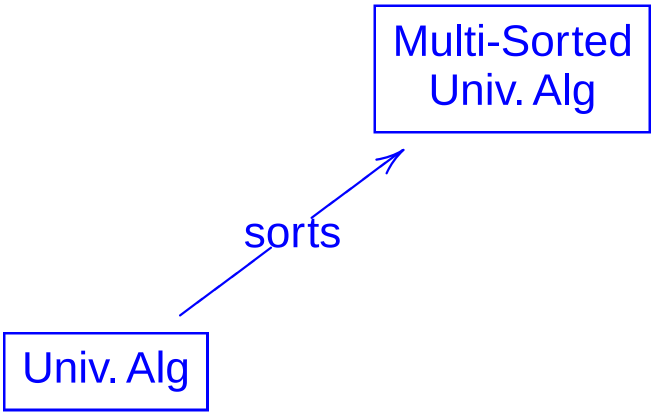
Programme

mathematical models $\rightsquigarrow$ meta-theories

- Development of *algebraic meta-theories* for formal languages.
    - *Semantics*
        - Model theory.
    - *Syntax*
        - Initial-algebra semantics.
        - Structural induction and recursion.
        - Substitution.
- Synthesis of *deduction systems* for equational reasoning and computation by rewriting.

- [29], [26].

The Space

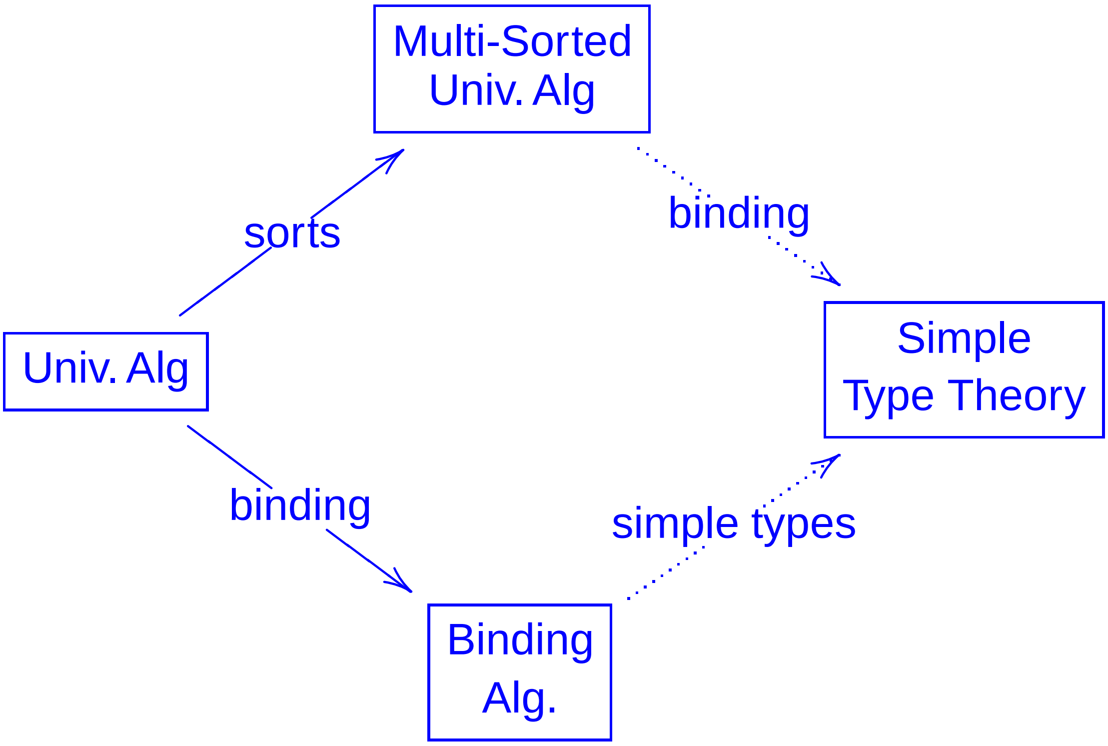

Multi-Sorted Univ. Alg

sorts

Univ. Alg

▶ [1]

The Space



Multi-Sorted Univ. Alg

sorts

binding

Univ. Alg

Simple Type Theory

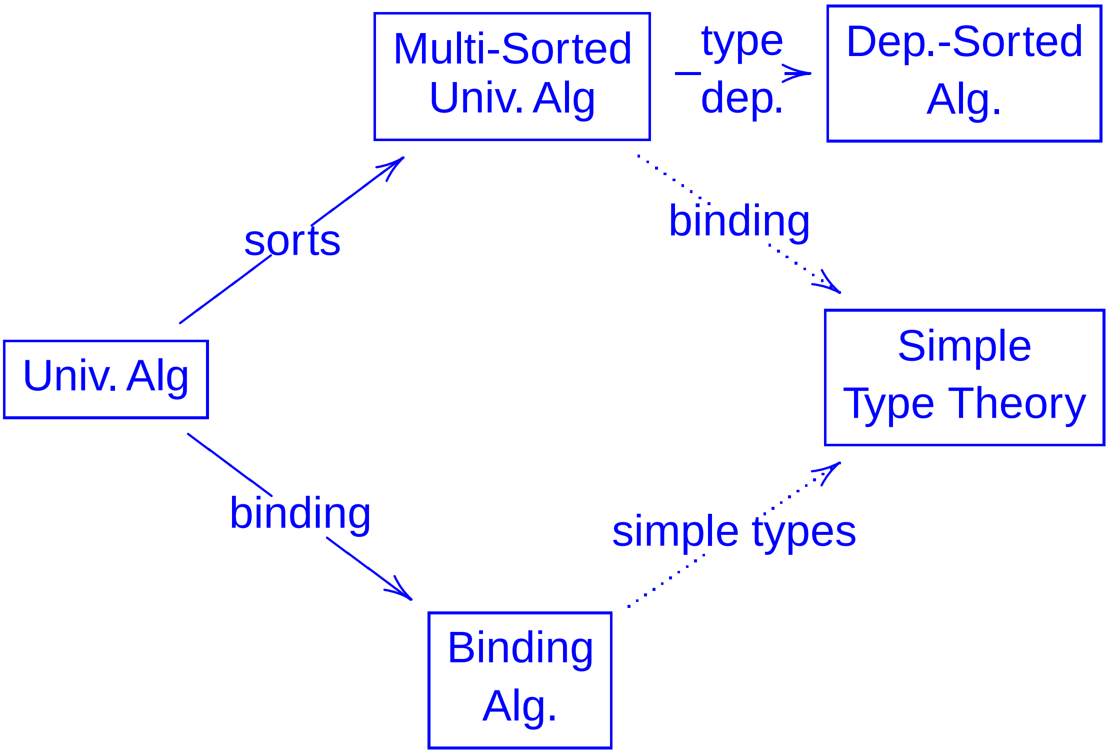
binding

simple types

Binding Alg.

- [1], [2]

The Space



Multi-Sorted Univ. Alg

type dep.

Dep.-Sorted Alg.

sorts

binding

Univ. Alg

Simple Type Theory

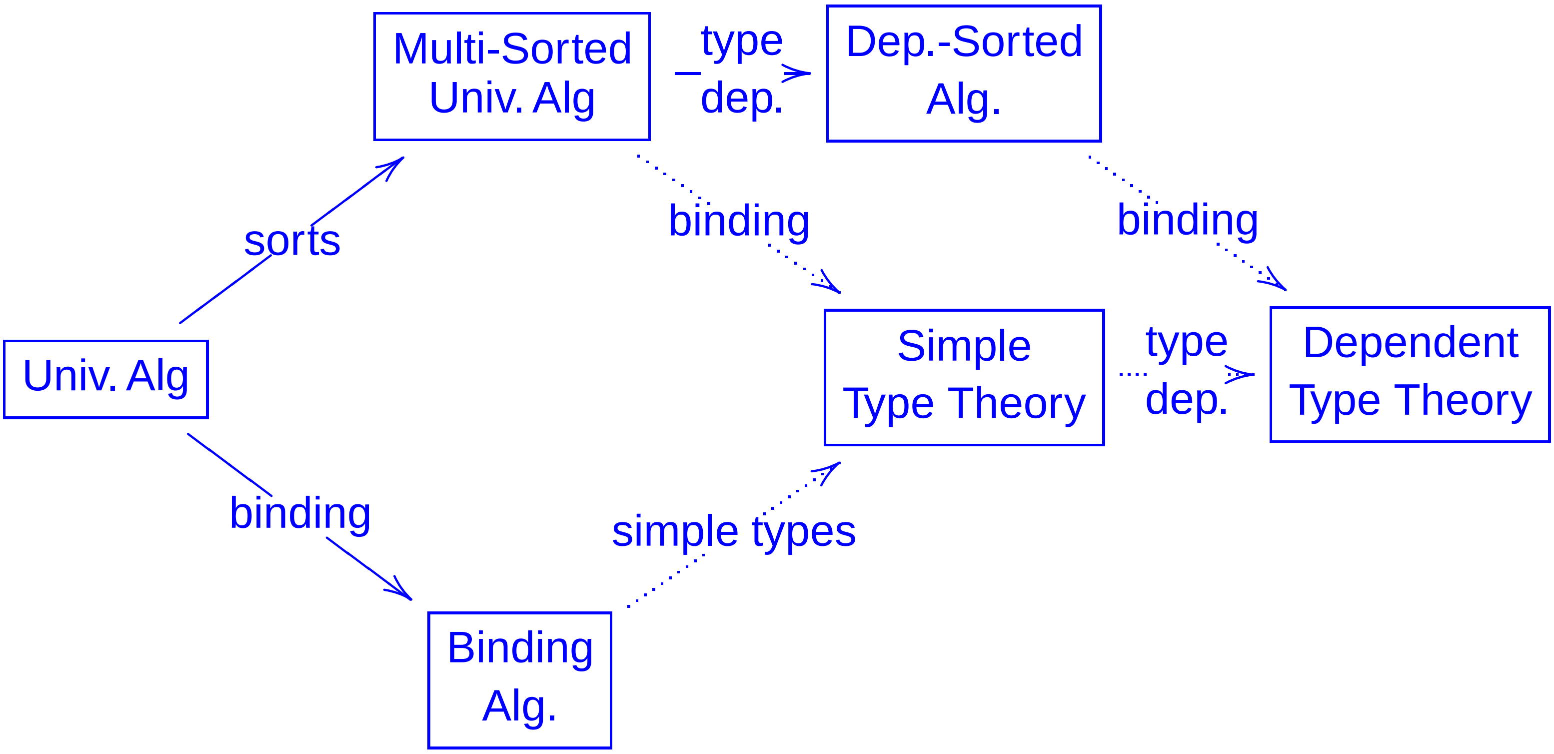
binding

simple types

Binding Alg.

- ▶ [1], [2], [14]

The Space



Multi-Sorted Univ. Alg

type dep.

Dep.-Sorted Alg.

sorts

binding

binding

Univ. Alg

Simple Type Theory

type dep.

Dependent Type Theory

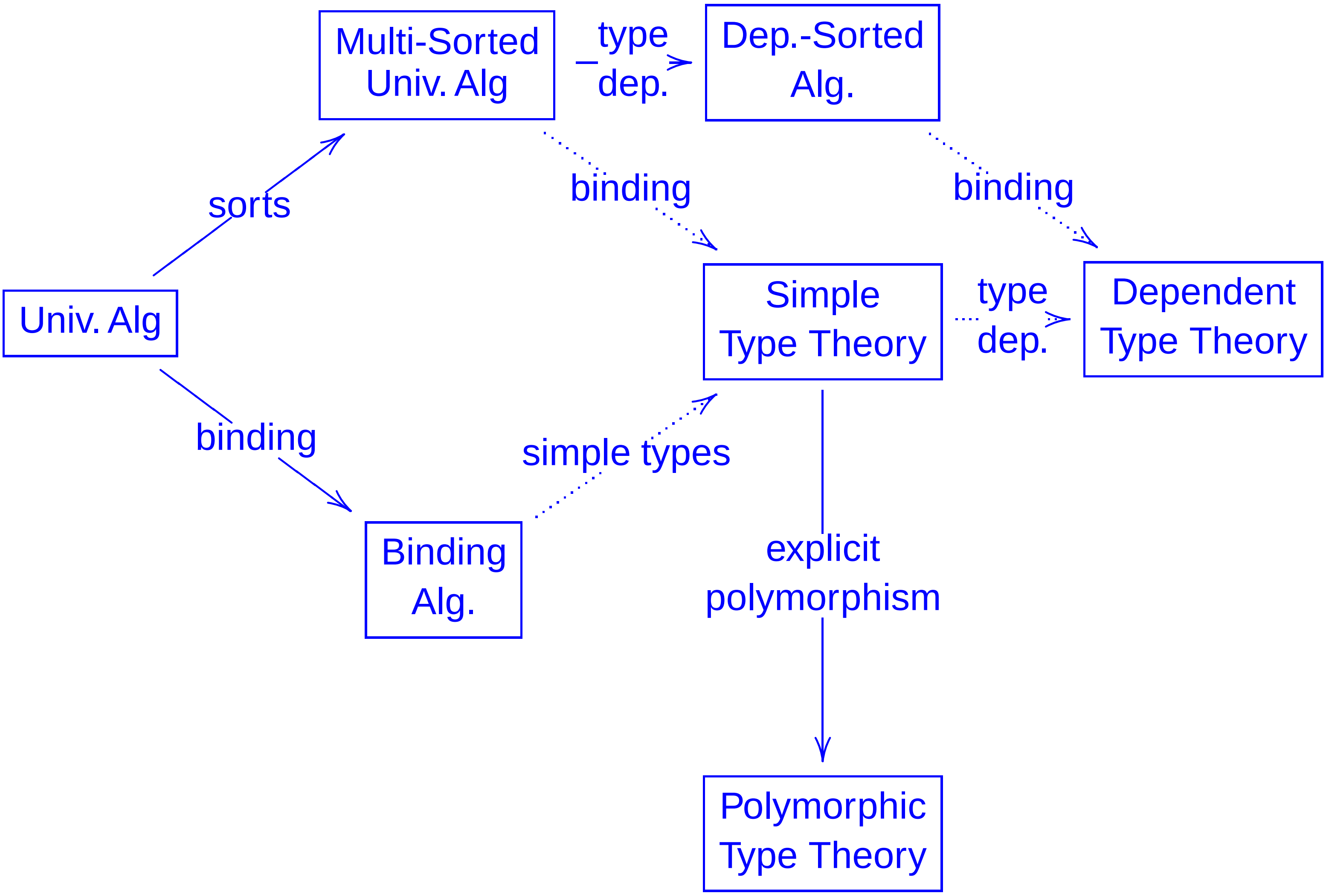
binding

simple types

Binding Alg.

▶ [1], [2], [14], [10]

The Space



Multi-Sorted Univ. Alg

type dep.

Dep.-Sorted Alg.

sorts

binding

binding

Univ. Alg

Simple Type Theory

type dep.

Dependent Type Theory

binding

simple types

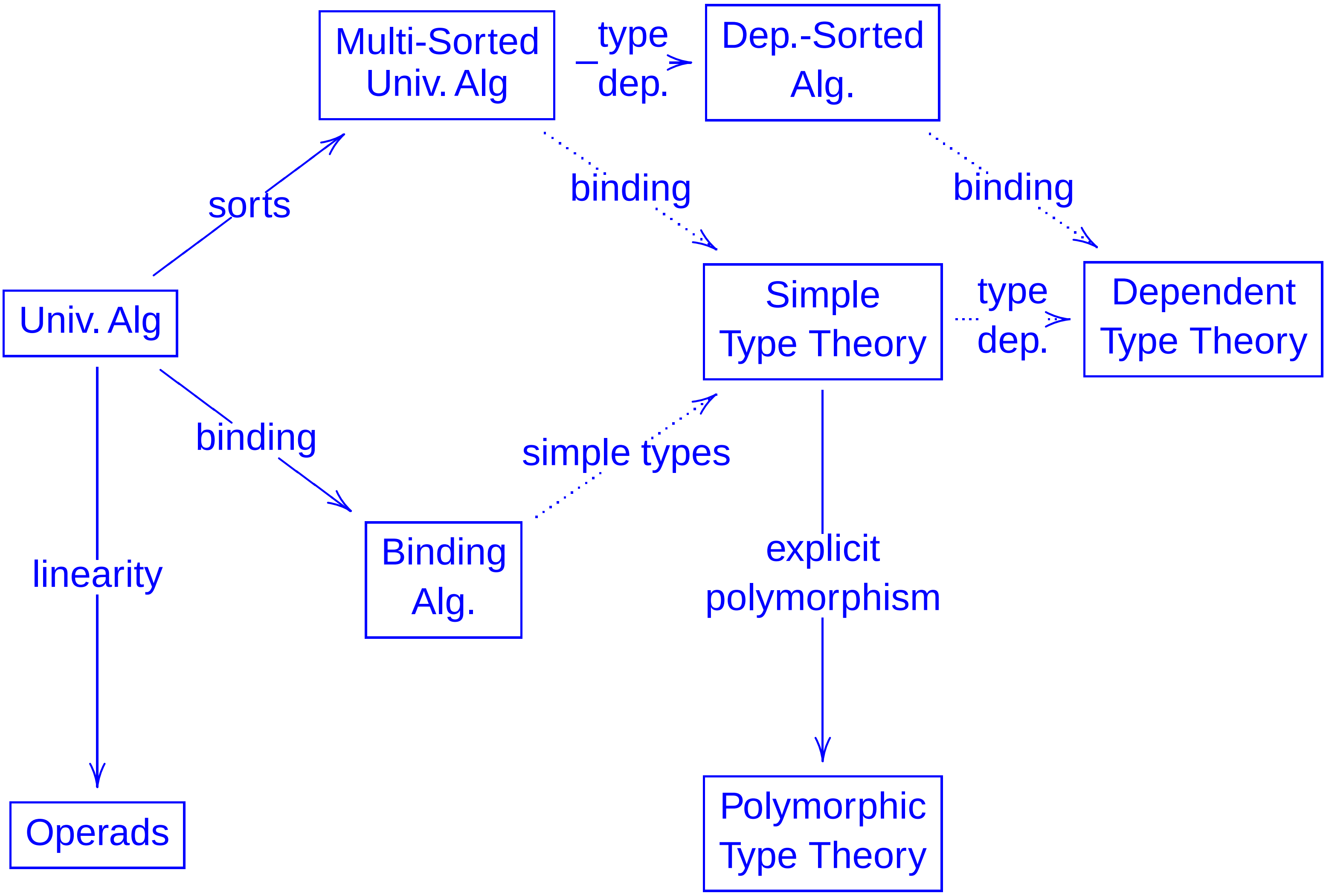
Binding Alg.

explicit polymorphism

Polymorphic Type Theory

- ▶ [1], [2], [14], [10], [8, 12]

The Space



Multi-Sorted Univ. Alg

type dep.

Dep.-Sorted Alg.

sorts

binding

binding

Univ. Alg

Simple Type Theory

type dep.

Dependent Type Theory

binding

simple types

linearity

Binding Alg.

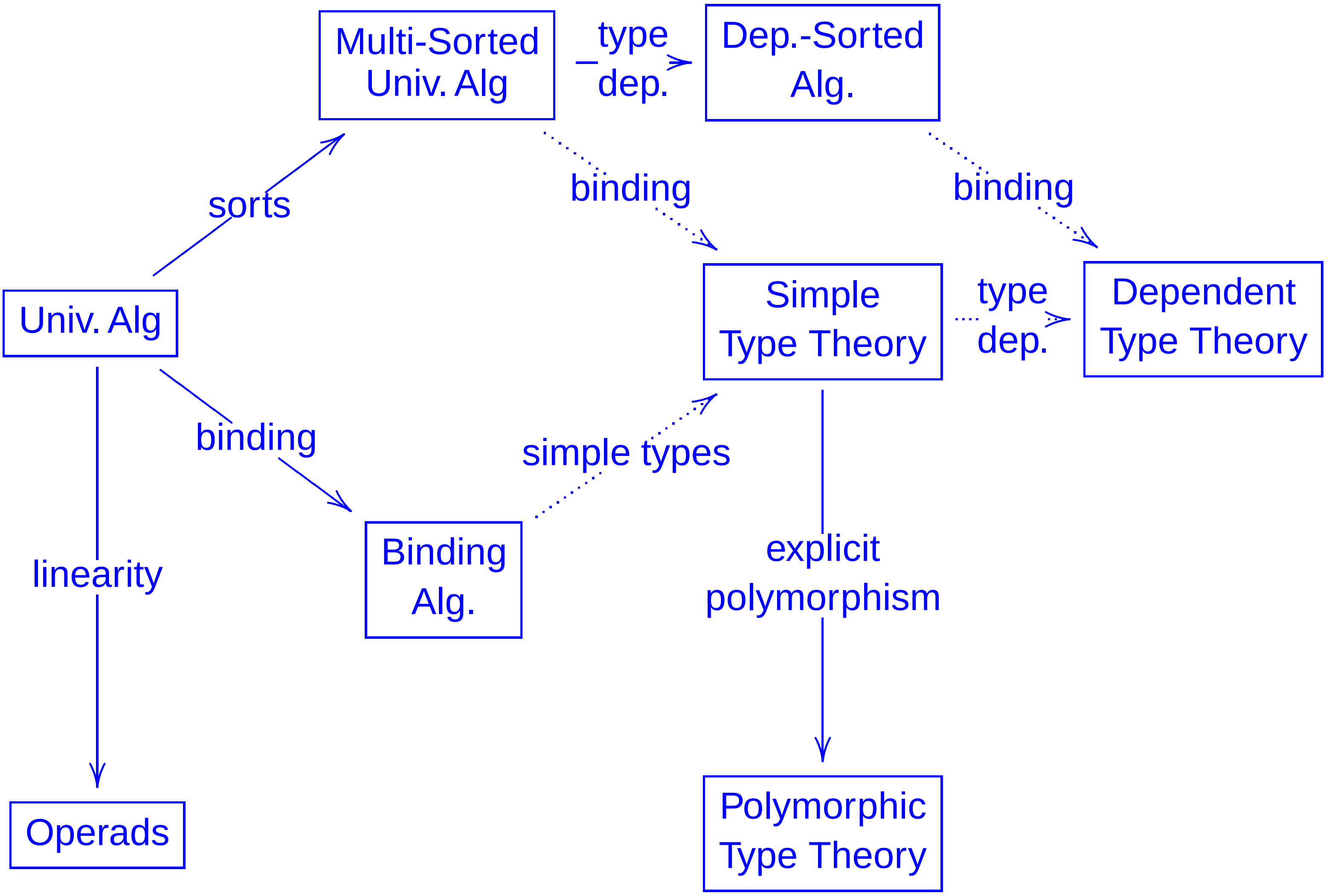
explicit polymorphism

Operads

Polymorphic Type Theory

- [1], [2], [14], [10], [8, 12], [9, 11].

The Space



The Talk

I Modelling of simple type theories.

II Modelling of dependent type theories.

III Foundations.

▶ [1], [2], [14], [10], [8, 12], [9, 11].

I

Algebraic Modelling of Simple Type Theories

Simple Type Theory

	algebraic theories	simply-typed theories
types	unstructured	algebraic
terms	algebraic	algebraic with binding

The syntactic theory should account for:

- variables and meta-variables
- variable binding and α-equivalence
- capture-avoiding and meta substitution
- mono and multi sorting

- [19, 20, 22, 25].

Categories of Contexts

Def: An S-sorted *context structure* is given by

- a small category $\mathbb{C}$ with terminal object,
- objects $\langle\sigma\rangle \in \mathbb{C}$ for all $\sigma \in S$,
- product diagrams

$$\begin{tikzcd} & \Gamma, \sigma \arrow[dl] \arrow[dr] & \\ \Gamma & & \langle\sigma\rangle \end{tikzcd}$$

for all $\Gamma \in \mathbb{C}$ and $\sigma \in S$.

Example: $\mathsf{FinSet}^{\mathrm{op}}$ is the initial *mono-sorted* context structure.

Algebraic Models of Variable Binding in $\widehat{\mathbb{C}}^{S}$

Example: Untyped λ-calculus.

- Syntax:

$$t \ ::= \ x \mid t'(t'') \mid \lambda x.\, t'$$

► [3]

Algebraic Models of Variable Binding in $\widehat{\mathbb{C}}^S$

Example: Untyped λ-calculus.

- Syntax:

$$t \quad ::= \quad x \mid t'(t'') \mid \lambda x.\, t'$$

- Algebras:

$$\begin{aligned} \mathsf{var} :&\quad 1 \longrightarrow A^{\mathbf{y}\langle s\rangle} \\ \mathsf{app} :&\quad A^2 \longrightarrow A \\ \mathsf{abs} :&\quad A^{\mathbf{y}\langle s\rangle} \longrightarrow A \end{aligned} \qquad \text{in } \widehat{\mathbb{C}}$$

$\widehat{\mathbb{C}} =^{\mathrm{def}} \mathcal{S}et^{\mathbb{C}^{\mathrm{op}}}$ and $\mathbf{y} : \mathbb{C} \hookrightarrow \widehat{\mathbb{C}}$ is the Yoneda embedding

---

- [3], [20, 22].

Algebraic Models of Variable Binding in $\widehat{\mathbb{C}}^{S}$

Example: Untyped λ-calculus.

- Syntax:

$$t \;::=\; x \mid t'(t'') \mid \lambda x.\, t'$$

- Algebras:

$$\begin{aligned} \mathsf{var} &: 1 \to A^{\mathbf{y}\langle s\rangle} \\ \mathsf{app} &: A^2 \to A \qquad\qquad \text{in } \widehat{\mathbb{C}} \\ \mathsf{abs} &: A^{\mathbf{y}\langle s\rangle} \to A \end{aligned}$$

$\widehat{\mathbb{C}} =^{\mathrm{def}} \mathcal{S}et^{\mathbb{C}^{\mathrm{op}}}$ and $\mathbf{y} : \mathbb{C} \hookrightarrow \widehat{\mathbb{C}}$ is the Yoneda embedding

NB:

$$P^{\mathbf{y}\langle s\rangle}(\Gamma) \;=\; P(\Gamma, s)$$

as

$$\begin{tikzcd}
\mathbb{C} \arrow[r, hook] \arrow[d, "{-,s}"'] & \widehat{\mathbb{C}} \arrow[d, shift right=6, "(-)\times\mathbf{y}\langle s\rangle"'] \arrow[d, phantom, shift right=3, "\dashv"] \arrow[d, phantom, shift left=3, "\dashv"] \arrow[d, shift left=6] \\
\mathbb{C} \arrow[r, hook] & \widehat{\mathbb{C}} \arrow[u, "(-)^{\mathbf{y}\langle s\rangle}"']
\end{tikzcd}$$

► [3], [20, 22].

Algebraic Models of Variable Binding in $\widehat{\mathbb{C}}^{\mathrm{S}}$

Example: Untyped λ-calculus.

- Syntax:

$$t \quad ::= \quad x \mid t'(t'') \mid \lambda x.\, t'$$

- Algebras:

$$\begin{aligned} \text{var} :&\quad \mathbf{y}\langle s \rangle \to A \\ \text{app} :&\quad A^2 \to A \\ \text{abs} :&\quad A^{\mathbf{y}\langle s \rangle} \to A \end{aligned} \qquad \text{in } \widehat{\mathbb{C}}$$

- Initial model:

$$\Lambda \in \mathcal{S}et^{\mathrm{FinSet}}$$

with $\Lambda(n)$ the set of α-equivalence classes of λ-terms with free variables amongst $x_1, \ldots, x_n$.

- [20, 22].

Single-Variable Substitution

- Substitution algebras:

$$\mathsf{subst} : A^{\mathbf{y}\langle s\rangle} \times A \to A$$

satisfying

. . . natural axioms . . .

- [20, 22].

Single-Variable Substitution

- Substitution algebras:

$$\mathsf{subst} : A^{\mathbf{y}\langle s\rangle} \times A \to A$$

satisfying

. . . natural axioms . . .

- Initial model:

$$\Lambda \in \boldsymbol{\mathcal{S}et}^{\mathsf{FinSet}}$$

with capture-avoiding single-variable substitution.

► [20, 22].

Substitution Algebras

1. $u : A \vdash \mathsf{var}(x)\left[{}^{u}/_{x}\right] = u$

2. $t : A, u : A \vdash t\left[{}^{u}/_{x}\right] = t$

3. $t : A^{\mathbf{y}\langle s\rangle \times \mathbf{y}\langle s\rangle}, x : \mathbf{y}\langle s\rangle$
   $\vdash t(x, y)\left[{}^{\mathsf{var}(x)}/_{y}\right] = t(x, x)$

4. $t : A^{\mathbf{y}\langle s\rangle \times \mathbf{y}\langle s\rangle}, u : A^{\mathbf{y}\langle s\rangle}, v : A$
   $\vdash \left(t(y, x)\left[{}^{u(x)}/_{y}\right]\right)\left[{}^{v}/_{x}\right]$
   $= \left(t(y, x)\left[{}^{v}/_{x}\right]\right)\left[{}^{u(x)\left[{}^{v}/_{x}\right]}/_{y}\right]$

- [20, 22].

Substitution Algebras

1. $u : A \vdash \mathsf{var}(x)[{}^{u}/{}_{x}] = u$

2. $t : A, u : A \vdash t[{}^{u}/{}_{x}] = t$

3. $t : A^{\mathbf{y}\langle s\rangle \times \mathbf{y}\langle s\rangle}, x : \mathbf{y}\langle s\rangle$

   $\vdash t(x, y)[{}^{\mathsf{var}(x)}/{}_{y}] = t(x, x)$

4. $t : A^{\mathbf{y}\langle s\rangle \times \mathbf{y}\langle s\rangle}, u : A^{\mathbf{y}\langle s\rangle}, v : A$

   $\vdash (t(y, x)[{}^{u(x)}/{}_{y}])[{}^{v}/{}_{x}]$

   $= (t(y, x)[{}^{v}/{}_{x}])[{}^{u(x)[{}^{v}/{}_{x}]}/{}_{y}]$

5. $t, t' : A^{\mathbf{y}\langle s\rangle}, u : A$

   $\vdash \mathsf{app}(t(x), t'(x))[{}^{u}/{}_{x}]$

   $= \mathsf{app}(t(x)[{}^{u}/{}_{x}], t'(x)[{}^{u}/{}_{x}])$

6. $t : A^{\mathbf{y}\langle s\rangle \times \mathbf{y}\langle s\rangle}, u : A$

   $\vdash \mathsf{abs}(\lambda y. t(y, x))[{}^{u}/{}_{x}]$

   $= \mathsf{abs}(\lambda y. t(y, x)[{}^{u}/{}_{x}])$

▶ [20, 22].

Free Constructions

$$\begin{tikzcd}
\mathsf{SubstAlg} \arrow[d, shift left=2] \arrow[d, phantom, "\dashv"] \\
\mathcal{S}et^{\mathsf{FinSet}} \arrow[u, shift left=2]
\end{tikzcd}$$

- Explicit description of syntax with variable binding.
- Induction principle for syntax with variable biding.
- Definition of capture-avoiding substitution by structural recursion.

- [23, 25].

Free Constructions

$$\begin{tikzcd}\mathsf{SubstAlg} \arrow[d, shift left] \\ \mathcal{S}et^{\mathsf{FinSet}} \arrow[u, shift left] \arrow[u, phantom, "\dashv"]\end{tikzcd}$$

- Explicit description of syntax with variable binding.
- Induction principle for syntax with variable biding.
- Definition of capture-avoiding substitution by structural recursion.
- Mathematical foundations for *metavariables*.

- [23, 25].

Metavariables

$$\begin{tikzcd} \mathsf{SubstAlg} \arrow[d, shift left=2] \arrow[d, phantom, "\dashv"] \\ \mathcal{S}et^{\mathsf{FinSet}} \arrow[u, shift left=2] \arrow[loop below, "\mathcal{M}"] \end{tikzcd}$$

- Kleisli maps

  $$\mathbf{y}(n) \to \mathcal{M}\big(\textstyle\coprod_i \mathbf{y}(m_i)\big)$$

  are in bijective correspondence with terms

  $$\begin{aligned} t \quad ::= \quad & x \mid t'(t'') \mid \lambda x.\, t' \\ \mid \quad & M_i[t_1, \ldots, t_{m_i}] \end{aligned}$$

  with free variables amongst $x_1, \ldots, x_n$.

- Definition of *meta-substitution* by structural recursion:

  $$\mathcal{M}(X) \times Y^X \to \mathcal{M}(Y)$$

- [13, 23, 25].

Second-Order
Equational Presentations

Example: Untyped λ-calculus.

$$(\beta)\ M : 1 \rhd x \vdash \big(\lambda x.\, M[x]\big)\, x = M\big[x\big]$$

$$(\eta)\ M : 0 \rhd \cdot \vdash \lambda x.\, M[\,]\,(x) = M[\,]$$

▶ [13]

Second-Order
Equational Presentations

Example: Untyped λ-calculus.

$$(\beta)\ M : 1 \rhd x \vdash \big(\lambda x.\, M[x]\big)\, x = M\big[x\big]$$

$$(\eta)\ M : 0 \rhd \cdot \vdash \lambda x.\, M[\,]\,(x) = M[\,]$$

Second-Order
Equational Logic

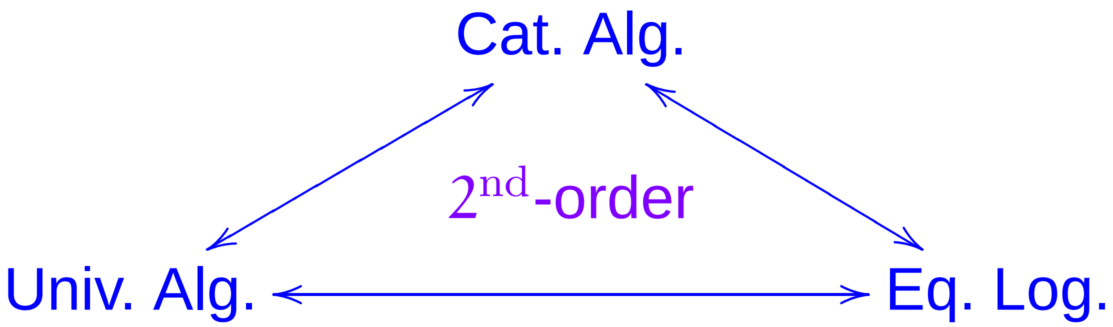
(Extended Metasubstitution Rule)

$$\frac{\begin{array}{c} M_1 : m_1, \ldots, M_k : m_k \rhd \Gamma \vdash s \equiv t \\ \Theta \rhd \Delta, x_{i,1}, \ldots, x_{i,m_i} \vdash s_i \equiv t_i \quad (1 \leq i \leq k) \end{array}}{\begin{array}{l} \Theta \rhd \Gamma, \Delta \\ \quad \vdash s\{M_i := (\vec{x_i})s_i\}_{1 \leq i \leq k} \equiv t\{M_i := (\vec{x_i})t_i\}_{1 \leq i \leq k} \end{array}}$$

▶ [13], [31].

Results

Extension of the mathematical theory of (first-order) algebraic structure to simple type theory:



Cat. Alg.

2$^{\text{nd}}$-order

Univ. Alg.

Eq. Log.

- Conservativity of Second-Order Equational Logic over Birkhoff's (first-order) Equational Logic.
- Soundness and completeness of Second-Order Equational Logic.
- Soundness and completeness of (bidirectional) Second-Order Term Rewriting.
- Presentation/theory correspondence via classifying categories and internal languages.
- Universal-algebra/categorical-algebra correspondence.
- Theory of syntactic algebraic translations.

- [31, 32].

II

Algebraic Modelling of Dependent Type Theories

Dependent Type Theory

	simply-typed theories	dependently-typed theories
types	algebraic	algebraic with binding
terms	algebraic with binding	algebraic with binding

The syntactic theory should account for:

- type dependency
- variable binding and α-equivalence
- term and type substitution

▶ [10].

- Dependency.

$$\begin{aligned} &\vdash \quad C_0 \\ x, y : C_0 &\vdash \quad C_1(x, y) \end{aligned}$$

- Binding.

$$\frac{\Gamma, x : \sigma \vdash \tau}{\Gamma \vdash \Pi x : \sigma.\, \tau}$$

$$\frac{\Gamma, x : \sigma \vdash t : \tau}{\Gamma \vdash \lambda x : \sigma.t \, : \, \Pi x : \sigma.\, \tau}$$

- Substitution.

$$\frac{\Gamma \vdash t : \Pi x : \sigma.\tau \qquad \Gamma \vdash u : \sigma}{\Gamma \vdash t(u) : \tau[{}^{u}/_{x}]}$$

$$(\lambda x : \sigma.\, t)(u) \; = \; t[{}^{u}/_{x}]$$

Categories of Dependent Contexts

Def: A *dependently-typed context structure* is given by

- a small category $\mathbb{C}$ with terminal object,
- a presheaf $S \in \widehat{\mathbb{C}}$,
- a functorial assignment of pullbacks

$$\begin{tikzcd}
 & \Delta, \sigma[f] \arrow[dl, "\pi_{\Delta \vdash \sigma[f]}"'] \arrow[dr, "f^*"] & \\
\Delta \arrow[dr, "f"'] & & \Gamma, \sigma \arrow[dl, "\pi_{\Gamma \vdash \sigma}"] \\
 & \Gamma &
\end{tikzcd}$$

for all $\Gamma \vdash \sigma$ and $f : \Delta \to \Gamma$ in $\mathbb{C}$.

► [14].

Type-Dependent Binding

- **Local context-extension lemma.**

  For $\Gamma \vdash \sigma$ and $P \in \widehat{\mathbb{C}_{/\Gamma}}$,

$$P^{\mathbf{y}(\pi_{\Gamma\vdash\sigma})}(\Delta \xrightarrow{f} \Gamma) \;\cong\; P(\Delta, \sigma[f] \to \Gamma) \quad .$$

Type-Dependent Binding

- **Local context-extension lemma.**

  For $\Gamma \vdash \sigma$ and $P \in \widehat{\mathbb{C}_{/\Gamma}}$,

  $$P^{\mathbf{y}(\pi_{\Gamma\vdash\sigma})}(\Delta \xrightarrow{f} \Gamma) \;\cong\; P(\Delta, \sigma[f] \to \Gamma)\ \ .$$

- Type-dependent binding operators.

  $$\Pi_\sigma : {S_\Gamma}^{\mathbf{y}(\pi_{\Gamma\vdash\sigma})} \to S_\Gamma \text{ in } \widehat{\mathbb{C}_{/\Gamma}}$$

Decomposition of Binding Arities

- For $\Gamma \vdash \sigma$, consider the adjunction

$$\begin{tikzcd}
\mathbb{C}_{/\Gamma} \arrow[r, shift left=2, "\pi^*"] \arrow[r, phantom, "\top"] & \mathbb{C}_{/\Gamma,\sigma} \arrow[l, shift left=2, "\mathbb{C}_{/\pi}"]
\end{tikzcd}$$

Decomposition of Binding Arities

- For $\Gamma \vdash \sigma$, consider the adjunction

$$\begin{tikzcd}
\mathbb{C}_{/\Gamma} \arrow[r, shift left=2, "\pi^*"] \arrow[r, phantom, "\top"] & \mathbb{C}_{/\Gamma,\sigma} \arrow[l, shift left=2, "\mathbb{C}_{/\pi}"]
\end{tikzcd}$$

- It induces the adjoint situation:

$$\begin{tikzcd}[column sep=huge]
\widehat{\mathbb{C}_{/\Gamma}} \arrow[r, shift left=9, "(\pi^*)_*"] \arrow[r, phantom, shift left=6, "\top"] \arrow[r, <-, shift left=3, "\delta_\sigma" description] \arrow[r, phantom, "\top"] \arrow[r, shift right=3, "\epsilon_\sigma" description] \arrow[r, phantom, shift right=6, "\top"] \arrow[r, <-, shift right=9, "(\mathbb{C}_{/\pi})_!" description] & \widehat{\mathbb{C}_{/\Gamma,\sigma}}
\end{tikzcd}$$

- **Binding-arity decomposition lemma.**

  For $\Gamma \vdash \sigma$, the monad $(-)^{\mathbf{y}(\pi_{\Gamma\vdash\sigma})}$ on $\widehat{\mathbb{C}_{/\Gamma}}$ is induced by the adjunction $\varepsilon_\sigma \dashv \delta_\sigma$.

Binding Arities

- We thus obtain the following situation

$$\begin{tikzcd}
\widehat{\mathbb{C}_{/\Gamma}} \arrow[loop left, "(-)^{\mathbf{y}(\pi_{\Gamma\vdash\sigma})}"] & \cong & \widehat{\mathbb{C}}_{/\mathbf{y}(\Gamma)} \arrow[rr, shift left=6] \arrow[rr, phantom, shift left=4, "\top"] \arrow[rr, leftarrow, shift left=2, "\prod_{\sigma}" description] \arrow[rr, phantom, "\top"] \arrow[rr, shift right=2, "(\mathbf{y}\pi)^{*}" description] \arrow[rr, phantom, shift right=4, "\top"] \arrow[rr, leftarrow, shift right=6, "\sum_{\sigma}" description] & & \widehat{\mathbb{C}}_{/\mathbf{y}(\Gamma,\sigma)} & \cong & \widehat{\mathbb{C}_{/\Gamma,\sigma}}
\end{tikzcd}$$

Binding Arities

- We thus obtain the following situation

$$\begin{tikzcd}[column sep=huge]
\widehat{\mathbb{C}_{/\Gamma}} \arrow[loop below, "(-)^{\mathbf{y}(\pi_{\Gamma\vdash\sigma})}"] \arrow[r, phantom, "\cong"] & \widehat{\mathbb{C}}_{/\mathbf{y}(\Gamma)} \arrow[r, shift left=14] \arrow[r, phantom, shift left=10, "\top"] \arrow[r, leftarrow, shift left=6, "\Pi_\sigma" description] \arrow[r, phantom, shift left=1, "\top"] \arrow[r, shift right=4, "(\mathbf{y}\pi)^*" description] \arrow[r, phantom, shift right=9, "\top"] \arrow[r, leftarrow, shift right=14, "\Sigma_\sigma" description] & \widehat{\mathbb{C}}_{/\mathbf{y}(\Gamma,\sigma)} \arrow[r, phantom, "\cong"] & \widehat{\mathbb{C}_{/\Gamma,\sigma}}
\end{tikzcd}$$

Type-Dependent Binding

- Type-dependent binding operators.

$$\Pi_\sigma : \prod_\sigma \big(\mathbf{y}(\Gamma,\sigma)^* S\big) \to S \quad \text{in } \widehat{\mathbb{C}}$$

Binding Arities

- We thus obtain the following situation

$$\begin{tikzcd}[column sep=large]
\widehat{\mathbb{C}_{/\Gamma}} \arrow[loop left, "(-)^{\mathbf{y}(\pi_{\Gamma\vdash\sigma})}"] & \cong & \widehat{\mathbb{C}}_{/\mathbf{y}(\Gamma)} \arrow[r, shift left=9] \arrow[r, phantom, shift left=6, "\top"] \arrow[r, leftarrow, shift left=3, "\Pi_\sigma"] \arrow[r, phantom, "\top"] \arrow[r, shift right=3, "(\mathbf{y}\pi)^*"] \arrow[r, phantom, shift right=6, "\top"] \arrow[r, leftarrow, shift right=9, "\Sigma_\sigma"] & \widehat{\mathbb{C}}_{/\mathbf{y}(\Gamma,\sigma)} & \cong & \widehat{\mathbb{C}_{/\Gamma,\sigma}}
\end{tikzcd}$$

Type-Dependent Binding

- Type-dependent binding operators.

$$\Pi_\sigma : \prod_\sigma \big(\mathbf{y}(\Gamma,\sigma)^* S\big) \to S \quad \text{in } \widehat{\mathbb{C}}$$

Term Binding

- Term binding operators.

  For $\Gamma \vdash \sigma$ and $\Gamma, \sigma \vdash \tau$,

$$\begin{tikzcd}
\prod_\sigma(T_\tau) \arrow[r, "\lambda"] \arrow[d] & T \arrow[d] \\
\mathbf{y}(\Gamma) \arrow[r, "\mathbf{y}(\Pi_\sigma(\tau))"'] & S
\end{tikzcd} \quad \text{in } \widehat{\mathbb{C}}$$

Variables

$$\begin{tikzcd}
\mathbf{y}(\Gamma, \sigma) \arrow[rr] \arrow[dr, "\mathbf{y}(\pi_{\Gamma \vdash \sigma})"'] & & \mathsf{T}_\sigma \arrow[dl] \\
& \mathbf{y}(\Gamma) &
\end{tikzcd}$$

- This is to give a term $\Gamma, \sigma \vdash p : \sigma[\pi_{\Gamma \vdash \sigma}]$.

Variables

$$\begin{tikzcd}
\mathbf{y}(\Gamma,\sigma) \arrow[rr] \arrow[dr, "\mathbf{y}(\pi_{\Gamma\vdash\sigma})"'] & & T_\sigma \arrow[dl] \\
& \mathbf{y}(\Gamma) &
\end{tikzcd}$$

- This is to give a term $\Gamma,\sigma\vdash p:\sigma[\pi_{\Gamma\vdash\sigma}]$.

NB: **Categories with attributes/families**

The condition

$$\begin{tikzcd}
\mathbf{y}(\Gamma,\sigma) \arrow[rr, "\cong"] \arrow[dr, "\mathbf{y}(\pi_{\Gamma\vdash\sigma})"'] & & T_\sigma \arrow[dl] \\
& \mathbf{y}(\Gamma) &
\end{tikzcd}$$

is equivalent to *Dybjer's context comprehension* property: For all maps $f:\Delta\to\Gamma$ in $\mathbb{C}$ and terms $\Delta\vdash t:\sigma[f]$ there exists a unique map $\langle f,t\rangle:\Delta\to(\Gamma,\sigma)$ in $\mathbb{C}$ such that $p\big[\langle f,t\rangle\big]=t$ and $\pi_{\Gamma\vdash\sigma}\circ\langle f,t\rangle=f$.

- The initial model is the *classifying category*.

► [14, 18].

Substitution Structure

For $\Gamma \vdash \sigma$,

$$\begin{tikzcd} {\mathsf{T}_\Gamma}^{\mathbf{y}(\pi_{\Gamma\vdash\sigma})} \times \mathsf{T}_{\Gamma\vdash\sigma} \arrow[r] \arrow[d] & \mathsf{T}_\Gamma \arrow[d] \\ {\mathsf{S}_\Gamma}^{\mathbf{y}(\pi_{\Gamma\vdash\sigma})} \times \mathsf{T}_{\Gamma\vdash\sigma} \arrow[r] & \mathsf{S}_\Gamma \end{tikzcd} \quad \text{in } \widehat{\mathbb{C}}_{/\mathbf{y}(\Gamma)}$$

subject to axioms.

II

Algebraic Foundations

Kan Extensions

Every

$$f : \mathbb{X} \to \mathbb{Y}$$

induces

$$\begin{tikzcd}
\mathcal{P}\mathbb{X} \arrow[r, shift left=5, "f_*"] \arrow[r, shift right=5, "f_!"'] \arrow[r, phantom, shift left=2.5, "\top"] \arrow[r, phantom, shift right=2.5, "\top"] & \mathcal{P}\mathbb{Y} \arrow[l, "f^*" description]
\end{tikzcd}$$

where

$$\mathcal{P}\mathbb{C} =^{\mathrm{def}} \mathcal{S}et^{\mathbb{C}}$$

and

$$\begin{aligned}
f_* P\, y &= \mathrm{Ran}_f P\, y = \int_{x \in \mathbb{X}} [\, \mathbb{Y}(y, fx) \Rightarrow Px \,] \\
f^* Q\, x &= Q(fx) \\
f_! P\, y &= \mathrm{Lan}_f P\, y = \int^{x \in \mathbb{X}} \mathbb{Y}(fx, y) \times Px
\end{aligned}$$

▶ [4].

Generalised
Dependent Polynomial Functors

The class of

generalised dependent polynomial functors

is the closure under natural isomorphism of the functors

$$\mathcal{P}\mathbb{A} \longrightarrow \mathcal{P}\mathbb{B}$$

arising as composites

$$\begin{tikzcd} \mathcal{P}\mathbb{A} \arrow[r, "s^*"] & \mathcal{P}\mathbb{I} \arrow[r, "f_*"] & \mathcal{P}\mathbb{J} \arrow[r, "t_!"] & \mathcal{P}\mathbb{B} \end{tikzcd}$$

from diagrams

$$\begin{tikzcd} \mathbb{A} & \mathbb{I} \arrow[l, "s"'] \arrow[r, "f"] & \mathbb{J} \arrow[r, "t"] & \mathbb{B} \end{tikzcd}$$

in $\mathcal{C}at$.

$$t_! f_* s^* A\, b = \int^{j \in \mathbb{J}} \mathbb{B}(tj, b) \times \int_{i \in \mathbb{I}} \left[\mathbb{J}(j, fi) \Rightarrow A(si) \right]$$

Examples:

- *Dependent polynomial functors* (aka *indexed containers*) between slices of $\mathcal{S}et$ are [isomorphic to] generalised dependent polynomial functors.

▶ [21, 24].

- **Untyped abstract syntax**
  1. The rule

$$\frac{\Gamma \vdash t \qquad \Gamma \vdash t'}{\Gamma \vdash t(t')}$$

     has associated the generalised dependent polynomial endofunctor represented by

$$\mathsf{FinSet} \xleftarrow{\nabla_2} 2 \cdot \mathsf{FinSet} \xrightarrow{\nabla_2} \mathsf{FinSet} \xrightarrow{\mathrm{id}} \mathsf{FinSet}$$

- **Untyped abstract syntax**
  1. The rule

     $$\frac{\Gamma \vdash t \qquad \Gamma \vdash t'}{\Gamma \vdash t(t')}$$

     has associated the generalised dependent polynomial endofunctor represented by

     $$\mathsf{FinSet} \xleftarrow{\nabla_2} 2 \cdot \mathsf{FinSet} \xrightarrow{\nabla_2} \mathsf{FinSet} \xrightarrow{\mathrm{id}} \mathsf{FinSet}$$

  2. The rule

     $$\frac{\Gamma, x \vdash t}{\Gamma \vdash \lambda x.\, t}$$

     has associated the generalised dependent polynomial endofunctor represented by

     $$\mathsf{FinSet} \xleftarrow{+1} \mathsf{FinSet} \xrightarrow{\mathrm{id}} \mathsf{FinSet} \xrightarrow{\mathrm{id}} \mathsf{FinSet}$$

- **Simply typed abstract syntax**

  Let S be the set of simple types and write C for the category $\mathsf{FinSet}_{/\mathsf{S}}$ of S-sorted contexts.

  1. The rule

     $$\frac{\Gamma \vdash t : \tau' \Rightarrow \tau \qquad \Gamma \vdash t' : \tau'}{\Gamma \vdash t(t') : \tau}$$

     has associated the generalised dependent polynomial endofunctor represented by

$$\mathsf{C} \times \mathsf{S} \xleftarrow{[\mathrm{id}\times\Rightarrow,\mathrm{id}\times\pi_1]} 2 \cdot (\mathsf{C} \times \mathsf{S}^2) \xrightarrow{\nabla_2} \mathsf{C} \times \mathsf{S}^2 \xrightarrow{\mathrm{id}\times\pi_2} \mathsf{C} \times \mathsf{S}$$

- **Simply typed abstract syntax**

  Let S be the set of simple types and write C for the category $\mathsf{FinSet}_{/\mathsf{S}}$ of S-sorted contexts.

  1. The rule

     $$\frac{\Gamma \vdash t : \tau' \Rightarrow \tau \qquad \Gamma \vdash t' : \tau'}{\Gamma \vdash t(t') : \tau}$$

     has associated the generalised dependent polynomial endofunctor represented by

$$\mathsf{C} \times \mathsf{S} \xleftarrow{[\mathrm{id}\times\Rightarrow,\mathrm{id}\times\pi_1]} 2 \cdot (\mathsf{C} \times \mathsf{S}^2) \xrightarrow{\nabla_2} \mathsf{C} \times \mathsf{S}^2 \xrightarrow{\mathrm{id}\times\pi_2} \mathsf{C} \times \mathsf{S}$$

  2. The rule

     $$\frac{\Gamma, x : \sigma \vdash t : \tau}{\Gamma \vdash \lambda x.\, t : \sigma \Rightarrow \tau}$$

     has associated the generalised dependent polynomial endofunctor represented by

$$\mathsf{C} \times \mathsf{S} \xleftarrow{+\times\mathrm{id}} \mathsf{C} \times \mathsf{S} \times \mathsf{S} \xrightarrow{\mathrm{id}} \mathsf{C} \times \mathsf{S} \times \mathsf{S} \xrightarrow{\mathrm{id}\times\Rightarrow} \mathsf{C} \times \mathsf{S}$$

NB: The association of generalised dependent polynomial functors to rules extends to *polymorphic languages*.
In this context, the last component of the representation plays a crucial role as a *pattern-matching* constructor.

► [34].

- **Convolution monoidal closed structure**
  1. *Day’s convolution tensor product* is [isomorphic to] a generalised dependent polynomial functor.
  2. *Exponentiation to a representable* with respect to the closed structure associated to the convolution monoidal structure is a generalised polynomial functor.

- [7].

Generalised Inductive Dependent Polynomial Functors

The class of generalised dependent polynomial functors represented by diagrams of the form

$$\begin{tikzcd} \mathbb{A} & \coprod_{k \in K} L_k \cdot \mathbb{J}_k \arrow[l] \arrow[r, "\coprod_{k \in K} \nabla_{L_k}"] & \coprod_{k \in K} \mathbb{J}_k \arrow[r] & \mathbb{B} \end{tikzcd}$$

where L_k is finite for all $k \in K$,

- is closed under constants, identities, coproducts, finite products, and composition; and
- admits a (cartesian) differential calculus.

These functors

- are inductive (*viz.* finitary and preserve epis); and
- admit inductively-defined free algebras for equational systems.

Application Areas

- Data types.
  (*e.g.* reasoning)
- Type theory.
  (*e.g.* formalisation)
- Logical frameworks.
  (*e.g.* synthesis)
- Dependently-typed programming.
  (*e.g.* zippers)
- Concurrency theory.
  (*e.g.* models)

Pointers

[1] G. Birkhoff. On the structure of abstract algebras. *P. Camb. Philos. Soc.*, 31:433–454, 1935.

[2] A. Church. A formulation of the Simple Theory of Types. *JSL*, 5:56–68, 1940.

[3] A. Church. The calculi of lambda-conversion. Princeton University Press, 1941.

[4] D. Kan. Adjoint functors. *Trans. Amer. Math. Soc.*, 87(2):294–329, 1958.

[5] F.W. Lawvere. Functorial semantics of algebraic theories. Ph.D. Thesis, Columbia University, 1963. (*TAC*, No. 5, pp. 1–121, 2004.)

[6] F. Linton. Some aspects of equational theories. *Proc. Conf. on Categorical Algebra at La Jolla*, pp. 84–95, 1966.

[7] B. Day. On closed categories of functors. LNM 137, pp. 1–38, 1970.

[8] J.-Y. Girard. Une extension de l'interpretation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types.

Second Scandinavian Logic Symposium, 1971.

[9] J. Boardman and R. Vogt. Homotopy invariant algebraic structures on topological spaces. LNM 347, 1973.

[10] P. Martin-Löf. An intuitionistic theory of types. British Logic Colloquium, 1973.

[11] J. P. May. The geometry of iterated loop spaces. LNM 271, 1972.

[12] J. Reynolds. Towards a theory of type structure. Colloque sur la Programmation, 1974.

[13] P. Aczel. A general Church-Rosser theorem. Typescript, 1978

[14] J. Cartmell. Generalised algebraic theories and contextual categories. Ph.D. Thesis, University of Oxford, 1978.

[15] P. Aczel. Frege structures and the notion of proposition, truth and set. *The Kleene Symposium*, pp. 31–59, 1980.

[16] J. Cartmell. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied*

Logic, 32:209–243, 1986.

[17] M. Makkai. First-order logic with dependent sorts, with applications to category theory. Preprint, 1995.

[18] P. Dybjer Internal type theory. *TYPES'95*, LNCS 1158, pp. 120–134, 1996.

[19] G. Plotkin. Binding algebras: A step from universal algebra to type theory. Invited talk at *RTA-98*, 1998.

[20] M. Fiore, G. Plotkin, and D. Turi. Abstract syntax and variable binding. *LICS'99*, pp. 193–202, 1999.

[21] N. Gambino and M. Hyland. Wellfounded trees and dependent polynomial functors. *TYPES'04*, LNCS 3085, pp. 210–225, 2004.

[22] M. Fiore. Mathematical models of computational and combinatorial structures. *FoSSaCS 2005*, LNCS 3441, pp. 25–46, 2005.

[23] M. Hamana. Free Σ-monoids: A higher-order syntax with metavariables. APLAS 2004, LNCS 3202, pp. 348–363, 2005.

[24] T. Altenkirch, N. Ghani, P. Hancock, C. McBride, and P. Morris. Indexed containers. Unpublished

manuscript, 2008.

[25] M. Fiore. Second-order and dependently-sorted abstract syntax. *LICS'08*, pp. 57–68, 2008.

[26] M. Fiore and C.-K. Hur. Term equational systems and logics. *MFPS'08*, ENTCS 218, pp. 171–192, 2008.

[27] T. Altenkirch and P. Morris. Indexed containers. *LICS'09*, pp. 277–285, 2009.

[28] R. Blute, J. R. Cockett and R. Seely. Cartesian differential categories. *TAC*, Vol. 22, No. 23, pp. 622–672, 2009.

[29] M. Fiore. Algebraic Meta-Theories and Synthesis of Equational Logics. Research Programme, 2009.

[30] M. Fiore and C.-K. Hur. On the construction of free algebras for equational systems. *TCS*, 410(18):1704–1729, 2009.

[31] M. Fiore and C.-K. Hur. Second-order equational logic. *CSL 2010*, LNCS 6247, pp. 320–335, 2010.

[32] M. Fiore and O. Mahmoud. Second-order algebraic theories. *MFCS 2010*, LNCS 6281, pp. 368–380, 2010.

[33] M. Fiore and C.-K. Hur. On the mathematical synthesis of equational logics. LMCS-7(3:12), 2011.

[34] M. Hamana. Polymorphic abstract syntax via Grothendieck construction. *FoSSaCS 2011*, LNCS 6604, pp. 381–395, 2011.

[35] M. Hamana and M. Fiore. A foundation for GADTs and Inductive Families: Dependent polynomial functor approach. *ACM WGP*, 2011.