Algebra and Normalisation

Ohad Kammar

slides

course homepage

Zulip channel #Algebra and normalisation

Scottish Programming Languages and Verification Summer School (SPLV)
3–7 August 2026
University of Glasgow

THE UNIVERSITY of EDINBURGH informatics lfcs | Laboratory for Foundations of Computer Science | BayesCentre | Funded by: THE ROYAL SOCIETY

Normalisation in PL and Logic: what?

Goal

Identify equivalent syntactic representations

Example: equivalent programs

$x := 1;$
$x := 2$

vs $x := 2$ vs

if $x > 0$
then $x := 2$
else $x := 2$

Example: algebraic expressions

$5 + x + (0 + (-6))$ vs $-1 + x$

Example: λ-terms (not today)

$(\lambda x : \mathbb{Z}.x + 1)(y - 3)$ vs $-2 + y$

Normalisation: why?

Proofs

Deal with fewer cases

Robust systems: type-checkers, compilers, and partial evaluators

Behave in the same way and compositionally for equivalent programs

Search and synthesis

Enumerate candidates over smaller search-space

Normalisation and rewriting

Blind Rewriting (BRew)

Temptation:

- Represent abstract syntax tree (AST)
- Apply all transformations
- Everywhere
- Blindly

BRew Caveats

- **Wasteful** in time & space (NB: **E-graphs**)
- **Chaotic** and **non-robust**: type errors after tiny tweaks to source
- **Syntax-sensitive** or **limited** (e.g.: $x + y$ vs $y + x$)

Normalisation and rewriting

Disclaimer!

Don't confuse **BRew** with **strategic** & **term rewriting**:

- Deep maths & time-tested properties:
  confluence, **strong normalisation**, **Knuth-Bendix completion**, ...
- Deep connections to foundations of **logic**, **mathematics**, & **computation**
  (e.g.: λ-calculus)

This course

Goal

relate **normalisation** and **modern algebra**

This course

Goal

relate **normalisation** and **modern algebra**

By the end of the course, you will be able to:

- Identify **free algebras** and **free extensions** with **normalisation** tasks.
- Phrase and prove **representation theorems**.
- Design **data-structures** for normalisation based on representation theorems.

This course comprises:

- 3 lecture hours.
- 3–6 independent self-study hours.
- 40 independent exercises hours.

So the majority of the learning will happen **after** SPLV.

course URL

Zulip URL

Course structure

Goal

relate **normalisation** and **modern algebra**

course URL

Zulip URL

- Part 1: universal algebra
  - Motivation
  - Universal algebra basics: signatures & algebras
  - Modern algebra: homomorphisms & representation theorems
  - Expression normalisation: free models & equational logic
  - Modern partial evaluation: free extensions
  - Categorical algebra basics
- Part 2: second-order multi-sorted algebra
- ~~Part 3: second-order normalisation~~

Universal algebra basics

template:	$(X_s)_{s \in \mathrm{Sort}}$ family of **carrier** sets	&	$\left(f_i : X_{s_1} \times \cdots \times X_{s_n} \to X_r\right)_{i \in I}$ family of **operations**	
instances: ($\mathrm{Sort} = \{\star\}$)	$\mathbb{Z} := \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ integers	&	$(+) : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$	addition
			$0 : \mathbb{1} := \{\star\} \to \mathbb{Z}$	zero
			$(\cdot) : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$	multiplication
			$1 : \mathbb{1} \to \mathbb{Z}$	one

Universal algebra basics

template:	$(X_s)_{s \in \mathrm{Sort}}$ family of **carrier** sets	&	$\left(f_i : X_{s_1} \times \cdots \times X_{s_n} \to X_r\right)_{i \in I}$ family of **operations**	
instances:	$\mathbb{Z}$	&	$(+) : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$	addition
$(\mathrm{Sort} = \{\star\})$	integers		$0 : \mathbb{1} := \{\star\} \to \mathbb{Z}$	zero
$(\mathrm{Sort} = \mathbb{N} \times \mathbb{N})$	$(\mathbb{M}_{m \times n})_{m,n \in \mathbb{N}}$ real matrices	&	$(\cdot) : \mathbb{M}_{m \times k} \times \mathbb{M}_{k \times n} \to \mathbb{M}_{m \times n}$ $\mathrm{I}_n : \mathbb{1} \to \mathbb{M}_{n \times n}$	multiplication unit matrix

Universal algebra basics

template:	$(X_s)_{s \in \mathsf{Sort}}$ family of **carrier** sets	&	$\left(f_i : X_{s_1} \times \cdots \times X_{s_n} \to X_r\right)_{i \in I}$ family of **operations**	
instances: $(\mathsf{Sort} = \{\star\})$	$\mathbb{Z}$ integers	&	$(+) : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ $0 : \mathbb{1} := \{\star\} \to \mathbb{Z}$	addition zero
$(\mathsf{Sort} = \mathbb{N} \times \mathbb{N})$	$\left(\mathbb{M}_{m \times n}\right)_{m,n \in \mathbb{N}}$ real matrices	&	$(\cdot) : \mathbb{M}_{m \times k} \times \mathbb{M}_{k \times n} \to \mathbb{M}_{m \times n}$ $I_n : \mathbb{1} \to \mathbb{M}_{n \times n}$	multiplication unit matrix
$\left(\mathbb{\Lambda}_A := \begin{pmatrix}\text{closed}\\ \lambda\text{-terms}\\ \text{of type } A\end{pmatrix} / \equiv_{\beta\eta}\right)_{A \in \mathsf{SimpleType}}$ combinatory logic		&	$(@) : \mathbb{\Lambda}_{A \to B} \times \mathbb{\Lambda}_A \to \mathbb{\Lambda}_B$ $I := (\lambda x : A.x) : \mathbb{1} \to \mathbb{\Lambda}_{A \to A}$ $K := (\lambda x : A, y : B.x) : \mathbb{1} \to \mathbb{\Lambda}_{A \to B \to A}$ $S := \left(\lambda x : A \to B \to C, y : A \to B, z : A.\right.$ $\left.x\ z (y\ z)\right) : \mathbb{1} \to \mathbb{\Lambda}_{(\ldots)}$	application combinators

⋮

Universal algebra basics

Universal algebra

Study **common properties** of all such structures

(i.e.: universally applicable)

We'll do it in two steps:

- Generic syntax
- Generic interpreters

Running examples

Lecture focus on $\mathbb{Z}$ and $\mathbb{M}$
Exercises explore other examples similarly

Formal syntax

Finitary algebraic signature $\mathbf{S} = (\mathsf{Sort}_{\mathbf{S}}, \mathsf{Op}_{\mathbf{S}}, \mathsf{Arity}_{\mathbf{S}})$ (NB: indexed containers [Altenkirch et al. 14])

- **sort** set: $\mathsf{Sort}_{\mathbf{S}} \ni s$, written as decorated stars, e.g.: $\star_s$
- **operator** set: $\mathsf{Op}_{\mathbf{S}} \ni f$
- **arity** assignment $\mathsf{Arity}_{\mathbf{S}} : \mathsf{Op}_{\mathbf{S}} \to (\mathsf{List}\,\mathsf{Sort}_{\mathbf{S}}) \times \mathsf{Sort}_{\mathbf{S}}$

Instead of $\mathsf{Arity}_{\mathbf{S}} f = \left(\left[s_1, \ldots, s_n\right], r\right)$ write $((f : \star_{s_1} \times \cdots \times \star_{s_n}) \to \star_r) \in \mathbf{S}$

Example: magma

- $\mathsf{Sort}_{\mathsf{Magma}} := \{\_\}$ (i.e.: underline)
- $\mathsf{Op}_{\mathsf{Magma}} := \{(*), 1\}$
- $\mathsf{Arity}_{\mathsf{Magma}}(*) := \left(\left[\_,\_\right], \_\right)$ so $((*) : \underline{\star} \times \underline{\star} \to \underline{\star}) \in \mathsf{Magma}$
- $\mathsf{Arity}\,1 = \left([], \underline{\star}\right)$ so $1 : \underline{\star}$

Succinctly: $\mathsf{Magma} = \left\{(*) : \underline{\star} \times \underline{\star} \to \underline{\star}, 1 : \underline{\star}\right\}$

Example: reflexive quivers over set X

- $\mathsf{Sort}_{X\text{-}\mathsf{Quiver}} := \left\{\star_{m \times k} \middle| m, k \in X\right\}$
- X-Quiver $:=$
  $\left\{(\cdot) : \star_{m \times k} \times \star_{k \times n} \to \star_{m \times n} \middle| m, k, n \in X\right\}$
  $\cup \left\{\mathrm{I}_n \middle| n \in X\right\}$

Interpretation

Algebra $\mathsf{A} = \left(\left(\mathsf{A}_s \right)_{s \in \mathsf{Sort} \mathsf{A}_s}, \left(\mathsf{f}_\mathsf{A} \right)_{\mathsf{f} \in \mathsf{Op}} \right)$ for signature $\mathbf{S}$

- Sort-indexed family of **carrier** sets: A_s for every $s \in \mathsf{Sort}$.
- Op-indexed family of **operations** between the corresponding carriers: $f_\mathsf{A} : \mathsf{A}_{s_1} \times \cdots \times \mathsf{A}_{s_n} \to \mathsf{A}_r$ for every operator $(f : \star_{s_1} \times \cdots \times \star_{s_n} \to \star_r) \in \mathbf{S}$

Example: magmas

algebras A for the signature $\mathsf{Magma} = \left\{ (*) : \underline{\star} \times \underline{\star} \to \underline{\star}, 1 : \underline{\star} \right\}$ amount to:

- carrier set $\underline{\mathsf{A}}$
- operation $(*_\mathsf{A}) : \underline{\mathsf{A}}^2 \to \underline{\mathsf{A}}$
- constant $1_\mathsf{A} \in \underline{\mathsf{A}}$

Example: quivers

sets of matrices as algebra $\mathbb{M}$ for $\mathbb{N}\text{-Quiver} := \left\{ (\cdot_{m,k,n}) : \star_{m \times k} \times \star_{k \times n} \to \star_{m \times n} \middle| m, k, n \in \mathbb{N} \right\} \cup \left\{ \mathsf{I}_n \middle| n \in \mathbb{N} \right\}$

- carriers: $\mathbb{M}_{m \times n}$ (m rows, n columns)
- multiplication: $\left(a_{i,j}\right)_{i,j} \cdot \left(b_{j,\ell}\right)_{j,\ell} := \sum_j a_{i,j} \cdot b_{j,\ell}$
- identity matrix: $\mathsf{I}_n := \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}$

Algebraic expressions

Algebras give **meta-level** expressions:

Example: magma $\mathsf{A} = \left(\underline{\mathsf{A}}, (\cdot) : \underline{\mathsf{A}}^2 \to \underline{\mathsf{A}}, 0\right)$

$$(x \cdot y) \cdot z \qquad x \cdot (y \cdot z) \qquad y \cdot (1 \cdot x) \qquad \left(x, y, z \in \underline{\mathsf{A}}\right)$$

Example: $\mathbb{N}$-Quiver of matrices $\mathbb{M}$

$$x \cdot (y \cdot z) \qquad x \cdot (y \cdot z) \qquad x \cdot (\mathrm{I}_k \cdot y) \qquad \left(x \in \mathbb{M}_{n\times k}, y \in \mathbb{M}_{k\times \ell}, z \in \mathbb{M}_{\ell\times p}\right)$$

Summary

We formulated the template and its instances:

- **Signatures** specify the operators and their argument and return **sorts**
- **Algebras** interpret these operators as operations of appropriate type

Both let us talk about our examples, and many other examples, generically.

Typically, consider sub-class of algebras **validating** axioms (e.g., **associativity**).
We cover these **varieties** later.

Course structure

Goal

relate **normalisation** and **modern algebra**

course URL

Zulip URL

- Part 1: universal algebra
  - Motivation
  - Universal algebra basics: signatures & algebras
  - Modern algebra: homomorphisms & representation theorems
  - Expression normalisation: free models & equational logic
  - Modern partial evaluation: free extensions
  - Categorical algebra basics
- Part 2: second-order multi-sorted algebra
- ~~Part 3: second-order normalisation~~

Modern algebra

The hallmarks of modern algebra are:

- studying **varieties** of algebras defined through **equational axioms**; and
- establishing **representation theorems** relating structures.

Both aspects pertinent for normalisation:

- The same **signature** can express many varieties of algebras.
  It is the varietythat determines **normalisation**.
- The same **family** may carry multiple algebraic structures.
  The variety determines normalisation of **residual programs**.
- Representation theorems expose structural **properties**.

Structure-preserving-maps—**homomorphism**—are core to these relationship.

Structure-preserving maps

Homomorphism $h : \mathsf{A} \to \mathsf{B}$

between $\mathbf{S}$-algebras A, B:

- indexed family of functions $h_s : \mathsf{A}_s \to \mathsf{B}_s$, for all $s \in \mathsf{Sort}$;
- such that:

$$h_r\left(f_{\mathsf{A}}(a_1, \ldots, a_n)\right) = f_{\mathsf{B}}(h_{s_1} a_1, \ldots, h_{s_n} a_n)$$

for all $(f : \star_{s_1} \times \cdots \times \star_{s_n} \to \star_r) \in \mathbf{S}$ and $a_i \in \mathsf{A}_{s_i}$, for $i = 1, \ldots, n$

Examples

- Exponentiation is a magma homomorphism: $(\lambda x.2^x) : (\mathbb{Z}, (+), 0) \to (\mathbb{Q}, (\cdot), 1)$
- Uniform base-change is a quiver homomorphism:

$$\left(\lambda \mathsf{T} : \mathbb{M}_{m \times n}.\boldsymbol{J}_m \cdot \mathsf{T} \cdot \boldsymbol{J}_n^{-1}\right)_{m,n} : \mathbb{M} \to \mathbb{M}$$

for family of invertible matrices $\left(\boldsymbol{J}_n\right)_{n \in \mathbb{N}}$. (exercise: generalise)

Abstract syntax

Abstract syntax provides **object-level** terms, e.g., for first-class axioms
It'll be our first representation theorem:

$\mathsf{Term}_{\mathbf{S}} X$

over signature $\mathbf{S}$ and Sort-indexed family variables X— Sort-indexed inductive family:

$$\frac{}{x \in (\mathsf{Term}_{\mathbf{S}} X)_s}(x \in X_s) \qquad \frac{t_i \in (\mathsf{Term}_{\mathbf{S}} X)_{s_i} \text{ for all } i = 1, \ldots, n}{f(t_1, \ldots, t_n) \in (\mathsf{Term}_{\mathbf{S}} X)_r}((f : \star_{s_1} \times \cdots \times \star_{s_n} \to \star_r) \in \mathbf{S})$$

Write $X \vdash_{\mathbf{S}} t : \star_s$ when $t \in (\mathsf{Term}_{\mathbf{S}} X)_s$.
Data structure representing **abstract syntax trees** with $\mathbf{S}$-operators and X-variables.

Abstract syntax as an abstract data type

The operations available on the abstract syntax are:

- form a **variable**;
- form a composite **term** using an operator and appropriately sorted sub-terms;
- eliminate a term by **traversal**, given:
  - Motive $\underline{\mathsf{A}}_s$ for every sort;
  - Variable case: $e : X \to \underline{\mathsf{A}}$
  - Visitor for every AST node: $f_{\mathsf{A}} : \underline{\mathsf{A}}_{s_1} \times \cdots \times \underline{\mathsf{A}}_{s_1} \to \underline{\mathsf{A}}_r$

$h := \mathsf{fold}\left(\underline{\mathsf{A}}, e, f\right) : \mathsf{Term}_{\mathbf{S}} X \to \underline{\mathsf{A}}$
uniquely defined by:
$h(\mathsf{var}\, x) = e_s x \qquad h(f(t_1, \ldots, t_n)) = f_{\mathsf{A}}(h\, t_1, \ldots, h\, t_n)$

Example

$\mathbf{S} := \mathsf{Magma} \quad \underline{X} := \{\mathsf{x}, \mathsf{y}, \mathsf{z}\} \quad \mathsf{A} := (\mathbb{Z}, (+), 0)$
$e : X = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\} \to \mathbb{Z}$
$\mathsf{x} \mapsto 10 \quad \mathsf{y} \mapsto 5 \quad \mathsf{z} \mapsto -6$

$$\mathsf{fold}\,\mathsf{A}\,e\,((\mathsf{y} \cdot \mathsf{x}) \cdot (1 \cdot \mathsf{z})) = (\overbrace{e\,\mathsf{y} + e\,\mathsf{x}}^{5+10}) + (\overbrace{0 + e\,\mathsf{z}}^{0+(-6)}) = 9$$

Abstract syntax as an abstract data type

The operations available on the abstract syntax are:

- form a **variable**;
  I.e., a family of functions $\mathsf{var} : X \to \mathsf{Term}_{\mathbf{S}} X$
- form a composite **term** using an operator and appropriately sorted sub-terms;
  I.e., $\mathbf{S}$-algebra structure $\mathsf{F}_{\mathbf{S}} X$:

$$(\mathsf{F}_{\mathbf{S}} X)_s := (\mathsf{Term}_{\mathbf{S}} X)_s \qquad f_{\mathsf{F}_{\mathbf{S}} X} : (\mathsf{F}_{\mathbf{S}} X)_{s_1} \times \cdots \times (\mathsf{F}_{\mathbf{S}} X)_{s_n} \to (\mathsf{F}_{\mathbf{S}} X)_r$$

- eliminate a term by **traversal**, given:
  - Motive $\underline{\mathsf{A}}_s$ for every sort;
  - Variable case: $e : X \to \underline{\mathsf{A}}$
  - Visitor for every AST node:
    $f_{\mathsf{A}} : \underline{\mathsf{A}}_{s_1} \times \cdots \times \underline{\mathsf{A}}_{s_1} \to \underline{\mathsf{A}}_r$

  $h := \mathsf{fold}\left(\underline{\mathsf{A}}, e, f\right) : \mathsf{Term}_{\mathbf{S}} X \to \underline{\mathsf{A}}$
  uniquely defined by:
  $h(\mathsf{var}\, x) = e_s x \qquad h(f_{\mathsf{F}_{\mathbf{S}} X}(t_1, \ldots, t_n)) = f_{\mathsf{A}}(h\, t_1, \ldots, h\, t_n)$
  - i.e., each $\mathbf{S}$-algebra $\mathsf{A} = \left(\underline{\mathsf{A}}, \left(f_{\mathsf{A}}\right)\right)$;
  - and Sort-family map $e : X \to \underline{\mathsf{A}}$

  uniquely define:
  - **homomorphism** $\mathsf{fold}\,\mathsf{A}\, e : \mathsf{F}_{\mathbf{S}} X \to \mathsf{A}$
  - s.t. $\mathsf{fold}\,\mathsf{A}$ **extends** $e : X \to \underline{\mathsf{A}}$ along $\mathsf{var} : X \to \underline{\mathsf{F}_{\mathbf{S}} X}$.

Abstract syntax as a mathematical structure

Focus on the underlying maths:

- a family of functions $\mathsf{var} : X \to \mathsf{Term}_{\mathbf{S}} X$
- $\mathbf{S}$-algebra structure $\mathsf{F}_{\mathbf{S}} X$:
  $(\mathsf{F}_{\mathbf{S}} X)_s := (\mathsf{Term}_{\mathbf{S}} X)_s \qquad f_{\mathsf{F}_{\mathbf{S}} X} : (\mathsf{F}_{\mathbf{S}} X)_{s_1} \times \cdots \times (\mathsf{F}_{\mathbf{S}} X)_{s_n} \to (\mathsf{F}_{\mathbf{S}} X)_r$
- uniquely define:
  - **homomorphism** $\mathsf{fold}\,\mathsf{A}\,e : \mathsf{F}_{\mathbf{S}} X \to \mathsf{A}$
  - s.t. $\mathsf{fold}\,\mathsf{A}$ **extends** $e : X \to \underline{\mathsf{A}}$ along $\mathsf{var} : X \to \underline{\mathsf{F}_{\mathbf{S}} X}$.

  - i.e., each $\mathbf{S}$-algebra $\mathsf{A} = \left(\underline{\mathsf{A}}, \left(f_{\mathsf{A}}\right)\right)$;
  - and Sort-family map $e : X \to \underline{\mathsf{A}}$

Abstract syntax as a mathematical structure

Focus on the maths, rearrange into the situation:

- $\mathsf{F}_{\mathbf{S}}X$ is a $\mathbf{S}$-algebra:
$$(\mathsf{F}_{\mathbf{S}}X)_s \coloneqq (\mathsf{Term}_{\mathbf{S}}X)_s \quad f_{\mathsf{F}_{\mathbf{S}}X} : (\mathsf{F}_{\mathbf{S}}X)_{s_1} \times \cdots \times (\mathsf{F}_{\mathbf{S}}X)_{s_n} \to (\mathsf{F}_{\mathbf{S}}X)_r$$
- together with a family of functions $\mathsf{var} : X \to \mathsf{Term}_{\mathbf{S}}X$
- For every:
  - $\mathbf{S}$-algebra $\mathsf{A} = \left(\underline{\mathsf{A}}, \left(f_{\mathsf{A}}\right)\right)$
  - Sort-family map $e : X \to \underline{\mathsf{A}}$

  uniquely defines:
  - a **homomorphism** $\mathsf{fold}\,\mathsf{A}\,e : \mathsf{F}_{\mathbf{S}}X \to \mathsf{A}$
  - s.t. $\mathsf{fold}\,\mathsf{A}$ **extends** $e : X \to \underline{\mathsf{A}}$ along $\mathsf{var} : X \to \underline{\mathsf{F}_{\mathbf{S}}X}$.

Underlying abstraction:
algebra over X:

- $\mathbf{S}$-algebra $\mathsf{B} = \left(\underline{\mathsf{B}}, \left(f_{\mathsf{B}}\right)\right)$
- Sort-family map $d : X \to \underline{\mathsf{B}}$, the **environment/valuation**

and their **maps** $h : (\mathsf{B}, d) \to (\mathsf{A}, e)$:

- homomorphism $h : \mathsf{B} \to \mathsf{A}$
- preserving the environments:
  h extends e along d: $\quad e = h \circ d$

Abstract syntax as a mathematical structure

Focus on the maths, use underlying abstraction:

- $\mathsf{F}_{\mathbf{S}}X$ is a $\mathbf{S}$-algebra:

$$(\mathsf{F}_{\mathbf{S}}X)_s \coloneqq (\mathsf{Term}_{\mathbf{S}}X)_s \qquad f_{\mathsf{F}_{\mathbf{S}}X} : (\mathsf{F}_{\mathbf{S}}X)_{s_1} \times \cdots \times (\mathsf{F}_{\mathbf{S}}X)_{s_n} \to (\mathsf{F}_{\mathbf{S}}X)_r$$
$$\mathsf{var} : X \to \mathsf{Term}_{\mathbf{S}}X$$

- For every (A, e) $\mathbf{S}$-algebra over X, we have a map:

$$\mathsf{fold}\,\mathsf{A}\,e : \left(\mathsf{F}_{\mathbf{S}}X, \mathsf{var}\right) \to (\mathsf{A}, e)$$

Underlying abstraction:
algebra over X:

- $\mathbf{S}$-algebra $\mathsf{B} = \left(\underline{\mathsf{B}}, \left(f_{\mathsf{B}}\right)\right)$
- Sort-family map $d : X \to \underline{\mathsf{B}}$, the **environment/valuation**

and their **maps** $h : (\mathsf{B}, d) \to (\mathsf{A}, e)$:

- homomorphism $h : \mathsf{B} \to \mathsf{A}$
- preserving the environments:
  h extends e along d: $\quad e = h \circ d$

Representation theorem for abstract syntax

Our first **representation theorem**:

Theorem (abstract syntax initiality)

Let $\mathbf{S}$ *be a signature, and* X *a* $\mathsf{Sort}_{\mathbf{S}}$*-family. Then:*

- *abstract syntax trees for a signature* $\mathbf{S}$ *over a* $\mathsf{Sort}_{\mathbf{S}}$*-family of variables* X*;*
- *its* $\mathbf{S}$*-algebra structure* $\mathsf{F}_{\mathbf{S}}X$*:*

$$(\mathsf{F}_{\mathbf{S}}X)_s \coloneqq (\mathsf{Term}_{\mathbf{S}}X)_s \qquad f_{\mathsf{F}_{\mathbf{S}}X} : (\mathsf{F}_{\mathbf{S}}X)_{s_1} \times \cdots \times (\mathsf{F}_{\mathbf{S}}X)_{s_n} \to (\mathsf{F}_{\mathbf{S}}X)_r$$

- *the variable map* $\mathsf{var} : X \to \underline{\mathsf{F}_{\mathbf{S}}X}$

form an initial $\mathbf{S}$*-algebra over* X*:*
for every $\mathbf{S}$*-algebra* (A, e) *over* X *there is a unique map* $\mathsf{fold}\,\mathsf{A}\,e : \left(\mathsf{F}_{\mathbf{S}}X, \mathsf{var}\right) \to (\mathsf{A}, e)$.

Succinctly—A-natural bijection:

$$\frac{e : X \longrightarrow \underline{\mathsf{A}}}{\mathsf{fold}\,\mathsf{A}\,e : \mathsf{F}_{\mathbf{S}}X \longrightarrow \mathsf{A}}$$

recover universal arrow as mate of identity:

$$\frac{\mathsf{var} : X \longrightarrow \underline{\mathsf{F}_{\mathbf{S}}X} = \mathsf{Term}_{\mathbf{S}}X}{\mathrm{id} : \mathsf{F}_{\mathbf{S}}X \longrightarrow \mathsf{F}_{\mathbf{S}}X}$$

Representation theorems for abstract syntax

- The representation theorem isolates the **interface** to abstract syntax
- $(\mathsf{F}_{\mathbf{S}}X)_s \coloneqq (\mathsf{Term}_{\mathbf{S}}X)_s$ is one implementation

Another implementation: "concrete" syntax

Given coproduct diagram:

$$\mathsf{Op}_{\mathbf{S}} \xrightarrow{\mathsf{Op}} \mathsf{Symbol} \xleftarrow{\mathsf{Var}} X$$

Define $(\mathsf{C}_{\mathbf{S}}X)_s \subseteq \mathsf{List}\,\mathsf{Symbol}$:

$$\frac{}{[\mathsf{Var}x] \in (\mathsf{C}_{\mathbf{S}}X)_s}(x \in X_s) \qquad \frac{a_i \in (\mathsf{C}_{\mathbf{S}}X)_{s_i} \text{ for all } i = 1, \ldots, n}{(\mathsf{Op}f :: a_1 +\!\!+ \cdots +\!\!+ a_n) \in (\mathsf{C}_{\mathbf{S}}X)_r}((f : \star_{s_1} \times \cdots \times \star_{s_n} \to \star_r) \in \mathbf{S})$$

equip it with $\mathbf{S}$-algebra structure over X:

$$(\mathsf{G}_{\mathbf{S}}X)_s \coloneqq (\mathsf{C}_{\mathbf{S}}X)_s \qquad f_{\mathsf{G}_{\mathbf{S}}X}(a_1, \ldots, a_n) \coloneqq (\mathsf{Op}f :: a_1 +\!\!+ \cdots +\!\!+ a_n) \qquad \mathsf{var}_s x \coloneqq [\mathsf{Var}x]$$

Representation theorems for abstract syntax

Theorem ("concrete" syntax initiality)

Let $\mathbf{S}$ *be a signature, and* X *a* $\mathsf{Sort}_{\mathbf{S}}$*-family. Take* Symbol, $\mathsf{C}_{\mathbf{S}}X$ *from previous slide. The following* $\left(\mathsf{G}_{\mathbf{S}}X, \mathsf{var}\right)$*:*

$$(\mathsf{G}_{\mathbf{S}}X)_s \coloneqq (\mathsf{C}_{\mathbf{S}}X)_s \qquad f_{\mathsf{G}_{\mathbf{S}}X}(a_1, \ldots, a_n) \coloneqq (\mathsf{Op}f \mathbin{::} a_1 +\!\!+ \cdots +\!\!+ a_n) \qquad \mathsf{var}_s x \coloneqq [\mathsf{Var}x]$$

is a free $\mathbf{S}$*-algebra over* X*.*

(Exercise sheet explores this implementation and others in more detail.)

Course structure

Goal

relate **normalisation** and **modern algebra**

- Part 1: universal algebra
  - Motivation
  - Universal algebra basics: signatures & algebras
  - Modern algebra: homomorphisms & representation theorems
  - Expression normalisation: free models & equational logic
  - Modern partial evaluation: free extensions
  - Categorical algebra basics
- Part 2: second-order multi-sorted algebra
- ~~Part 3: second-order normalisation~~

course URL

Zulip URL

Main takeaway

Core idea: algebra and normalisation

representation theorem provides **normalisation** procedure

Already appears in **abstract syntax**—different concrete representations

"(x_ + _y)" vs. "$x + y$"

6 characters 3 characters

reflected by same abstract representation:

$$x + y \qquad \text{i.e.:} \begin{cases} \text{tree representation:} & (+)(x, y) \\ \text{“concrete” representation:} & \left[\mathsf{Op}\,(+), \mathsf{Var}\,x, \mathsf{Var}\,y\right] \end{cases}$$

reifying the representation as a string gives the **same** concrete syntax, say:

"$x + y$"

We'll unpack this for **monoids**.

Monoid representation

Monoid $\mathsf{A} = \left(\underline{\mathsf{A}}, (\cdot) : \underline{\mathsf{A}}^2 \to \underline{\mathsf{A}}, 1 : \mathsf{A}^2\right)$

associative unital magma:

- **associative:**

$$(x \cdot y) \cdot z = x \cdot (y \cdot z) \qquad (x, y, z \in \underline{\mathsf{A}})$$

- **unital:**

$$x \cdot 1 = x = 1 \cdot x \qquad (x \in \underline{\mathsf{A}})$$

monoid homomorphism $h : \mathsf{A} \to \mathsf{B}$ is a magma homomorphism

Monoid A over set X

- function $d : X \to \underline{\mathsf{A}}$, the **environment**

and thier maps $h : (\mathsf{A}, d) \to (\mathsf{B}, e)$:

- homomorphism $h : \mathsf{A} \to \mathsf{B}$
- preserving the environment: h extends e along d.

Monoid representation

Theorem

Let X be a set. The following data defines a free monoid over X:

$$\underline{\mathsf{F}_{\mathsf{Monoid}} X} \coloneqq \mathsf{List}\, X \qquad u \cdot v \coloneqq u +\!\!+ v \qquad 1 \coloneqq [] \qquad \eta^{\mathsf{Monoid}} x \coloneqq [x]$$

I.e.:

- $\mathsf{F}_{\mathsf{Monoid}} X \coloneqq (\mathsf{List}\, X, (+\!\!+), [])$ *is a monoid, and*
- *for every $e : X \to \underline{\mathsf{A}}$ monoid A over X there is a unique monoid homomorphism:*

$$\mathsf{eval}\, \mathsf{A}\, e : \mathsf{F}_{\mathsf{Monoid}} X \to \mathsf{A}$$

extending e along η^{Monoid}. It is given by:

$$\mathsf{eval}\, \mathsf{A}\, e\, [x_1, \ldots, x_n] \coloneqq 1 \cdot (e\, x_1) \cdot \cdots \cdot (e\, x_n)$$

$$\frac{e : X \longrightarrow \underline{\mathsf{A}} \text{ in } \mathbf{Set}}{\mathsf{eval}\, \mathsf{A}\, e : \mathsf{F}_{\mathsf{Monoid}} X \longrightarrow \mathsf{A} \text{ in } \mathbf{Monoid}}$$

Monoid normalisation

Core idea: algebra and normalisation

representation theorem provides **normalisation** procedure

We have a representation theorem, so we deserve a normalisation procedure:

Theorem (extensional normalisation, informally)

Let $\left(\mathsf{F}_{\mathsf{Monoid}} X, \eta^{\mathsf{Monoid}}\right)$ *be a free monoid over the set* X*, and* t, s *be monoid expressions with variables from* X*.*
The equation $t = s$ *is valid in all monoids iff it is valid in* $\mathsf{F}_{\mathsf{Monoid}} X$*.*

To phrase and prove this theorem precisely, we need to define **validity** of equations.

Validity

Equation $X \vdash_{\mathbf{S}} t = u : \star_s$ over signature $\mathbf{S}$

- set X
- terms $t, s \in (\mathsf{Term}_{\mathbf{S}} X)_s$

Satisfaction $(\mathsf{A}, e) \vDash (X \vdash t = u : \star_s)$

by $\mathbf{S}$-algebra $e : X \to \underline{\mathsf{A}} : X \to \underline{\mathsf{A}}$ over X:

$$\mathsf{fold}\,\mathsf{A}\,e\,t = \mathsf{fold}\,\mathsf{A}\,e\,s$$

Algebra A **validates** equation $\mathsf{A} \vDash (X \vdash t = u : \star_s)$ when every environment satisfies it:

$$\forall e : X \to \underline{\mathsf{A}}. \qquad (\mathsf{A}, e) \vDash (X \vdash t = u : \star_s)$$

Validity

Satisfaction $(\mathsf{A}, e) \vDash (X \vdash t = u : \star_s)$

by $\mathbf{S}$-algebra $e : X \to \underline{\mathsf{A}} : X \to \underline{\mathsf{A}}$ over X:

$$\mathsf{fold}\,\mathsf{A}\,e\,t = \mathsf{fold}\,\mathsf{A}\,e\,s$$

Algebra A **validates** equation $\mathsf{A} \vDash (X \vdash t = u : \star_s)$ when every environment satisfies it:

$$\forall e : X \to \underline{\mathsf{A}}. \qquad (\mathsf{A}, e) \vDash (X \vdash t = u : \star_s)$$

Examples

- The unit of the list monoid satisfies the equation:

  $$(\mathsf{F}_{\mathsf{Monoid}}\,\{x, y\}\,, \eta^{\mathsf{Monoid}}) \vDash (\{x, y\} \vdash (x \cdot 1) \cdot y = 1 \cdot (x \cdot y) : \underline{\star}$$

  since:

  $$([x] +\!\!+ []) +\!\!+ [y] = [x, y] = [] +\!\!+ ([x] +\!\!+ [y])$$

Validity

Satisfaction $(\mathsf{A}, e) \vDash (X \vdash t = u : \star_s)$

by $\mathbf{S}$-algebra $e : X \to \underline{\mathsf{A}} : X \to \underline{\mathsf{A}}$ over X:

$$\mathsf{fold}\,\mathsf{A}\,e\,t = \mathsf{fold}\,\mathsf{A}\,e\,s$$

Algebra A **validates** equation $\mathsf{A} \vDash (X \vdash t = u : \star_s)$ when every environment satisfies it:

$$\forall e : X \to \underline{\mathsf{A}}. \qquad (\mathsf{A}, e) \vDash (X \vdash t = u : \star_s)$$

Examples

- In fact, every monoid validates the equation:

$$\mathsf{A} \vDash (\{x, y\} \vdash (x \cdot 1) \cdot y = 1 \cdot (x \cdot y) : \underline{\star})$$

since:

$$\begin{array}{ccccc} & & \overset{\text{unitality}}{\downarrow} & & \overset{\text{unitality}}{\downarrow} \\ ((e\,x) \cdot 1) \cdot (e\,y) & = & (e\,x) \cdot (e\,y) & = & 1 \cdot ((e\,x) \cdot (e\,y)) \end{array}$$

for every environment $e : \{x, y\} \to \underline{\mathsf{A}}$

Validity

Satisfaction $(\mathsf{A}, e) \vDash (X \vdash t = u : \star_s)$

by $\mathbf{S}$-algebra $e : X \to \underline{\mathsf{A}} : X \to \underline{\mathsf{A}}$ over X:

$$\mathsf{fold}\,\mathsf{A}\,e\,t = \mathsf{fold}\,\mathsf{A}\,e\,s$$

Algebra A **validates** equation $\mathsf{A} \vDash (X \vdash t = u : \star_s)$ when every environment satisfies it:

$$\forall e : X \to \underline{\mathsf{A}}. \qquad (\mathsf{A}, e) \vDash (X \vdash t = u : \star_s)$$

Examples

- A monoid is a Magma-algebra A validating the associativity and unitality:

$$\mathsf{A} \vDash (\{x, y, z\} \vdash (x \cdot y) \cdot z = x \cdot (y \cdot z) : \underline{\star}) \qquad \mathsf{A} \vDash (\{x\} \vdash x \cdot 1 = x = 1 \cdot x : \underline{\star})$$

Validity

Satisfaction $(\mathsf{A}, e) \vDash (X \vdash t = u : \star_s)$

by $\mathbf{S}$-algebra $e : X \to \underline{\mathsf{A}} : X \to \underline{\mathsf{A}}$ over X:

$$\mathsf{fold}\,\mathsf{A}\,e\,t = \mathsf{fold}\,\mathsf{A}\,e\,s$$

Algebra A **validates** equation $\mathsf{A} \vDash (X \vdash t = u : \star_s)$ when every environment satisfies it:

$$\forall e : X \to \underline{\mathsf{A}}. \qquad (\mathsf{A}, e) \vDash (X \vdash t = u : \star_s)$$

Lemma (homomorphic images preserve equations)

Let $h : \mathsf{A} \to \mathsf{B}$ *be a* $\mathbf{S}$*-homomorphism. Then:*

$$(\mathsf{A}, e \circ h) \vDash (X \vdash t = u : \star_s) \iff (\mathsf{B}, e) \vDash (X \vdash t = u : \star_s)$$

So if $h : \underline{\mathsf{A}} \twoheadrightarrow \underline{\mathsf{B}}$ *is* **surjective**, *then:* $\mathsf{A} \vDash (X \vdash t = u : \star_s) \implies \mathsf{B} \vDash (X \vdash t = u : \star_s)$.

Monoid normalisation

Theorem (extensional normalisation)

Let $\left(\mathsf{F}_{\mathsf{Monoid}}X, \eta^{\mathsf{Monoid}}\right)$ *be a free monoid over* X*, and* $X \vdash t = s : \underline{\star}$ *an equation. The following are equivalent (TFAE):*

1. *The unit satisfies the equation:* $(\mathsf{F}_{\mathsf{Monoid}}X, \eta^{\mathsf{Monoid}}) \vDash (X \vdash t = u : \underline{\star})$
2. *The free monoid validates the equation:* $\mathsf{F}_{\mathsf{Monoid}}X \vDash (X \vdash t = u : \underline{\star})$
3. *Every monoid* $\mathsf{A} \in \mathbf{Monoid}$ *validates the equation:* $\mathsf{A} \vDash (X \vdash t = u : \underline{\star})$

Proof idea, implemented over next few slides

By suitable instantiation: (1) $\Longleftarrow$ (2) $\Longleftarrow$ (3). For (3) $\Longleftarrow$ (1):

- Develop proof theory for univeral algebra: **equational logic**
- By **soundness** of equational logic, a provable equation is valid in all monoids
- The **quotient** of the free magma by **provability** is a **free monoid**. ■

Proving this theorem for monoids is as easy as proving it for every variety of algebras.

Varieties of algebras

Presentation $\mathcal{P} = (\mathbf{S}, \mathbf{Ax})$

- signature $\mathbf{S}$
- set $\mathbf{Ax}$ of $\mathbf{S}$-equations $(X \vdash s = t) \in \mathbf{Ax}$, the **axioms**

$\mathcal{P}$-**model**: $\mathbf{S}$-algebra A validating every axiom $\mathsf{A} \vDash (X \vdash t = s : \star_s) \in \mathbf{Ax}$

Example: monoids

Monoid $= \left(\mathsf{Magma}, \mathbf{Ax}_{\mathsf{Monoid}}\right)$ presentation:

$$\{x, y, z\} \vdash (x \cdot y) \cdot z = x \cdot (y \cdot z) : \underline{\star} \qquad \{x\} \vdash x \cdot 1 = x = 1 \cdot x : \underline{\star}$$

monoids are Monoid-models

Quotienting an algebra

Congruence ($\equiv$) over $\mathbf{S}$-algebra A

Sort-indexed family of equivalence relations ($\equiv_s$) over A_s that respects the operations:

$$\frac{a \in \mathsf{A}_s}{a \equiv_s a}(\mathsf{refl}) \qquad \frac{a \equiv_s b}{b \equiv_s a}(\mathsf{sym}) \qquad \frac{a \equiv_s b \qquad b \equiv_s c}{a \equiv_s c}(\mathsf{trans})$$

$$\frac{X \vdash e_1 \equiv e_2 \qquad X \vdash t : \star_s}{\mathsf{fold}\,\mathsf{A}\,e_1\,t \equiv_s \mathsf{fold}\,\mathsf{A}\,e_2\,t}(\mathsf{cong})$$

where $X \vdash e_1 \equiv e_2$ for environments $e_1, e_2 : X \to \underline{\mathsf{A}}$ when:

$$e_1\,x \equiv_s e_2\,x \text{ for every } (x : \star_s) \in X.$$

Example: length parity over $\mathsf{F}_{\mathsf{Monoid}} X = (\mathsf{List}\,X, (+\!\!+), [])$.

$u \equiv_{\underline{\star}} v$ when u,v both odd-length or both even-length is a congruence

Quotienting an algebra

Quotient $\mathbf{S}$-algebra $\mathsf{A}/(\equiv)$

(where $(\equiv)$ congruence over A)

$$(\mathsf{A}/(\equiv))_s := \mathsf{A}_s/(\equiv_s) \qquad f_{\mathsf{A}/(\equiv)}([a_1]_{s_1}, \ldots, [a_n]_{s_n}) := [f_{\mathsf{A}}(a_1, \ldots, a_n)]_r$$
$$((f : \star_{s_1} \times \cdots \times \star_{s_n} \to s_r) \in \mathbf{S})$$

Lemma

The quotient map is a $\mathbf{S}$*-homomorphism* $[-]\_ : \mathsf{A} \to \mathsf{A}/(\equiv)$*, it is the universal* $(\equiv)$**-respecting homomorphism** *out of* A*:*

- *for all* $a \equiv_s b$*, we have* $[a]_s = [b]_s$*;*
- *for every* $h : \mathsf{A} \to \mathsf{B}$ *satisfying* $a \equiv_s b \implies h_s a = h_s b$*, there is a unique homomorphism* $\therefore h : \mathsf{A}/(\equiv) \to \mathsf{B}$ *extending* h *along* $[-]$*.*

Equational logic: preliminaries

A substitution $X \vdash_{\mathbf{S}} \theta : Y$

family map $\mathsf{Term}_{\mathbf{S}} X \xleftarrow{\theta} Y$, i.e.: $(X \vdash \theta\, x : \star_s) \mapsto (x : \star_s) \in Y$

Substitution operation (given $X \vdash_{\mathbf{S}} \theta : Y$)

family map $- [\theta] : \mathsf{Term}_{\mathbf{S}} X \xleftarrow{\mathsf{fold}(\mathsf{F}_{\mathbf{S}} X)\, \theta} \mathsf{Term}_{\mathbf{S}} Y$, i.e.:

$$(X \vdash t\,[\theta] : \star_s) \mapsto (Y \vdash t : \star_s)$$

Example

Substitution does what you think it should:

$$((x \cdot y) \cdot (x \cdot 1))\overbrace{[x \mapsto z, y \mapsto 1 \cdot z]}^{\theta :=} = ((\theta x) \cdot (\theta y)) \cdot ((\theta x) \cdot 1) = (z \cdot (1 \cdot z)) \cdot (z \cdot 1)$$

Equational logic

Provability $X \vdash_{\mathcal{P}} t = u : \star_s$

$\mathbf{S}$-congruence closure of axiom:

$$\frac{}{X \vdash_{\mathcal{P}} t = t : \star_s}(\text{refl}) \qquad \frac{X \vdash_{\mathcal{P}} t = u : \star_s}{X \vdash_{\mathcal{P}} u = t : \star_s}(\text{sym})$$

$$\frac{X \vdash_{\mathcal{P}} t = u : \star_s \qquad X \vdash_{\mathcal{P}} u = v : \star_s}{X \vdash_{\mathcal{P}} t = v : \star_s}(\text{trans})$$

$$\frac{X \vdash_{\mathcal{P}} \theta_1 = \theta_2 : Y \quad Y \vdash_{\mathbf{S}} t : \star_s}{X \vdash_{\mathcal{P}} t\left[\theta_1\right] = t\left[\theta_1\right] : \star_s}(\text{cong}) \qquad \frac{(Y \vdash t = u : \star_s) \in \mathbf{Ax}_{\mathcal{P}} \quad X \vdash_{\mathbf{S}} \theta : Y}{X \vdash_{\mathcal{P}} t[\theta] = u[\theta] : \star_s}(\text{axiom})$$

where $X \vdash_{\mathcal{P}} \theta_1 = \theta_2 : Y$ when $X \vdash \theta_1 x = \theta_2 x : \star_r$ for all $(x : \star_r) \in Y$.

Equational logic

Example

We write equational proof referring to axioms only:

$$\{x, y\} \vdash_{\mathsf{Monoid}} (x \cdot 1) \cdot y = x \cdot y = 1 \cdot (x \cdot y)$$

(unitality, unitality)

but mean derivations:

$$\dfrac{\dfrac{\dfrac{\dfrac{(\{x\} \vdash x \cdot 1 = x) \in \mathbf{Ax}}{\{x, y\} \vdash x \cdot 1 = x}(\text{axiom}) \qquad \dfrac{}{y = y}(\text{refl})}{\{x, y\} \vdash (z \mapsto x \cdot 1, y \mapsto y) = (z \mapsto x, y \mapsto y) : \{z, y\}}}{(x \cdot 1) \cdot y = x \cdot y}(\text{cong}) \qquad \dfrac{(z = 1 \cdot z) \in \mathbf{Ax}}{x \cdot y = 1 \cdot (x \cdot y)}(\text{axiom})}{\{x, y\} \vdash (x \cdot 1) \cdot y = 1 \cdot (x \cdot y) : \underline{\star}}(\text{trans})$$

Relating proofs and models

Theorem (soundness)

For every $\mathcal{P}$-model A:

$$X \vdash_{\mathcal{P}} t = u : \star_s \implies \mathsf{A} \vDash (X \vdash t = u : \star_s)$$

Proof

By induction on the derivation of the provability relation.

- **reflexivity, symmetry, transitivity** use corresponding property of equality, e.g.:

  $$(\mathsf{A}, e) \vDash t = u \implies \mathsf{fold}\,\mathsf{A}\,e\,t = \mathsf{fold}\,\mathsf{A}\,e\,u \implies \mathsf{fold}\,\mathsf{A}\,e\,u = \mathsf{fold}\,\mathsf{A}\,e\,t \implies (\mathsf{A}, e) \vDash u = t$$

- **congruence** uses Leibniz's law, and basic proeprties of substitutions:

  $$\underset{\text{substitution lemma}}{\mathsf{fold}\,\mathsf{A}\,e\,(t\,[\theta_1])} = \mathsf{fold}\,\mathsf{A}\,(\mathsf{fold}\,\mathsf{A}\,e \circ \theta_1)\,(t) = \underset{\text{IH+Leibniz's law}}{\mathsf{fold}\,\mathsf{A}\,(\mathsf{fold}\,\mathsf{A}\,e \circ \theta_2)\,(t)} = \mathsf{fold}\,\mathsf{A}\,e\,(t\,[\theta_2]) \implies (\mathsf{A}, e) \vDash t\,[\theta_1] = t\,[\theta_2]$$

- **axiom** uses the $\mathcal{P}$-model assumpion. ■

Free model by quotienting

Theorem

Let $\mathcal{P}$ be a presentation and X a family. We have a free $\mathcal{P}$-model over X:

$$\mathsf{F}_{\mathcal{P}}X \coloneqq \mathsf{F}_{\mathbf{S}}X/(X \vdash (=)) \qquad \eta^{\mathcal{P}} : X \xrightarrow{\mathsf{var}} \mathsf{F}_{\mathbf{S}}X \xrightarrow{[-]} \big(\mathsf{F}_{\mathbf{S}}X/(=)\big) =: \underline{\mathsf{F}_{\mathcal{P}}X}$$

I.e.,for every $\mathcal{P}$-model A and family map $e : X \to \underline{\mathsf{A}}$ there is a unique $\mathbf{S}$-homomorphism extending e along $\eta^{\mathcal{P}}$ given by $\therefore \mathsf{fold}\,\mathsf{A}\,e : \mathsf{F}_{\mathcal{P}}X \to \mathsf{A}$.

Proof assembles existing parts:

$$\frac{e : X \longrightarrow \underline{\mathsf{A}} \text{ family map}}{\mathsf{fold}\,\mathsf{A}\,e : \mathsf{F}_{\mathbf{S}}X \longrightarrow \mathsf{A} \ \mathbf{S}\text{-homomorphism, by freeness}}$$

$$\frac{\mathsf{fold}\,\mathsf{A}\,e : \mathsf{F}_{\mathbf{S}}X \longrightarrow \mathsf{A} \ \mathbf{S}\text{-homomorphism, by freeness}}{\therefore \mathsf{fold}\,\mathsf{A}\,e : \mathsf{F}_{\mathcal{P}}X \longrightarrow \mathsf{A} \ \mathbf{S}\text{-homomorphism, by lemma}}$$

■

Normalisation via representability

Theorem (extensional normalisation)

Let $(\mathsf{F}_{\mathcal{P}}X, \eta^{\mathcal{P}})$ *be a free* $\mathcal{P}$*-model over a family* X*, and* $X \vdash t = s : \underline{\star}$ *an equation. The following are equivalent (TFAE):*

1. *The unit satisfies the equation:* $(\mathsf{F}_{\mathcal{P}}X, \eta^{\mathcal{P}}) \vDash (X \vdash t = u : \underline{\star})$
2. *The given free* $\mathcal{P}$*-model validates the equation:* $\mathsf{F}_{\mathcal{P}}X \vDash (X \vdash t = u : \underline{\star})$
3. *Every* $\mathcal{P}$*-model* A *validates the equation:* $\mathsf{A} \vDash (X \vdash t = u : \underline{\star})$
4. *The equation is provable from* $\mathcal{P}$*:* $X \vdash_{\mathcal{P}} t = u : \underline{\star}$

Proof

Suitable instantiation/soundness: (1) $\Longleftarrow$ (2) $\Longleftarrow$ (3) $\Longleftarrow$ (4). For (4) $\Longleftarrow$ (1):

- The free $\mathcal{P}$-models are canonically isomorphic $h : (\mathsf{F}_{\mathcal{P}}X, \eta^{\mathcal{P}}) \xrightarrow{\cong} (\mathsf{F}_{\mathbf{S}}X/(=), [\mathsf{var}])$.
- Homo. images preserve equations $\Longrightarrow$ $(\mathsf{F}_{\mathbf{S}}X/(=), [\mathsf{var}]) \vDash t = u \Longrightarrow X \vdash_{\mathcal{P}} t = u$

as we wanted. ∎

Summary

Core idea: algebra and normalisation

representation theorem provides extensional **normalisation** procedure

Refinement: **effective** algebra and normalisation

effective representation theorem provides **normalisation** procedure

Methodology:

- Prove a constructive representation theorem for the free **setoid** $\mathcal{P}$-model.
- Carry the normalisation proof over setoids, extracting a proof derivation.

For more details, see:

Frex: dependently-typed algebraic simplification. Guillaume Allais, Edwin Brady, Nathan Corbyn, Ohad Kammar, and Jeremy Yallop. 2025. ACM Program. Lang. 9, ICFP, Article 237, DOI: 10.1145/3747506.

Course structure

Goal

relate **normalisation** and **modern algebra**

- Part 1: universal algebra
  - Motivation
  - Universal algebra basics: signatures & algebras
  - Modern algebra: homomorphisms & representation theorems
  - Expression normalisation: free models & equational logic
  - Modern partial evaluation: free extensions
  - Categorical algebra basics
- Part 2: second-order multi-sorted algebra
- ~~Part 3: second-order normalisation~~

course URL

Zulip URL

Modern partial evaluation: motivation

Partially-static representations

Expressions mixing:

- constants;
- algebraic operators; and
- variables

E.g., over the commutative monoid $(\mathbb{Q}, (+), 0)$:

$$-\frac{3}{4}+x+\frac{1}{4}+y+\frac{1}{2} = x+y$$

Variables can represent:

- function arguments;
- expression holes;
- delayed evaluation place-holders; or
- sub-expressions outside the signature $\mathbf{S}$ (e.g., uninterpreted function calls)

Partially-static representations

Extension (E, s) of $\mathcal{P}$-model A

- $\mathcal{P}$-model E;
- $\mathcal{P}$-homomorphism $s : \mathsf{A} \to \mathsf{E}$;

Extension homomorphism $h : (\mathsf{E}, s) \to (\mathsf{D}, r)$:

- $\mathcal{P}$-model homomorphism $h : \mathsf{E} \to \mathsf{D}$;
- extending $r : \mathsf{A} \to \mathsf{D}$ along $s : \mathsf{A} \to \mathsf{E}$.

I.e., the **coslice category** $\mathsf{A} / \mathbf{Mod}\,\mathcal{P}$

Freely extending commutative monoids

Theorem (commutative monoid frex representation)

Let $\mathsf{A} = \left(\underline{\mathsf{A}}, (+), 0\right)$ *be a* **commutative monoid**, *and* X *a set.*
*The free extension (***frex***) of* A *by* X *is given by* **alternating lists**:

$$\underline{\mathsf{A}[X]} \coloneqq \left\{ \left[a_0, x_1, a_1, \ldots, x_n, a_n\right] \middle| n \in \mathbb{N}, a_i \in \underline{\mathsf{A}}, x_i \in X \right\} \qquad 0_{\mathsf{A}[X]} \coloneqq \left[0_{\mathsf{A}}\right]$$

$$\left[a_0, x_1, a_1, \ldots, x_n, a_n\right] +_{\mathsf{A}[X]} \left[b_0, y_1, b_1, \ldots, y_k, b_k\right]$$
$$\coloneqq \left[a_0, x_1, a_1, \ldots, x_n, a_n +_{\mathsf{A}} b_0, y_1, b_1, \ldots, y_k, b_k\right]$$

$$s : \mathsf{A} \to \mathsf{A}[X] \quad s\, a \coloneqq [a] \qquad \eta : X \to \underline{\mathsf{A}[X]} \quad \eta\, x \coloneqq \left[0_{\mathsf{A}}, x, 0_{\mathsf{A}}\right]$$

I.e., for every extension $r : \mathsf{A} \to \mathsf{E}$ *and environment map* $e : X \to \underline{\mathsf{E}}$, *there is a unique extension homomorphism* $[r, e] : (\mathsf{A}[X], s) \to (\mathsf{E}, r)$ *extending* e *along* η, *given by:*

$$[r, e]\left[a_0, x_1, a_1, \ldots, x_n, a_n\right] \coloneqq (r\, a_0) +_{\mathsf{E}} (e\, x_1) +_{\mathsf{E}} (r\, a_1) +_{\mathsf{E}} \cdots +_{\mathsf{E}} (e\, x_n) +_{\mathsf{E}} (r\, a_n)$$

Presenting extensions

Free extensions of $\mathcal{P}$-models reduce to free models:

Evaluation presentation $\mathsf{Eval}\,\mathsf{A}$ in $\mathbf{S}$-algebra A

- Add constants for each A-element: $\mathbf{S}_{\mathsf{Eval}\,\mathsf{A}} \coloneqq \mathbf{S} \cup \left\{ \underline{a} : \star_s \middle| a \in \mathsf{A}_s \right\}$
- Evaluation equation for each operator:

$$\mathbf{Ax}_{\mathsf{Eval}\,\mathsf{A}} \coloneqq \left\{ \begin{array}{l} \{x_1, \ldots, x_n\} \vdash \\ f(\underline{a}_1, \ldots, \underline{a}_n) = \underline{f_{\mathsf{A}}(a_1, \ldots, a_n)} : \star_r \end{array} \middle| \begin{array}{l} (f : \star_{s_1} \times \cdots \times \star_{s_n} \to \star_r) \in \mathbf{S}, \\ a_i \in \mathsf{A}_{s_i} \end{array} \right\}$$

For example:

$$\left(\underline{\frac{1}{2}} + - \underline{\frac{1}{4}} \right) = \underline{\frac{1}{4}} \in \mathbf{Ax}_{\mathsf{Eval}(\mathbb{Q},+,0)}$$

For $\mathcal{P}$-model A, the **A-extension presentation** $\mathcal{P}_{\mathsf{Eval}\,\mathsf{A}}$:

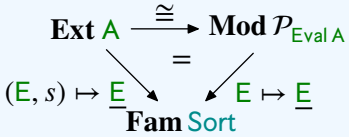
$$\mathbf{S}_{\mathcal{P}_{\mathsf{Eval}\,\mathsf{A}}} \coloneqq \mathbf{S}_{\mathsf{Eval}\,\mathsf{A}} \qquad \mathbf{Ax}_{\mathcal{P}_{\mathsf{Eval}\,\mathsf{A}}} \coloneqq \mathbf{Ax}_{\mathcal{P}} \cup \mathbf{Ax}_{\mathsf{Eval}\,\mathsf{A}}$$

Presenting extensions

There is nothing more to the theory of frex than our current development:

Theorem (extensions as models)

Let A *be a* $\mathcal{P}$*-model.*
The categories of A*-extensions and* $\mathcal{P}_{\mathsf{Eval}\,\mathsf{A}}$*-models are isomorphic:*



so the frex of A *by* X *is the free* $\mathcal{P}_{\mathsf{Eval}\,\mathsf{A}}$*-model over* X*.*

This perspective sometimes help in constructing the frex.

Course structure

Goal

relate **normalisation** and **modern algebra**

- Part 1: universal algebra
  - Motivation
  - Universal algebra basics: signatures & algebras
  - Modern algebra: homomorphisms & representation theorems
  - Expression normalisation: free models & equational logic
  - Modern partial evaluation: free extensions
  - Categorical algebra basics
- Part 2: second-order multi-sorted algebra
- ~~Part 3: second-order normalisation~~

course URL

Zulip URL