The Programming of Algebra

Fritz Henglein Robin Kaarsgaard Mikkel Kragh Mathiesen

April 2, 2022

Goals

Query languages should be:

- Efficient
- Expressive
- Reasonable

Answer: algebraic and categorical structure.

Module, Algebra

A K-*module* V comprises:

- $0 : V$
- $(+) : V \times V \to V$ (associative, commutative)
- $(\cdot) : K \times V \to V$ (associative, distributive)

A K-*algebra* V additionally comprises:

- $(\cdot) : V \times V \to V$ (associative, commutative, distributes over $+$)
- $1 : V$ (if algebra is *unital*)

Scalar

The scalar module K is generated by 1. Algebra structure inherited from ring. Elimination principle: given $f : \{1\} \to |V|$

$$\hat{f}(1) = f(1)$$

Hom characterisation:

$$K \to_1 V \cong V$$

Tensor product

The tensor product $U \otimes V$ of U and V is generated by $u \otimes v$. Algebra structure:

$$(u_1 \otimes v_1) \cdot (u_2 \otimes v_2) = (u_1 \cdot u_2) \otimes (v_1 \cdot v_2)$$

Elimination principle: given $f : U \to_1 V \to_1 W$

$$\hat{f}\,(u \otimes v) = f(u, v)$$

Hom characterisation:

$$U \otimes V \to_1 W \cong U \to_1 V \to_1 W$$

Free module

Free module $\mathbf{F}_K[A]$ is generated by $\langle a \rangle$. Algebra structure:

$$\langle a \rangle \cdot \langle a \rangle = \langle a \rangle$$
$$\langle a \rangle \cdot \langle b \rangle = 0 \quad \text{for } a \neq b$$

Elimination principle: given $f : A \to |V|$

$$\hat{f} : \mathbf{F}_K[A] \to_1 V$$
$$\hat{f}\,(\langle x \rangle) = f(x)$$

Hom characterisation:

$$\mathbf{F}_K[A] \to_1 V \cong (A \to |V|)$$

Free module

Compact free module $\mathbf{F}^*_K[A]$ is generated by $\langle a \rangle$ and $*$. Algebra structure:

$$
\begin{aligned}
* \cdot y &= y & \qquad \langle a \rangle \cdot \langle a \rangle &= \langle a \rangle \\
x \cdot * &= x & \qquad \langle a \rangle \cdot \langle b \rangle &= 0 \quad \text{for } a \neq b
\end{aligned}
$$

Elimination principle: given $f : A \to |V|$ and $u : |V|$

$$
\begin{aligned}
\hat{f} &: \mathbf{F}^*_K[A] \to_1 V \\
\hat{f}(\langle x \rangle) &= f(x) \\
\hat{f}(*) &= u
\end{aligned}
$$

Hom characterisation:

$$\mathbf{F}^*_K[A] \to_1 V \cong (A \to |V|) \times (\{*\} \to |V|)$$

Finite map

Finite map module over A and U is generated by $a \mapsto u$.
Algebra structure:

$$
\begin{aligned}
(a \mapsto u) \cdot (a \mapsto v) &= a \mapsto u \cdot v \\
(a \mapsto u) \cdot (b \mapsto v) &= 0 \quad \text{for } a \neq b
\end{aligned}
$$

Elimination principle: given $f : A \to |U \to_1 V|$

$$
\begin{aligned}
\hat{f} &: (A \Rightarrow U) \to_1 V \\
\hat{f}\,(a \mapsto v) &= f(a, v)
\end{aligned}
$$

Hom characterisation:

$$(A \Rightarrow U) \to_1 V \cong (A \to |U \to_1 V|)$$

Compact map

Compact map module over A and U is generated by $a \mapsto u$ and $* \mapsto u$.
Algebra structure:

$$
\begin{aligned}
(* \mapsto u) \cdot (a \mapsto v) &= a \mapsto u \cdot v & \quad (a \mapsto u) \cdot (a \mapsto v) &= a \mapsto u \cdot v \\
(a \mapsto u) \cdot (* \mapsto v) &= a \mapsto u \cdot v & \quad (a \mapsto u) \cdot (b \mapsto v) &= 0 \quad \text{for } a \neq b \\
(* \mapsto u) \cdot (* \mapsto v) &= * \mapsto u \cdot v
\end{aligned}
$$

Elimination principle: given $f : A \to |U \to_1 V|$ and $u : U \to_1 V$

$$
\begin{aligned}
\hat{f} &: (A \Rightarrow^* U) \to_1 V \\
\hat{f}\,(a \mapsto v) &= f(a, v) \\
\hat{f}\,(* \mapsto v) &= u(v)
\end{aligned}
$$

Hom characterisation:

$$(A \Rightarrow^* U) \to_1 V \cong (A \to |U \to_1 V|) \times (U \to_1 V)$$

Compact map lookup

An element of $A \Rightarrow^* U$ can be used as a function:

$$
\begin{aligned}
(a \mapsto u)(a) &= u \\
(a \mapsto u)(b) &= 0 \quad \text{for } a \neq b \\
(a \mapsto u)(*) &= 0 \\
(* \mapsto u)(a) &= u \\
(* \mapsto u)(*) &= u
\end{aligned}
$$

Let

$$x = (* \mapsto 2) + (a \mapsto 3) + (b \mapsto -2)$$

Consider the following lookups:

$$x(*) = 2 \qquad x(a) = 2 + 3 = 5 \qquad x(b) = 2 - 2 = 0 \qquad x(c) = 2$$

A *baseline* rather than a *default*.

Isomorphisms

Module (not algebra) isomorphisms:

$$
\begin{aligned}
\mathbf{F}_K[0] &\cong \mathbf{0} & \mathbf{F}_K[1] &\cong K \\
0 \Rightarrow U &\cong \mathbf{0} & 1 \Rightarrow U &\cong U \\
A \Rightarrow \mathbf{0} &\cong \mathbf{0} & A \Rightarrow K &\cong \mathbf{F}_K[A] \\
\mathbf{F}_K[A + B] &\cong \mathbf{F}_K[A] \oplus \mathbf{F}_K[B] & \mathbf{F}_K[A \times B] &\cong \mathbf{F}_K[A] \otimes \mathbf{F}_K[B] \\
(A + B) \Rightarrow U &\cong (A \Rightarrow U) \oplus (B \Rightarrow U) & (A \times B) \Rightarrow U &\cong A \Rightarrow B \Rightarrow U \\
A \Rightarrow (U \oplus V) &\cong (A \Rightarrow U) \oplus (A \Rightarrow V) & A \Rightarrow (U \otimes V) &\cong (A \Rightarrow U) \otimes V \\
A \Rightarrow U &\cong \mathbf{F}_K[A] \otimes U & A \Rightarrow^* U &\cong \mathbf{F}^*_K[A] \otimes U \\
\mathbf{F}^*_K[A] &\cong \mathbf{F}_K[A] \oplus K & A \Rightarrow^* U &\cong (A \Rightarrow U) \oplus A
\end{aligned}
$$

Are all such modules free? Well yes, but actually no.

Representation

Consider the isomorphism:

$$\varphi : \mathbf{F}_K[A \times B] \cong \mathbf{F}_K[A] \otimes \mathbf{F}_K[B]$$

Take an element $x : \mathbf{F}_K[A] \otimes \mathbf{F}_K[B]$.

$$x = (\langle a_1 \rangle + \langle a_2 \rangle + \langle a_3 \rangle) \otimes (\langle b_1 \rangle + \langle b_2 \rangle + \langle b_3 \rangle)$$

Transport it there...

$$\begin{aligned}\varphi^{-1}(x) = {} & \langle (a_1, b_1) \rangle + \langle (a_1, b_2) \rangle + \langle (a_1, b_3) \rangle + \\ & \langle (a_2, b_1) \rangle + \langle (a_2, b_2) \rangle + \langle (a_2, b_3) \rangle + \\ & \langle (a_3, b_1) \rangle + \langle (a_3, b_2) \rangle + \langle (a_3, b_3) \rangle\end{aligned}$$

...and back again

$$\begin{aligned}\varphi(\varphi^{-1}(x)) = {} & \langle a_1 \rangle \otimes \langle b_1 \rangle + \langle a_1 \rangle \otimes \langle b_2 \rangle + \langle a_1 \rangle \otimes \langle b_3 \rangle + \\ & \langle a_2 \rangle \otimes \langle b_1 \rangle + \langle a_2 \rangle \otimes \langle b_2 \rangle + \langle a_2 \rangle \otimes \langle b_3 \rangle + \\ & \langle a_3 \rangle \otimes \langle b_1 \rangle + \langle a_3 \rangle \otimes \langle b_2 \rangle + \langle a_3 \rangle \otimes \langle b_3 \rangle\end{aligned}$$

Rosetta Stone

Query paradigms:

- SQL: mixed set/multiset semantics, updates do not commute
- Relational algebra: set semantics, updates do not commute
- Algebra: semantics depend on K, updates commute

Choice of ring:

- $\mathbf{F}_{\mathbb{F}_2}[A]$ represent sets
- $\mathbf{F}_{\mathbb{N}}[A]$ represent multisets (caveat: only a semimodule)
- $\mathbf{F}_{\mathbb{Z}}[A]$ represent polysets
- $\mathbf{F}_{\mathbb{R}}[A]$ represent generalised fuzzy sets

Rosetta Stone: Selection

SQL:

```
SELECT * FROM table WHERE condition
```

Relational algebra:

$$\sigma_{\text{condition}}(\text{table}) = \{t \in \text{table} \mid \text{condition}(\text{t})\}$$

Algebra:

$$\begin{aligned} \sigma_{\text{condition}} &: \mathbf{F}_K[A] \to_1 \mathbf{F}_K[A] \\ \sigma_{\text{condition}}(\langle a \rangle) &= \langle a \rangle \quad \text{condition is true for } a \\ \sigma_{\text{condition}}(\langle a \rangle) &= 0 \quad \text{condition is false for } a \end{aligned}$$

Rosetta Stone: Projection

SQL:

```
SELECT attr FROM table
```

Relational algebra:

$$\pi_2(\text{table}) = \{y \mid (x, y) \in \text{table}\}$$

Algebra:

$$\begin{aligned}
\# : \mathbf{F}_K[A] &\to_1 K \\
\#\langle a \rangle &= 1 \\
\pi_2 : \mathbf{F}_K[A] \otimes V &\to_1 V \\
\pi_2(u \otimes v) &= \#u \cdot v
\end{aligned}$$

Note: multiplicities are preserved.

Rosetta Stone: Union

SQL:

```
table1 UNION table2
table1 UNION ALL table2
```

Relational algebra:

$$\text{table1} \cup \text{table2}$$

Algebra:

$$\text{table1} + \text{table2}$$

Note: different semantics for multiplicities.

Rosetta Stone: Cartesian product

SQL:

```
SELECT * FROM table1, table2
```

Relational algebra:

$$\mathrm{table1} \times \mathrm{table2}$$

Algebra:

$$\mathrm{table1} \otimes \mathrm{table2}$$

Note: beware of expression swell.

Rosetta Stone: Intersection

SQL:

```
SELECT * FROM table1 NATURAL JOIN table2
SELECT * FROM table1 INTERSECT SELECT * FROM table2
```

Relational algebra:

$$\text{table1} \bowtie \text{table2}$$

Algebra:

$$\text{table1} \cdot \text{table2}$$

Note: assume both tables have the same schema.

Rosetta Stone: Natural join

SQL:

```
SELECT * FROM table1
JOIN table2 ON table1.attr = table2.attr
```

Relational algebra:

$$\alpha_1(\text{table1}) \bowtie \alpha_2(\text{table2})$$

Algebra:

$$\begin{aligned}
&\alpha_1 : \mathbf{F}_K[A] \otimes \mathbf{F}_K[B] \to_1 \mathbf{F}^*_K[A] \otimes \mathbf{F}^*_K[B] \otimes \mathbf{F}^*_K[C] \\
&\alpha_1(\langle a \rangle \otimes \langle b \rangle) = \langle a \rangle \otimes \langle b \rangle \otimes 1_C \\
&\alpha_2 : \mathbf{F}_K[B] \otimes \mathbf{F}_K[C] \to_1 \mathbf{F}^*_K[A] \otimes \mathbf{F}^*_K[B] \otimes \mathbf{F}^*_K[C] \\
&\alpha_2(\langle b \rangle \otimes \langle c \rangle) = 1_A \otimes \langle b \rangle \otimes \langle c \rangle
\end{aligned}$$

$$\alpha_1(\text{table1}) \cdot \alpha_2(\text{table2})$$

Simplification

Simplification depends on the desired observation. Examples include:

- Zero or non-zero.
- Any form without multiplications.
- A list of all basis elements $\langle a \rangle$ with multiplicities.

For instance, $a \otimes b = 0$ only when $a = 0$ and $b = 0$.

Simplification

Finite maps can be simplified via isomorphisms:

$$
\begin{aligned}
&\mathbf{cp}_0 : 0 \Rightarrow U \cong \mathbf{0} &\qquad& \mathbf{cp}_+ : (A + B) \Rightarrow U \cong (A \Rightarrow U) \oplus (B \Rightarrow U) \\
&\mathbf{cp}_1 : 1 \Rightarrow U \cong U &\qquad& \mathbf{cp}_\times : (A \times B) \Rightarrow U \cong A \Rightarrow B \Rightarrow U
\end{aligned}
$$

Thus, generic tries arise naturally from algebra.

Simplification

Main challenge is reducing a product of sums.
Suppose $v_1, \ldots, v_n : A_1 \Rightarrow^* \cdots \Rightarrow^* A_m \Rightarrow^* K$.
Recurse on the type, not the expression:

$$
\begin{aligned}
& v_1 v_2 \cdots v_n \\
= \; & \left(\left(\textstyle\sum_{1 \le i \le k} a_i \mapsto v_{1,i}\right) + (* \mapsto v_{1,*})\right) v_2 \cdots v_n \\
= \; & \left(\textstyle\sum_{1 \le i \le k} (a_i \mapsto v_{1,i}) v_2 \cdots v_n\right) + (* \mapsto v_{1,*}) v_2 \cdots v_n \\
= \; & \left(\textstyle\sum_{1 \le i \le k} a_i \mapsto v_{1,i} v_2(a_i) \cdots v_n(a_i)\right) + (* \mapsto v_{1,*} v_2(*) \cdots v_n(*))
\end{aligned}
$$

This strategy is *worst-case output optimal*.

Summary

- Modules, unital algebras
- Universal constructions
- Isomorphisms, representation
- Queries
- Simplification, joins