Relational algebraic ornaments

Josh Ko & Jeremy Gibbons

Department of Computer Science
University of Oxford

Workshop on Dependently Typed Programming

Boston, MA, US, 24 September 2013

Workshop on Dependently Typed Programming

Nijmegen, the Netherlands, 27 August 2011

Internalism Externalism

Internalism

```
_++_ : Vec A m → Vec A n → Vec A (m + n)
[]        ++ ys = ys
(x :: xs) ++ ys = x :: (xs ++ ys)
```

proof structure follows program structure

How far can internalism go?

Minimum Coin Change

2/6 2/- 1/- 6d 3d 1d

shillings pence (d)

30d 24d 12d 6d 3d 1d

48d

30d 12d 6d 24d 24d

50¢ 25¢ 10¢ 5¢ 1¢

48¢

25¢ 10¢ 10¢ 1¢ 1¢ 1¢

specification

global optimisation

proved to meet

proof obligations about

“precisely typed” program

local optimisation

specification

global optimisation

? derive / reduce to

precise type

local optimisation

✔ develop interactively

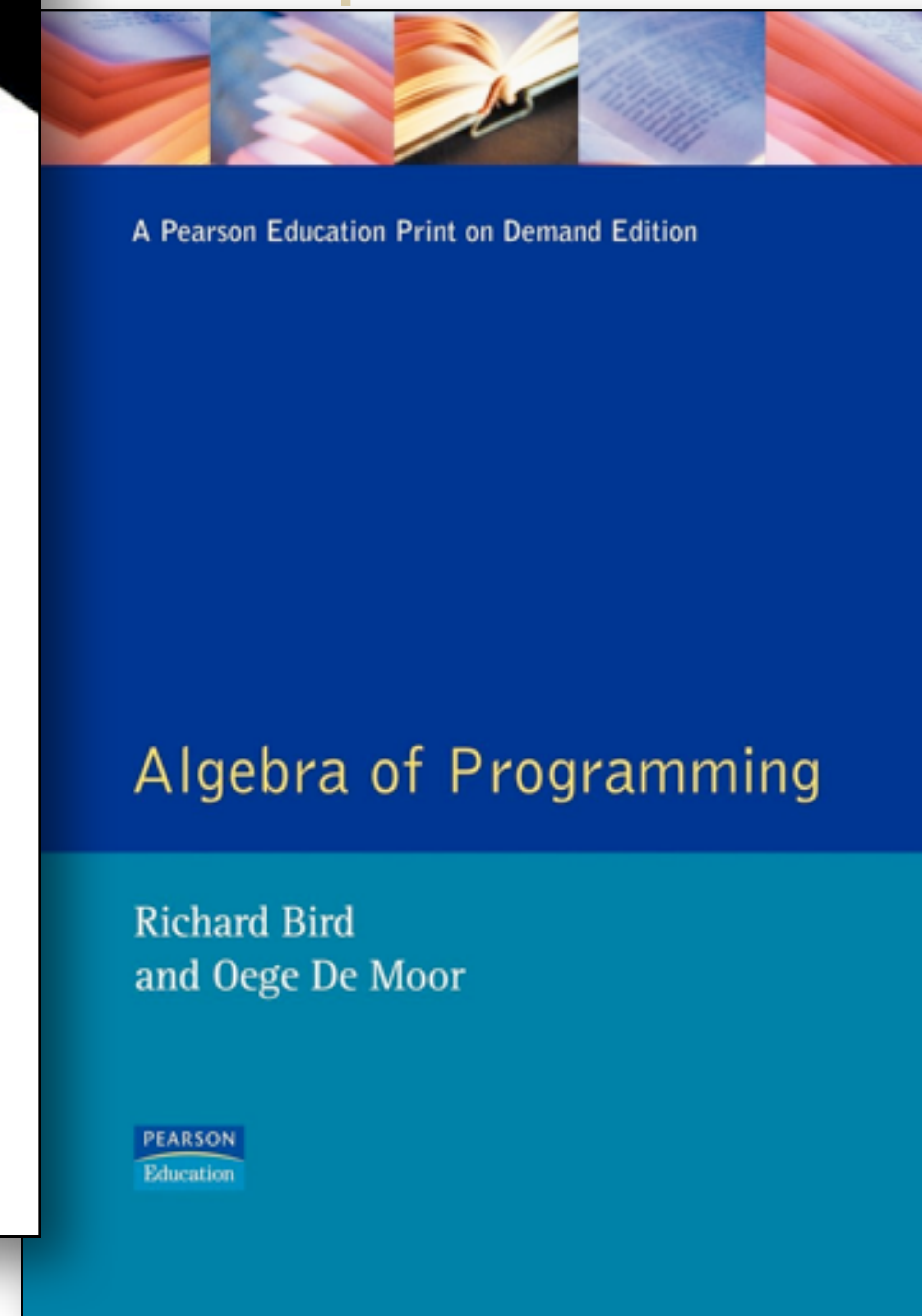
precisely typed program

need only exist; no inspection of the program tion

lobal optimisation

?

✓



relational specification

global optimisation

↓ refine by relational calculation

relational fold °

local optimisation

↓ derive by algebraic ornamentation

precise type

local optimisation

✔ develop interactively ↓

precisely typed program

Relations

potentially partial and nondeterministic mappings
(generalising functions)

$$A \to B \to \mathsf{Set}$$

predicates on (subsets of) $A \times B$

$R : A \to B \to \mathsf{Set}$ relates a to b
if $R\ a\ b : \mathsf{Set}$ is inhabited

Relations

potentially partial and nondeterministic mappings
(generalising functions)

```
A → (B → Set)
```

functions from A to subsets of B

Relations

potentially partial and nondeterministic mappings
(generalising functions)

$$A \rightsquigarrow B$$

relational programs from A to B

$\mathsf{R} : \mathsf{A} \rightsquigarrow \mathsf{B}$ nondeterministically maps a to b
if $\mathsf{R}\ a\ b : \mathsf{Set}$ is inhabited

inclusion

specification

$$
\begin{array}{ll}
& \mathit{ordered} \cdot \mathit{perm} \\
\supseteq & \quad \{\text{since } \mathit{flatten} \text{ is a function}\} \\
& \mathit{ordered} \cdot \mathit{flatten} \cdot \mathit{flatten}^\circ \cdot \mathit{perm} \\
= & \quad \{\text{claim: } \mathit{ordered} \cdot \mathit{flatten} = \mathit{flatten} \cdot \mathit{inordered} \text{ (see below)}\} \\
& \mathit{flatten} \cdot \mathit{inordered} \cdot \mathit{flatten}^\circ \cdot \mathit{perm} \\
= & \quad \{\text{converses}\} \\
& \mathit{flatten} \cdot (\mathit{perm} \cdot \mathit{flatten} \cdot \mathit{inordered})^\circ \\
\supseteq & \quad \{\text{fusion, for an appropriate definition of } \mathit{split}\} \\
& \mathit{flatten} \cdot (\!|\mathit{nil}, \mathit{split}^\circ|\!)^\circ .
\end{array}
$$

towards an executable program

Converse

$R : A \rightsquigarrow B = A \to B \to Set$

$R^{\circ} = flip\ R : B \rightsquigarrow A$

running R backwards

Relational folds

functional	relational
$f : 1 + A \times B \to B$	$S : 1 + A \times B \rightsquigarrow B$
$\mathrm{fold}\ f : \mathrm{List}\ A \to B$	$(\!\| S \|\!) : \mathrm{List}\ A \rightsquigarrow B$

$B = \mathrm{List}\ A$

$$
\begin{aligned}
S\ (\mathrm{inl}\ \_) &= \{\ [\,]\ \} \\
S\ (\mathrm{inr}\ (x\ ,\ xs)) &= \{\ xs\ ,\ x :: xs\ \}
\end{aligned}
$$

$\Rightarrow$ $(\!\| S \|\!)$ computes a subsequence of its input

Converse of relational folds

well-founded unfolds (generating inductive data)

```
sum  : List Nat ⇝ Nat

sum° : Nat ⇝ List Nat
```

breaks n into a (finite) list summing to n

Minimisation

generate all possible results of T

$$\mathtt{min}\ R \cdot \Lambda\ T$$

choose a minimum under R

T = the relation that nondeterministically breaks n into a list of coins representing n

R = the length ordering on lists

Greedy Theorem

$$\mathtt{min}\ R \cdot \Lambda\ (\!|\, S \,|\!)^{\circ} \quad S'$$
$$\supseteq\ (\!|\, \boxed{(\mathtt{min}\ Q \cdot \Lambda\ S^{\circ})^{\circ}} \,|\!)^{\circ}$$

if there exists Q such that ...

the minimum coin change problem can be solved by repeatedly choosing the largest possible denomination

$$\mathit{min}\ R \cdot \Lambda\, (\!| S |\!)^{\circ} \supseteq (\!| S' |\!)^{\circ}$$

$$p : \mathit{Nat} \to \mathit{List}\ \mathit{Coin}$$

same structure

$$(\!| S' |\!)^{\circ}\ n\ (p\ n)$$

$\Rightarrow$ { Greedy Theorem }

$$(\mathit{min}\ R \cdot \Lambda\, (\!| S |\!)^{\circ})\ n\ (p\ n)$$

Algebraic ornamentation

$S : 1 + A \times B \rightsquigarrow B$

$\mathsf{data}\ \mathrm{AlgList}\ S : B \to \mathrm{Set}$

$\mathrm{AlgList}\ S\ b \cong (xs : \mathrm{List}\ A) \times (\!| S |\!)\ xs\ b$

```
S : 1 + A × B  ⇝  B
AlgList S b  ≅  (xs : List A) × ⦇S⦈ xs b

data AlgList S : B → Set where
 nil   : {b : B} → S (inl tt) b →
         AlgList S b
 cons : {b : B} →
         (x : A) →
         {b' : B} → S (inr (x , b')) b →
         AlgList S b' → AlgList S b
```

```
AlgList S′ : Nat → Set
```

indexed by total value

the head of a nonempty list can only be the largest possible denomination

```
greedy : (n : Nat) → AlgList S′ n
```

$$\mathtt{greedy} : (\mathtt{n} : \mathtt{Nat}) \to \mathtt{AlgList}\ \mathtt{S'}\ \mathtt{n}$$

$$\boxed{\mathtt{AlgList}\ \mathtt{S'}\ \mathtt{n}} \cong (\mathtt{xs} : \boxed{\mathtt{List}\ \mathtt{Coin}}) \times (\!|\ \mathtt{S'}\ |\!)\ \mathtt{xs}\ \mathtt{n}$$

forget

$$(\!|\ \mathtt{S'}\ |\!)\ (\mathtt{forget}\ (\mathtt{greedy}\ \mathtt{n}))\ \mathtt{n}$$

$$\mathtt{p} = \mathtt{forget} \circ \mathtt{greedy} : \mathtt{Nat} \to \mathtt{List}\ \mathtt{Coin}$$

$$\mathsf{greedy} : (n : \mathsf{Nat}) \to \mathsf{AlgList}\ S'\ n$$

$$p = \mathsf{forget} \circ \mathsf{greedy} : \mathsf{Nat} \to \mathsf{List}\ \mathsf{Coin}$$

$$(\!|\, S' \,|\!)\ (p\ n)\ n$$

$\Rightarrow$ { converse }

$$(\!|\, S' \,|\!)^{\circ}\ n\ (p\ n)$$

$\Rightarrow$ { Greedy Theorem }

$$(\mathsf{min}\ R \cdot \Lambda\ (\!|\, S \,|\!)^{\circ})\ n\ (p\ n)$$

relational specification $\quad \mathrm{min}\ R \cdot \Lambda\ (\!|\ S\ |\!)^{\circ}$

$\downarrow$ relational program derivation

relational fold $^{\circ}$ $\quad (\!|\ S'\ |\!)^{\circ}$

$\downarrow$ algebraic ornamentation

internalist type $\quad$ `AlgList S′`

$\downarrow$ interactive development

internalist program $\quad$ `(n : Nat) → AlgList S′ n`

[Internalist type] derivation

Relational program derivation being one possible way