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
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)
specification

proved to meet

“precisely typed” program

proof obligations about

global optimisation

local optimisation
precisely typed program

need only exist; no inspection of the program
Richard Bird
Oege de Moor
Algebra of Programming

Algebra of Programming
Richard Bird
and Oege De Moor
relational specification

refine by
relational calculation

derive by
algebraic ornamentation

precise type

develop interactively

precisely typed program
Relations

potentially partial and nondeterministic mappings
(generalising functions)

\[ A \rightarrow B \rightarrow \text{Set} \]

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

\[ R : A \rightarrow B \rightarrow \text{Set} \] relates \( a \) to \( b \)

if \( R \ a \ b : \text{Set} \) is inhabited
Relations

potentially partial and nondeterministic mappings
(generalising functions)

\[ A \rightarrow (B \rightarrow \text{Set}) \]

functions from A to subsets of B
Relations

potentially partial and nondeterministic mappings (generalising functions)

\[ A \xrightarrow{\sim} B \]

relational programs from A to B

\[ R : A \xrightarrow{\sim} B \text{ nondeterministically maps } a \text{ to } b \]

if \( R \ a \ b : \text{ Set} \) is inhabited
The so-called 'advanced' sorting algorithms (quicksort, mergesort, heapsort, and so on) all use some form of tree as an intermediate datatype. Here we sketch the development of Hoare's quicksort (Hoare 1962), which follows the path of selection sort quite closely.

Consider the type $\text{tree } A$ defined by

$$\text{tree } A ::= \text{null} \mid \text{fork}(\text{tree } A, A, \text{tree } A).$$

The function $\text{flatten} : \text{list } A \to \text{tree } A$ is defined by

$$\text{flatten} = \text{Qnil,joinD},$$

where $\text{join}(x, a, y) = x \ast[a] \ast y$. Thus $\text{flatten}$ produces a list of the elements in a tree in left to right order.

In outline, the derivation of quicksort is

$$\text{ordered} \cdot \text{perm} \overset{\subseteq}{\supseteq} \{\text{since } \text{flatten} \text{ is a function}\} \text{ordered} \cdot \text{flatten} \cdot \text{flatten}^\circ \cdot \text{perm}$$

$$\overset{=}{{}\supseteq} \{\text{claim: } \text{ordered} \cdot \text{flatten} = \text{flatten} \cdot \text{inordered} \text{ (see below)}\} \text{flatten} \cdot \text{inordered} \cdot \text{flatten}^\circ \cdot \text{perm}$$

$$\overset{=}{{}\supseteq} \{\text{converses}\} \text{flatten} \cdot (\text{perm} \cdot \text{flatten} \cdot \text{inordered})^\circ$$

$$\overset{\subseteq}{{}\supseteq} \{\text{fusion, for an appropriate definition of } \text{split}\} \text{flatten} \cdot ([\text{nil}, \text{split}^\circ])^\circ. \text{ towards an executable program}$$
Converse

\[ R : A \leftrightarrow B = A \rightarrow B \rightarrow \text{Set} \]

\[ R^\circ = \text{flip } R : B \leftrightarrow A \]

running \( R \) backwards
Relational folds

<table>
<thead>
<tr>
<th>functional</th>
<th>relational</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f : 1 + A \times B \to B )</td>
<td>( S : 1 + A \times B \rightsquigarrow B )</td>
</tr>
<tr>
<td>( \text{fold } f : \text{List } A \to B )</td>
<td>( \langle S \rangle : \text{List } A \rightsquigarrow B )</td>
</tr>
</tbody>
</table>

\( B = \text{List } A \)

\( S (\text{inl } _) = \{ \text{[]} \} \)

\( S (\text{inr } (x, xs)) = \{ xs, x :: xs \} \)

\( \Rightarrow \langle S \rangle \) computes a subsequence of its input
Converse of relational folds

well-founded unfolds (generating inductive data)

\[
\text{sum} \ : \ \text{List Nat} \to \text{Nat}
\]

\[
\text{sum}^\circ \ : \ \text{Nat} \to \text{List Nat}
\]

breaks \(n\) into a (finite) list summing to \(n\)
Minimisation

generate all possible results of \( T \)

\[
\min R \cdot \Lambda T
\]

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

\[ \min R \cdot \Lambda (S) \quad S' \]
\[ \supseteq \bigcup (\min Q \cdot \Lambda S^\circ) \quad D^\circ \]

if there exists \( Q \) such that ...

the minimum coin change problem can be solved by repeatedly choosing the largest possible denomination.
\[ \min R \cdot \land (\downarrow S)^\circ \supseteq (\downarrow S')^\circ \]

\[ p : \text{Nat} \to \text{List Coin} \]

\[ (\downarrow S')^\circ \ n \ (p \ n) \]

\[ \Rightarrow \quad \{ \text{Greedy Theorem} \} \]

\[ (\min R \cdot \land (\downarrow S)^\circ) \ n \ (p \ n) \]

same structure
Algebraic ornamentation

\[ S : 1 + A \times B \leadsto B \]

data AlgList S : B \rightarrow Set

AlgList S b \cong (xs : List A) \times (\downarrow S) \; xs \; b
\[ S : 1 + A \times B \rightsquigarrow B \]
\[ \text{AlgList } S \ b \cong (xs : \text{List } A) \times (\{S\} \ xs \ b) \]

\text{data AlgList } S : B \to \text{Set where}

\[ \text{nil} : \{b : B\} \to S \ (\text{inl } \text{tt}) \ b \to \text{AlgList } S \ b \]

\[ \text{cons} : \{b : B\} \to (x : A) \to \{b' : B\} \to S \ (\text{inr } (x, b')) \ b \to \text{AlgList } S \ b' \to \text{AlgList } S \ b \]
AlgList \( S' \) : Nat \( \rightarrow \) Set

indexed by total value

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

greedy : (n : Nat) \( \rightarrow \) AlgList \( S' n \)
greedy : (n : Nat) → AlgList S' n

\[ \text{AlgList } S' n \cong (\text{List Coin}) \times \langle S' \rangle \text{ xs n} \]

\[ \text{forget} \]

\[ \langle S' \rangle (\text{forget (greedy n)}) n \]

\[ p = \text{forget} \circ \text{greedy} : \text{Nat} \rightarrow \text{List Coin} \]
\text{greedy} : (n : \text{Nat}) \to \text{AlgList } S' \ n

\quad p = \text{forget} \circ \text{greedy} : \text{Nat} \to \text{List Coin}

\quad \{ \text{converse} \}

\quad \{ \text{Greedy Theorem} \}

\quad (\min R \cdot \forall \ y \quad \{ S' \}^o \ n \ (p \ n))
relational specification

min \ R \cdot \land (\forall S)^°

relational program derivation

relational fold °

(\forall S')^°

algebraic ornamentation

internalist type

AlgList S'

interactive development

internalist program

(n : Nat) → AlgList S'n
Internalist type derivation

Relational program derivation being one possible way