Finding Functors in Relational ADTs

Generic Programming Workshop
Oxford, June 2004

Eerke Boiten
University of Kent
Canterbury, UK

Good Morning

Generic implementations are great but

. . .

What generic specifications do they implement?

How do they relate to “real” programming?

(Underspecification,
hidden state,
reference semantics)

Relational ADTs

Local theory.

ADT $(\mathit{State}, \mathit{Init}, \{\mathit{Op}_i\}_{i \in I})$

Observations: *not* (necessarily) *State*

Z refinement, Object-Z classes

Functors on ADTs

Goal: *specify* data-type generic programs

Strategy: identify structure, generalise.
(Parametric polymorphism generalising to polytypism)

Inspiration: FP world, category theory

Pragmatic "functor": mapping with coherent effect on data and operations.

Formal Methods Folklore

- product of ADTs is monotonic wrt refinement;

- so is “free promotion”.

So which product functor?

Version 1: synchronous product.

$$\begin{aligned}
A &= (AS, AInit, \{AOp_i\}_{i \in I}) \\
B &= (BS, BInit, \{BOp_i\}_{i \in I}) \\
A \times B &= (AS \times BS, \\
&\qquad AInit \| BInit, \\
&\qquad \{AOp_i \| BOp_i\}_{i \in I})
\end{aligned}$$

Version 2: asynchronous product.

$$\begin{aligned}
A &= (AS, AInit, \{AOp_i\}_{i \in I}) \\
B &= (BS, BInit, \{BOp_j\}_{j \in J}) \\
A \times B &= (AS \times BS, \\
&\qquad AInit \| BInit, \\
&\qquad \{AOp_i \| id_{BS}\}_{i \in I} \uplus \{id_{AS} \| BOp_j\}_{j \in J})
\end{aligned}$$

Version 3: mixed product.

$$
\begin{aligned}
&A = (AS, AInit, \{AOp_i\}_{i\in I})\\
&B = (BS, BInit, \{BOp_j\}_{j\in J})\\
&A \times B = (AS \times BS, AInit \| BInit,\\
&\qquad\qquad \{AOp_i \| BOp_j\}_{i\in I, j\in J})
\end{aligned}
$$

Version 4: synchronous product, fixed state.

$$
\begin{aligned}
&A = (S, AInit, \{AOp_i\}_{i\in I})\\
&B = (S, BInit, \{BOp_i\}_{i\in I})\\
&A \times B = (S, AInit \wedge BInit,\\
&\qquad\qquad \{AOp_i \wedge BOp_i\}_{i\in I})
\end{aligned}
$$

Version 5: asynchronous product, fixed state.

$$
\begin{aligned}
&A = (S, AInit, \{AOp_i\}_{i\in I})\\
&B = (S, BInit, \{BOp_i\}_{i\in I})\\
&A \times B = (S, AInit \wedge BInit,\\
&\qquad\qquad \{AOp_i\}_{i\in I} \uplus \{BOp_j\}_{j\in J})
\end{aligned}
$$

Total?

V 2, V 3: yes

V 1: on conformal ADTs

V 4: on conformal ADTs with identical state

V 5: on ADTs with identical state

Associative?

V 1, V 4: yes. V 2: as $\times$, V 3, V 5: as $\uplus$

Units of product?

V 1: yes, $(\mathbb{1}, G \times \mathbb{1}, \{id_{\mathbb{1}}\}_{i \in I})$

V 2, V 5: yes, $I = \varnothing$.

V 3: as V 1.

v 4: yes, $(S, G \times S, \{S \times S\}_{i \in I})$

Monotonic?

With respect to weaken pre, strengthen post

V 1, V 2, V 3, V 5: yes

V 4: no. Consider $x' \in \{1, 2\}$ and $x' \in \{2, 3\}$, refined to $x' = 1$ and $x' = 3$.

Sums

Starting from $AS \uplus BS$:
"separated at birth".

Maybe

Adding $\perp$.

Monotonicity: relates to total correctness refinement theories.

Free Promotion

General informal statement:

(naturality of polymorphism)

Most common case:

From

$$(S, \mathit{Init}, \{\mathit{Op}_i\}_{i \in I})$$

to

$$(J \to S, \|_{j \in J}\mathit{Init}, \mathrm{promops})$$

where promops is
$\{$ "do Op_i on component j, skip on others " $\}_{i \in I, j \in J}$

is repeated asynchronous product so $\surd$.

Later

More “functors”

The category of operations

Recursion

Reference semantics: more monotonicity