Compound Monads in Specification Languages

Jeremy Dawson

Logic and Computation Program, NICTA [1]

Automated Reasoning Group,
Australian National University, Canberra, ACT 0200, Australia
http://users.rsise.anu.edu.au/~jeremy/

September 4, 2007

[1] National ICT Australia is funded by the Australian Government's Dept of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Centre of Excellence program.

Outline

Outline

Introduction

Several sorts of refinement suggested by Dunne.

- General Correctness
- Total Correctness
- Chorus Angelorum

Each is based, implicitly or explicitly, on a notion of what a computation is, an underlying "model of computation"

Each underlying "model of computation" is based on a monad

Each of these monads is, or is somewhat like, a compound monad

Outline

The General Correctness Operational Model

Want to distinguish computations which (on a given initial state)

- fail to terminate
- terminate in final state s
- non-deterministically, either of the above

Neither *wlp* / partial correctness
nor *wp* / total correctness does this.

General correctness refinement (Dunne):

$$A \sqsubseteq B \equiv wp(A, Q) \Rightarrow wp(B, Q) \wedge wlp(A, Q) \Rightarrow wlp(B, Q)$$

The General Correctness Operational Model

Type of Computations

A computation (on given state) produces a set of outcomes.
An outcome is either

- NonTerm, indicating non-termination, or
- Term s, indicating termination in the state s.

In Isabelle: `datatype` σ `TorN = NonTerm | Term` σ
For a non-deterministic computation (from given initial state), result is a set of outcomes.

type $outcome$ = TorN $state$

type of computations is $state \to set$ TorN $state$

The Total Correctness Operational Model

Related to semantics of the B-method,
only interested in total correctness (weakest preconditions).

A computation which may fail to terminate fails every post-condition.

Such computation is refinement-equivalent to a computation which does fail to terminate.

Type of results is either

- NonTerm, indicating possible non-termination, or
- Term S, indicating termination in a state $s \in S$.

type of result *tcres* ("total correctness result") $=$ TorN *set state*

type of computations is *state* $\rightarrow$ TorN *set state*

weakest precondition function (hence refinement):

$$[C]\ Q\ s = \exists S.\ (\forall x \in S.\ Q\ x) \wedge C\ s = \texttt{Term}\ S$$

The Chorus Angelorum Operational Model

Ordinarily, non-determinism is demonic choice
(all possible results must satisfy post-condition $\equiv$
the result chosen by a demon satisfies post-condition)

Want to model angelic and demonic non-determinism

Computation returns a set of sets $\mathcal{A}$ of states:

- angel chooses set $A \in \mathcal{A}$
- demon chooses state $a \in A$

weakest precondition function (hence refinement):

$$[C]\ Q\ s = \exists U \in C\ s.\ (\forall u \in U.\ Q\ u)$$

If $A \in \mathcal{A}$, $A' \supseteq A$, to include A' in $\mathcal{A}$, or not, makes no difference:
consider only $\mathcal{A}$ up-closed: if $A' \supseteq A$ and $A \in \mathcal{A}$ then $A' \in \mathcal{A}$.

Confirming the Models

In each case, to confirm model is appropriate,

- we show two computations refinement-equivalent iff they are the same function (of type used in model)
- we define operations operationally, and prove these definitions correspond to Dunne's definitions (which use weakest preconditions)

(Caveat: we ignore "frames").

Note: all proofs in the theorem prover Isabelle/HOL

Outline

Monads

Long known in category theory.

Define unit and extension functions, satisfying rules

$$\begin{aligned} \mathit{unit} &: \alpha \to M\alpha \\ \mathit{ext} &: (\alpha \to M\beta) \to (M\alpha \to M\beta) \end{aligned}$$

$$\begin{aligned} \mathit{ext}\ f \circ \mathit{unit} &= f \\ \mathit{ext}\ \mathit{unit} &= \mathit{id} \\ \mathit{ext}\ (\mathit{ext}\ g \circ f) &= \mathit{ext}\ g \circ \mathit{ext}\ f \end{aligned}$$

or functions *unit*, *map* and *join* (7 axioms for these)

Can represent the structure of a computation (Moggi)

Monads — the Kleisli category

ext B models the action of B on result of previous computation

Define $B \odot A = ext\ B \circ A$: sequencing computations B and A.

$$f \odot unit = f \tag{1}$$
$$unit \odot f = f \tag{2}$$
$$h \odot (g \odot f) = (h \odot g) \odot f \tag{3}$$

Monads — the Kleisli category

ext B models the action of B on result of previous computation

Define $B \odot A = \mathit{ext}\ B \circ A$: sequencing computations B and A.

$$f \odot \mathit{unit} = f \tag{1}$$
$$\mathit{unit} \odot f = f \tag{2}$$
$$h \odot (g \odot f) = (h \odot g) \odot f \tag{3}$$

Properties (1) to (3) show that we have a category:

- objects are types
- arrow from α to β is function $\alpha \to M\beta$,
- the identity arrow for object α is the function $\mathit{unit} : \alpha \to M\alpha$
- composition is given by $\odot$.

Called the Kleisli category of M, $\mathcal{K}(M)$.

Monads — Examples

The non-termination monad: a computation either terminates in a new state, or fails to terminate.

$$\mathit{unit\_nt}\ s = \mathtt{Term}\ s$$

$$\mathit{map\_nt}\ f\ \mathtt{NonTerm} = \mathtt{NonTerm} \quad \mathit{map\_nt}\ f\ (\mathtt{Term}\ s) = \mathtt{Term}\ (f\ s)$$

$$\mathit{ext\_nt}\ f\ \mathtt{NonTerm} = \mathtt{NonTerm} \quad \mathit{ext\_nt}\ f\ (\mathtt{Term}\ s) = f\ s$$

Monads — Examples

The non-termination monad: a computation either terminates in a new state, or fails to terminate.

$$unit\_nt\ s = \mathtt{Term}\ s$$
$$map\_nt\ f\ \mathtt{NonTerm} = \mathtt{NonTerm} \qquad map\_nt\ f\ (\mathtt{Term}\ s) = \mathtt{Term}\ (f\ s)$$
$$ext\_nt\ f\ \mathtt{NonTerm} = \mathtt{NonTerm} \qquad ext\_nt\ f\ (\mathtt{Term}\ s) = f\ s$$

The set monad: models non-deterministic (but necessarily terminating) computations.

$$unit\_s\ s = \{s\} \qquad join\_s\ \mathcal{A} = \bigcup \mathcal{A}$$
$$map\_s\ f\ S = \{f\ s \mid s \in S\} \qquad ext\_s\ f\ S = \bigcup_{s \in S} f\ s$$

Compound Monads

Let M and N, each with unit and extension functions, be monads.

Then is $MN\alpha$ a monad? Need $unit_{MN} : \alpha \rightarrow MN\alpha$ and ext_{MN}

ext_{MN} "extends" a function f from domain α to $MN\alpha$.

pext, "partial extension", does part of this

$$ext_{MN} : (\alpha \longrightarrow MN\beta) \longrightarrow (MN\alpha \longrightarrow MN\beta)$$
$$pext : (\alpha \longrightarrow MN\beta) \longrightarrow (N\alpha \longrightarrow MN\beta)$$

Compound Monads

Let M and N, each with unit and extension functions, be monads.

Then is $MN\alpha$ a monad? Need $unit_{MN} : \alpha \to MN\alpha$ and ext_{MN}

ext_{MN} "extends" a function f from domain α to $MN\alpha$.

pext, "partial extension", does part of this

$$ext_{MN} : (\alpha \to MN\beta) \to (MN\alpha \to MN\beta)$$
$$pext : (\alpha \to MN\beta) \to (N\alpha \to MN\beta)$$

Definitions using *pext* for a compound monad

$$ext_{MN}\ g = ext_M\ (pext\ g)$$
$$unit_{MN} = unit_M \circ unit_N$$

Compound Monads — rules for *pext*

pext also must satisfy three rules

$$pext\ f \circ unit_N = f$$
$$pext\ unit_{MN} = unit_M$$
$$pext\ (ext_{MN}\ g \circ f) = ext_{MN}\ g \circ pext\ f$$

$unit_{MN}$ and *pext* are the unit and extension functions of a monad *in* the category $\mathcal{K}(M)$, whose Kleisli category is also $\mathcal{K}(MN)$.

Compound Monads — Distributive Law

Jones & Duponcheel: two conditions, J(1) and J(2), which compound monads may satisfy.

Assuming $unit_{MN} = unit_M \circ unit_N$ and $map_{MN} = map_M \circ map_N$, compound monads arise from a function *pext* iff J(1) holds

Compound monads satisfying J(1) and J(2) are those arising from a distributive law $swap : NM\alpha \to MN\alpha$
A distributive law satisfies S(1) to S(4) of Jones & Duponcheel

$$swap = pext\ (map_M\ unit_N)$$

The General Correctness Compound Monad

Want *set* $\mathtt{TorN}\ \alpha$ is a monad;
in fact, for any monad M, $M\ \mathtt{TorN}\ \alpha$ is a monad

$$pext : (\alpha \rightarrow M\ \mathtt{TorN}\ \beta) \rightarrow (\mathtt{TorN}\ \alpha \rightarrow M\ \mathtt{TorN}\ \beta)$$

$$pext\ f\ (\mathtt{Term}\ a) = f\ a$$
$$pext\ f\ \mathtt{NonTerm} = unit_M\ \mathtt{NonTerm}$$

Proof of *pext* axioms easy.

Arises from a distributive law: $swap = pext\ (map_M\ unit_N)$, so

$$swap\_gc : \mathtt{TorN}\ set\ \alpha \rightarrow set\ \mathtt{TorN}\ \alpha$$

$$swap\_gc\ \mathtt{NonTerm} = \{\mathtt{NonTerm}\}$$
$$swap\_gc\ (\mathtt{Term}\ S) = \{\mathtt{Term}\ s \mid s \in S\}$$

The Total Correctness Compound Monad

Recall $tcres = \texttt{TorN}\ set\ state$.

$$pext\_tc : (state \to tcres) \to set\ state \to tcres$$

defined using

$$prod\_tc : set\ tcres \to tcres$$

$$prod\_tc\ S = \texttt{NonTerm} \qquad \text{if } \texttt{NonTerm} \in S$$
$$prod\_tc\ \{\texttt{Term}\ s \mid s \in S\} = \texttt{Term}\ (\bigcup S)$$

The Total Correctness Compound Monad

A Distributive Law and Monad Morphism

Total Correctness monad also arises from a distributive law:

$$\begin{aligned} \mathit{swap\_tc} : \mathit{set}\ \mathtt{TorN}\ \sigma &\longrightarrow \mathtt{TorN}\ \mathit{set}\ \sigma \\ \mathit{swap\_tc}\ S &= \mathtt{NonTerm} \qquad \text{if } \mathtt{NonTerm} \in S \\ \mathit{swap\_tc}\ \{\mathtt{Term}\ s \mid s \in S\} &= \mathtt{Term}\ S \end{aligned}$$

Relating the General and Total Correctness monads

$swap\_tc : set\ \texttt{TorN}\ \sigma \rightarrow \texttt{TorN}\ set\ \sigma$ is also a monad morphism from the general correctness monad to the total correctness monad.

$$unit\_tc\ a = swap\_tc\ (unit\_gc\ a)$$
$$ext\_tc\ (swap\_tc \circ f)\ (swap\_tc\ x) = swap\_tc\ (ext\_gc\ f\ \ x)$$

Since it is *surjective*, could use monad axioms for general correctness monad to prove axioms for total correctness monad.

The Chorus Angelorum Monad

up-closure, swapping angel and demon

Result $\mathcal{A}$: *set set state* (up-closed):
angel chooses $A \in \mathcal{A}$, demon chooses $a \in A$.

Alternative model: demon chooses first, then angel.

swap_uc turns angel-chooses-first result into demon-chooses-first.

up_cl: the *up-closure* of a set of sets.

$$swap\_uc\ \mathcal{A} = \{B \mid \forall A \in \mathcal{A}.\ B \cap A \neq \{\}\}$$
$$up\_cl\ \mathcal{A} = \{A' \mid \exists A \in \mathcal{A}.\ A \subseteq A'\}$$

The Chorus Angelorum Monad

up-closure, swapping angel and demon

Result $\mathcal{A}$: *set set state* (up-closed):
angel chooses $A \in \mathcal{A}$, demon chooses $a \in A$.

Alternative model: demon chooses first, then angel.

swap_uc turns angel-chooses-first result into demon-chooses-first.

up_cl: the *up-closure* of a set of sets.

$$\mathit{swap\_uc}\ \mathcal{A} = \{B \mid \forall A \in \mathcal{A}.\ B \cap A \neq \{\}\}$$
$$\mathit{up\_cl}\ \mathcal{A} = \{A' \mid \exists A \in \mathcal{A}.\ A \subseteq A'\}$$

$$\mathit{up\_cl}\ (\mathit{up\_cl}\ \mathcal{A}) = \mathit{up\_cl}\ \mathcal{A} \qquad \mathit{swap\_uc}\ (\mathit{swap\_uc}\ \mathcal{A}) = \mathit{up\_cl}\ \mathcal{A}$$
$$\mathit{swap\_uc}\ (\mathit{up\_cl}\ \mathcal{A}) = \mathit{swap\_uc}\ \mathcal{A} \qquad \mathit{up\_cl}\ (\mathit{swap\_uc}\ \mathcal{A}) = \mathit{swap\_uc}\ \mathcal{A}$$

So work on equivalence classes of sets of sets of states
$\mathcal{A} \equiv \mathcal{A}'$ iff $\mathit{up\_cl}\ \mathcal{A} = \mathit{up\_cl}\ \mathcal{A}'$
each equivalence class has exactly one up-closed member.

The Chorus Angelorum Monad

proofs of monad rules

- try to prove S(1) to S(4) (to show distributive law):
  cannot, but we can prove them modulo up-closure, eg

$$\begin{aligned} \textit{swap\_uc}\ A &= \textit{up\_cl}\ (\textit{map\_s}\ \textit{unit\_s}\ A) && \mathrm{S}(2)' \\ \textit{swap\_uc}\ (\textit{map\_s}\ \textit{unit\_s}\ A) &= \textit{up\_cl}\ A && \mathrm{S}(3)' \end{aligned}$$

- proofs of the monad axioms for *set set* α
  (again, some equalities only modulo up-closure)
  difficult, but imitated usual proofs from S(1) to S(4)
- defined type *ucss* α : *up-closed* sets of sets
  (ie, a representative of each equivalence class)
- defined the monad functions for the *ucss* α type
- translated results about *set set* α to *ucss* α: it is a monad!

The Chorus Angelorum Monad

Link to Continuation Monad

First, recall functions used by Jones & Duponcheel

$$\mathit{join} : M\ N\ M\ N\ \alpha \to M\ N\ \alpha \qquad \mathit{prod} : N\ M\ N\ \alpha \to M\ N\ \alpha$$
$$\mathit{dorp} : M\ N\ M\ \alpha \to M\ N\ \alpha \qquad \mathit{swap} : N\ M\ \alpha \to M\ N\ \alpha$$

Think of M (N) as a set from which angel (demon) chooses.

"evaluation function" $\mathit{eval\_uc} : \mathit{set}\ \mathit{set}\ \alpha \to (\alpha \to \mathit{bool}) \to \mathit{bool}$,

$\mathit{eval\_uc}\ \mathcal{A}\ P$ tells whether the post-condition P is satisfied when angel and demon have made their choices from $\mathcal{A}$.

$\mathit{eval\_uc}\ \mathcal{B}\ P \equiv \exists B \in \mathcal{B}.\ \forall b \in B.\ P\ b.$

$(\alpha \to \mathit{bool}) \to \mathit{bool}$ is type of continuation monad $K\ \alpha$

Ball and Bex: $\mathit{set}\ \alpha \to (\alpha \to \mathit{bool}) \to \mathit{bool}$, ie : $\mathit{set}\ \alpha \to K\ \alpha$ express quantification over a given set: $\mathit{Ball}\ S\ P \equiv \forall s \in S.\ P\ s$

The Chorus Angelorum Monad

Link to Continuation Monad – ctd

$$eval\_uc = Ball \odot_K Bex$$
$$eval\_uc \circ swap\_uc = Bex \odot_K Ball$$

Using obvious isomorphism $K\ \alpha \to set\ set\ \alpha$, called K_to_SS:

$$\begin{aligned}
join_uc &= K_to_SS \circ (Ball \odot_K Bex \odot_K Ball \odot_K Bex) \\
dorp_uc &= K_to_SS \circ (Bex \odot_K Ball \odot_K Bex) \\
prod_uc &= K_to_SS \circ (Ball \odot_K Bex \odot_K Ball) \\
swap_uc &= K_to_SS \circ (Bex \odot_K Ball) \\
ext_uc\ f &= K_to_SS \circ (Ball \odot_K (Bex \circ f) \odot_K Ball \odot_K Bex) \\
pext_uc\ f &= K_to_SS \circ (Ball \odot_K (Bex \circ f) \odot_K Ball)
\end{aligned}$$

Angelic and Demonic Choice

We defined these as follows (simplified by

- omitting conversion between the *set set* α and *ucss* α types
- assuming up-closed families of sets)

$$\mathrm{dem}\ \mathcal{B}\ s = \bigcap \{B\ s \mid B \in \mathcal{B}\}$$
$$\mathrm{ang}\ \mathcal{B}\ s = \bigcup \{B\ s \mid B \in \mathcal{B}\}$$

giving these results (which would normally be the definitions)

$$[\mathrm{dem}\ \mathcal{B}]\ Q\ s = \forall B \in \mathcal{B}.\ [B]\ Q\ s$$
$$[\mathrm{ang}\ \mathcal{B}]\ Q\ s = \exists B \in \mathcal{B}.\ [B]\ Q\ s$$