Relating Backtracking Monads

Dale Vaillancourt and Mitchell Wand

Northeastern University
Backtracking computation

- Simply-typed PCF with backtracking.
- Might write something like:

\[
\text{nats-from } n = n \lor \text{nats-from}(n + 1) \\
\text{nats} = \text{nats-from 0}
\]

- How might we model such a language?
Two models

- Represent a backtracking computation as a stream of answers.

\[ \text{nats} = \langle 0, 1, 2, \ldots \rangle \]
\[ \text{fail} = \langle \rangle \]
Two models

- Represent a backtracking computation as a stream of answers.
  \[
  \text{nats} = \langle 0, 1, 2, \cdots \rangle \\
  \text{fail} = \langle \rangle
  \]

- Represent a backtracking computation as a procedure that consumes two arguments.
  \[
  \text{nats} = \lambda \kappa. \lambda \phi. (\kappa \ 0 \ (\text{nats-from} \ 1 \ \kappa \ \phi)) \\
  \text{fail} = \lambda \kappa. \lambda \phi. \phi
  \]
Two models

- In “The Design of a Pretty-printing Library”, Hughes defines a backtracking monad.
- Both of these models are, indeed, backtracking monads.
Two models

- So, following Moggi:

   - PCF / Backtracking
   - translate
   - Metalanguage
   - Stream
   - 2-Continuation
   - Model 1
   - Model 2
Two models

- So, following Moggi:
Past attempts

- Fold/unfold derivations: Sound, but not adequate. [Hughes, Hinze]
- Final algebra semantics: Ditto. [Wand]
- Scott representation: Types do not work.
- Church representation: Ditto.
Represent a stream $\langle v_1, v_2, \ldots, v_n \rangle$ by
$\lambda \kappa. (\kappa v_1) \circ (\kappa v_2) \circ \cdots \circ (\kappa v_n)$.

Cannot handle higher-order quantities.

Infinite streams?
Logical Relations

- Technique for relating two meanings of a program.

- Even works when program manipulates higher-order quantities.
Logical Relations

- A family of relations, indexed by type.
- At scalar type, choose a “well-behaved” relation (like =).
- Functions:
Logical Relations

- We extend the logical relation to relate computations: If \((w, v) \in R_A\) then
  \((\langle w \rangle, \lambda \kappa. \lambda \varphi. (\kappa v \varphi)) \in R_{\text{Comp}(A)}\).
Logical Relations

- We extend the logical relation to relate computations: If \((w, v) \in R_A\) then \((\langle w \rangle, \lambda \kappa. \lambda \phi. (\kappa v \phi)) \in R_{\text{Comp}(A)}\).

- And now, by the fundamental theorem of logical relations, the meaning of a metalanguage term in the stream world is related to its meaning in the 2-continuation world.
In particular, if, in the metalanguage, \( \Gamma \vdash e : \text{Comp}(b) \) then

\[ e^S = e^K \text{ cons nil} \]

Done!
Handling infinite streams

- Simply add a fixed-point operator to the metalanguage!
- We close our logical relation is under $\omega$-chains of related quantities, so the computations representing infinite streams are related.
Summing up

- We have related two models of backtracking computation.
- We accommodate higher-order quantities and infinite streams.
- We did not appeal to category theory!