Refinement Calculus
(and Martin-Löf type theory)

Peter Hancock
peter@premise.demon.co.uk
http://www.dcs.ed.ac.uk/~pgh

March 2002
Summary

Some (unexpected) connections between the refinement calculus (Back, Morris, Morgan, von-Wright, ...) and Petersson-Synek trees in Martin-Löf type theory.

Suggests a normal form for specifications of certain kinds of interactive program (angelic “user-side” programs and demonic “system-side” programs), expressible with dependent types. A proof that a specification is satisfiable is in principle executable as a program of the appropriate kind.

Many questions raised. I’d like your opinion.

Collaborators
Anton Setzer (Swansea)
– input-output monads and coalgebras
Pierre Hyvernat (Lyons/Chalmers)
– implementation
My own interest
– specifications using dependent types.
How to deal with interaction
(action/reaction)
in type theory?

What kind of proof is it that

- runs an internet server to book plane-flights and hotel rooms?
- prevents the brakes on a bus from locking in a skid?
- flies a cruise missile?

What proposition does it prove, and how is this connected with a specification of the desired behaviour?
Context: strength of a programming logic

**batch**  \[ \text{Input} \xrightarrow{f} \text{Output} \]

Input/output is available *in its entirety* when execution starts/terminates.

Strength: the set of batch programs that can be proved to terminate.
(Termination strength, provably total recursive functions)

**transaction** Input is consumed and output is produced *piece by piece*. Eventual termination.

Strength: the set of transaction programs that can be proved to terminate (*i.e.* with output available *in its entirety*) given a sufficiently long sequence of inputs.
(Continuity, well-foundedness, . . . .)
A model of imperative interfaces

Two levels of choice:

\[\text{angel, client} \quad \text{demon, server}\]

stimulus, response

command, response

action, reaction

move, counter-move

call, return

\[C, R\]

\(S: \text{set,}\) \hspace{1cm} (\text{States})

\(C(x): \text{set } (x \in S),\) \hspace{1cm} (\text{Angel})

\(R(x, y): \text{set } (x \in S, y \in C(x)),\) \hspace{1cm} (\text{Demon})

\(n(x, y, z): S (x \in S, y \in C(x), z \in R(x, y))\) \hspace{1cm} (\text{next})

For each \(s \in S\) a family of families of outcomes:

\[\{ \{ n(s, c, r) \mid r \in R(s, c) \} \mid c \in C(s) \}\]
Interaction structure

\[ \Phi : S \rightarrow \mathcal{F}(\mathcal{F}(S')) \]

\( s \in S \) a state (position)
\( c \in C(s) \) an input (action) in state \( s \in S \)
\( r \in R(s,c) \) an output (reaction)
in response to \( c \in C(s) \)
\( s[c/r] : S' \) the new state after interaction \( c/r \).
\( = n(s,c,r) \) notation

Interaction system

\((S : \text{set, } \Phi : S \rightarrow \mathcal{F}(\mathcal{F}(S')), s_0 \in S)\)
Notions of powerset

\[ \mathbb{P}(S) = \text{set}^S \]
\[ \mathbb{F}(S) = (\exists T : \text{set}) S^T \]
\[ P = \{ s \in S \mid P(s) \} \]
\[ \langle T, s \rangle = \{ s(t) \mid t \in T \} \]

Predicates to families:
‘\( \Sigma \)-types’ and (first) projection.
\[ P \mapsto \{ \pi_0(z) \mid z \in (\exists s \in S) P(s) \} \]

Families to predicates: singleton predicates*
\[ \{ s \} = \{ s' \in S \mid s' =_S s \} \]
\[ \{ s(t) \mid t \in T \} \mapsto \{ s' \in S \mid s' =_S s(t) \} \]

(*: Singleton predicates are evil.)
The programmer’s firmament

Function
\[ A \to B \]

Relation
\[ A \to \mathcal{P}(B) \]

Transition Structure
\[ A \to \mathcal{F}(B) \]

Predicate Transformer
\[ A \to \mathcal{P}(\mathcal{P}(B)) \]
\[ \cong \quad \mathcal{P}(B) \to \mathcal{P}(A) \] (flip)

Interaction Structure
\[ A \to \mathcal{F}(\mathcal{F}(B)) \]
Interaction structures as predicate transformers

Given \( \Phi : S \rightarrow \mathbb{F}(\mathbb{F}(S')) \),
define \( \Phi^\circ : \mathcal{P}(S') \rightarrow \mathcal{P}(S) \).

\[
\Phi^\circ(X) = \{ s \in S \mid (\exists c \in C_\Phi(s)) \\
(\forall r \in R_\Phi(s, c)) \\
X(n_\Phi(s, c, r)) \}
\]

"DNF" (Disjunctive Normal Form).

Aside: conjunctive normal form doesn't work.
The refinement calculus

- predicate transformers (business end):
  \[ \Phi, \Psi ::= \text{abort, magic}, \]
  \[ \Phi \sqcup \Psi, \Phi \sqcap \Psi, \]
  \[ \sqcup_i \Phi_i, \sqcap_i \Phi_i, \]
  \[ \langle \phi \rangle, \llbracket \phi \rrbracket \]
  \[ \Phi \sqsubseteq \Psi = \forall X. \Phi(X) \subseteq \Psi(X) \]

- relations: \( R ::= \ldots \)
  \[ \phi ::= \ldots \]

- predicates: \( P, Q ::= \ldots \)

- state transformers: \( f, g ::= \ldots \)

- ergonomics.
Semantic hijack
e.g. sequential composition

\[
C_{\Phi;\Psi}(s) \\
= (\exists c \in C_{\Phi}(s))(\forall r \in R_{\Phi}(s,c))C_{\Psi}(n_{\Phi}(s,c,r)) \\
= \Phi^\circ(C_{\Psi},s) \\
R_{\Phi;\Psi}(s,\langle c, f \rangle) \\
= (\exists r \in R_{\Phi}(s,c))R_{\Psi}(n_{\Phi}(s,c,r), f(r)) \\
n_{\Phi;\Psi}(s,\langle c, f \rangle,\langle r, r' \rangle) \\
= n_{\Psi}(n_{\Phi}(s,c,r), f(r), r')
\]

Have to check \((\Phi;\Psi)^\circ = \Phi^\circ \cdot \Psi^\circ\).
Proof: axiom of choice, amalgamation of same-sex quantifiers.
Two forms of recursion

\[ \Phi^* = \mu \Psi. \text{skip} \sqcup (\Phi ; \Psi) \]
\[ \Phi^\infty = \nu \Psi. \text{skip} \sqcap (\Phi ; \Psi) \]

\( \Phi^* \): inductively defined (Petersson and Synek). We have to terminate eventually, but we can choose when. Formally a closure operator.

\( \Phi^\infty \): coinductively defined. They can choose to terminate at any point, or not at all. Formally an interior operator.

\( Y = \Phi^\infty(X) \) is the weakest invariant of \( \Phi \) (i.e. post fixed point, satisfying \( Y \subseteq \Phi(Y) \)) that implies \( X \).

\( Y = \Phi^*(X) \) is also an invariant (Lambek). It is the strongest invariant of \( \Phi \) that is implied by \( X \) holding eventually.

\( \Phi^\sim = \Phi \) with the angel and the demon swapped.
What theorem is proved by a $c/r$-program?

(First approximation.)

**a client** (terminating) \[ A \subseteq \Phi^*(B) \]
Requires $A$ initially, guarantees $B$ finally (provided there is a ‘finally’).

\[
\begin{array}{c}
\text{‘overlaps’} \\
\downarrow \\
\text{a server} \ (\text{perpetual}) \ \ \ \ \ \ \ \ A \not\subseteq \Psi^\infty(B) \\
\text{Guarantees } A \text{ initially, and } B \text{ perpetually.}
\end{array}
\]

where $\Psi = \Phi^\sim$. Problem: relate state in $B$ to state in $A$. 

12
More grit

Given

\[ A : \mathbb{P}(S) \]
\[ B : \forall s \in S. A(s) \rightarrow \mathbb{P}(S) \]

initial condition, termination/invariant condition

\[ (\forall s \in S, p \in A(s)) \Phi^*(B(s, p), s) \]
\[ (\exists s \in S, p \in A(s)) \Psi^\infty(B(s, p), s) \]
Inconclusion

I freely admit I don’t (continuously) find these suggestions very convincing.

Some directions:

- Case studies.
- Thorough study of refinement calculus. (Different sub-species of predicate transformer: conjunctive, continuous, . . . commuting with intersections/unions of various kinds)
- Relate to work linear-time temporal logic. (eg. Lamport’s TLA.)
- Relate to work in formal topology. (Sambin, Mulvey etc.)

Work on type theory: coinduction (in ‘intensional’ type theory).