Retrenchment: an overview

R. Banach
Computer Science Department
University of Manchester, UK

Contents:

1. Origins ... model oriented refinement.
2. What's up with model oriented refinement?
3. The inception of retrenchment.
4. Retrenchment: key issues and opportunities.
5. The Retrenchment Homepage.

1. Origins ... model oriented refinement.

Refinement is a word used to mean a large number of different things in different contexts.

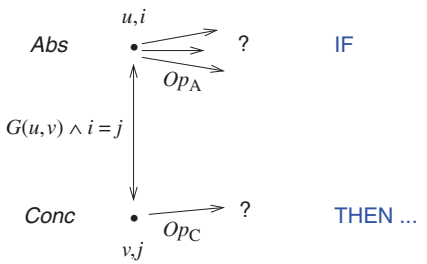
In model oriented refinement, we build models of the system, by specifying:

- the state (and I/O) space of a model
- the operations (or events) of a model: via eg.
  transition systems,
  programming notations,
  predicate transformers,
  etc.

Models can then be related pairwise by REFINEMENT. This usually involves a notion of *correctness*, relying on the *substitutivity*, of some **concrete** system behaviours for some **abstract** system ones, intended to help move closer to an implementation, and leading to *sufficient conditions* for refinement. Often there are two key conditions:

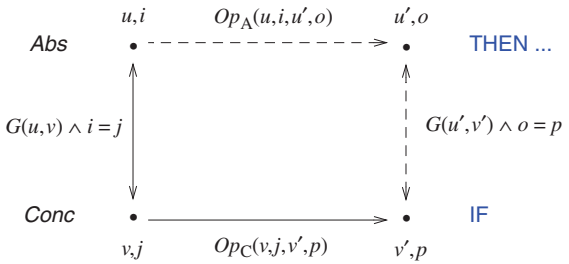
Applicability and Correctness

Applicability



If the abstract system can make an Op_{A} move, then the concrete system can make an Op_{C} move.

Correctness



If the concrete system actually makes an Op_C move, then the move can be simulated by the abstract system making an Op_A move.
The forward simulation criterion, the focus of our interest from now on.

Refinement Success Stories

There are various detailed theoretical variations on model oriented refinement, and various specific languages and implementations. The implementations of refinement have had some outstanding successes in the construction of dependable industrial scale systems of high criticality.

Some languages: Z, VDM, B, RAISE, ASM.

Some key projects:

- Mondex Purse, Multos OS (Z)
- MÉTÉOR (and many more French and other railway systems) (B)
- Prolog, C, Java (and many more) language definitions (ASM)

2. What's up with model oriented refinement?

Refinement can work wonderfully well in certain circumstances. Its paradigm of building models of the system at various levels of abstraction fits very naturally with the desire of designers to manipulate 'solid' representations of the system.

However, there are aspects of the real world design activity that can get into tension with refinement.

Refinement 1: Utopia

1. We design the abstract model. It captures the requirements of the system.
2. We refine the abstract model, moving towards a more efficient one.

Wonderful when it works.

Crucially, it presupposes that the natural intuitions about the structures of the desired system correspond fairly closely to the ingredients of the abstract model.

Refinement 2: Reality

1. Designers have a fairly reasonable intuitive idea of how the concrete model looks. They are much less clear about how the abstract model looks.
2. They work backwards to reverse engineer the abstract model from the concrete one, removing those elements which experience has taught that refinement can reinstate.

Where have the system requirements gone?

What is the relationship of the manufactured abstract model to the desired system?

What use is the refinement activity if the abstract model bears little or no relation to original system desiderata?

Well ..... it was mental gymnastics. It forced you to think deeply about the system (clearly it did, the refinement wasn't obvious, else we would be in Utopia).

Thinking hard about a problem is always beneficial.

Refinement 3: Tragedy

Communities who would make eager and serious use of refinement ... often can't.

1. Critical systems developers:
   - Need techniques giving very high assurance.
   - Understand and can benefit from the formal approach.
   - Can finance expensive technology (eg. intellectual) to gain high assurance.
2. Often physical models are involved:
   - The continuous / discrete transition in modelling (as understood by engineers) is invariably not doable within (strict) refinement, restricting the scope of formal modelling.
   - So at best, the abstract model ends up *already* in the discrete domain, bypassing most of the serious design.
3. Even for purely discrete applications:
   - The real world never starts from a blank sheet, impeding ideal refinement.
   - The complexity of real world applications can prohibit 100% faithful models.
   - Management issues can prevent 100% adherence to refinement ideals.

More on Management Issues:

The management usually *don't care* about the technical niceties of refinement.

- The management *may not permit* some change in a model in order to fix some technical hitch.
- Real engineering applications *never start from a blank sheet*.
- There *may be no time left* to fix things.
- The same basic idea may have different implementations *not refinable* from a common abstraction.
- The need of the specification to communicate *may override* its need to be refinable to implementation.

Things like these may compromise the pursuit of refinement in the ideal sense.

Example: Adding an element to a set

Abstract world:

$Add_elem_A(new) =$
$\quad myset := myset \cup \{new\}$

$myset$ is just a set
with a **finite** cardinality

constraint trickles up

Concrete world:

$Add_elem_C(new) =$
$\quad$ IF $new \notin \text{ran}(myseq)$
$\quad$ THEN $myseq := myseq \frown [new]$

$myset$ is represented as a sequence
with a **finite** length

constraint trickles up

Implementation world:

$Add_elem_I(new) =$
$\quad$ IF $new \notin \text{ran}(myarray)$
$\quad$ THEN $myarray(next) := new \ ; \ next := next + 1$

$myseq$ is represented as an array
with a **finite** bound

Example: Jet engine fuel supply control

Engineer's View	Refinement theorist's view
Step 1: Physics of hot fluids High temperature materials Aerodynamics etc.	Step 1: Finite arithmetic
... ...	... ...
Step n: Discrete algorithm	Step n: Put in the physics

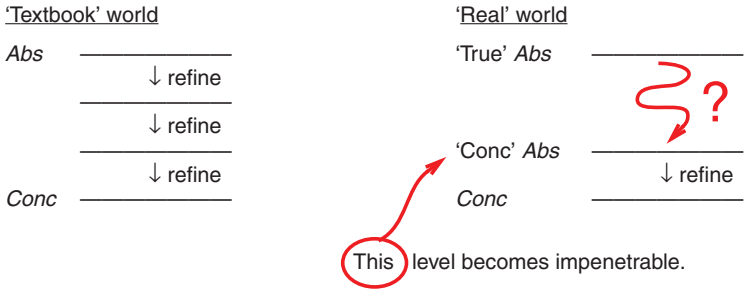
Why? ... Because refinement is a kind of conservative extension.

It is normally relatively straightforward to impose additional constraints during refinement, but much more difficult to contradict previously adopted properties.

In fact many refinements can be seen to be built out of orderings of the constraints whose conjunction defines the concrete model. **(Refinement as spec. constructor.)**

OK if it makes sense to domain experts, more questionable if not.

Example: Small and large applications in general



The bigger the application gets, the worse the problem becomes.

And it is absolutely a question of ***SCALE.***

3. The inception of retrenchment.

The ferocity of the refinement POs is what restricts their application in many areas.

What can we do about the ferocity of the refinement POs?

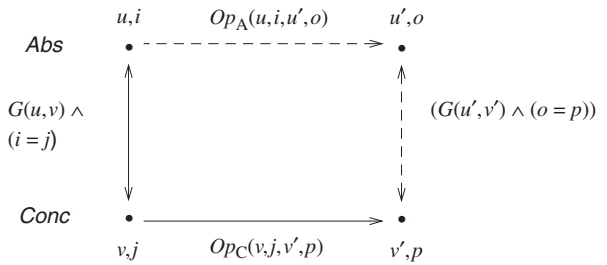
We can attempt to judiciously weaken them.

This of course would have consequences ... Refinement is **derived** from the prior assumption of substitutivity of concrete for abstract. Any interference with the refinement POs can fatally wound this link with substitutivity properties.

We are prepared to forgo this highly desirable aspect for the sake of:

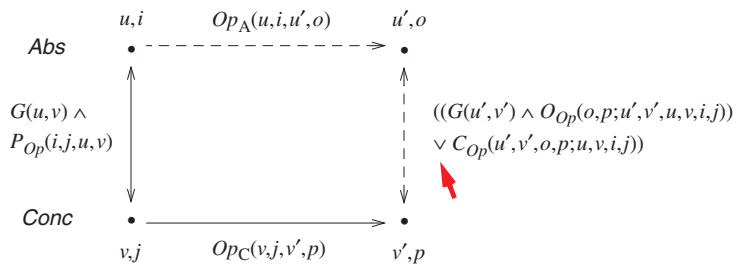
- Being able to address more (and more of) applications contexts formally.
- Being able to live with real world and management constraints.

The refinement PO



$$G(u,v) \wedge (i=j) \wedge Op_C(v,j,v',p) \Rightarrow$$
$$(\exists\, u',o \bullet Op_A(u,i,u',o) \wedge (G(u',v') \wedge (o=p)))$$

The retrenchment PO



$$G(u,v) \wedge P_{Op}(i,j,u,v) \wedge Op_{\mathrm{C}}(v,j,v',p) \Rightarrow (\exists\, u', o \bullet Op_{\mathrm{A}}(u,i,u',o) \wedge ((G(u',v') \wedge O_{Op}(o,p;u',v',u,v,i,j)) \vee C_{Op}(u',v',o,p;u,v,i,j)))$$

Definition of a retrenchment between *Abs* and *Conc*

A retrenchment is defined by the following data:

$\mathrm{Ops}_{\mathrm{A}} \subseteq \mathrm{Ops}_{\mathrm{C}}$ (the inclusion of operation names can be proper)

For each abstract operation name a relation $Op_{\mathrm{A}} : \mathrm{U} \times \mathrm{I}_{Op_{\mathrm{A}}} \leftrightarrow \mathrm{U} \times \mathrm{O}_{Op_{\mathrm{A}}}$
For each concrete operation name a relation $Op_{\mathrm{C}} : \mathrm{V} \times \mathrm{J}_{Op_{\mathrm{C}}} \leftrightarrow \mathrm{V} \times \mathrm{P}_{Op_{\mathrm{C}}}$

$G(u,v)$ (the retrieve (or glueing) relation)

$P_{Op}(i,j,u,v)$ (the within (or provided) relation)
$O_{Op}(o,p;u',v',u,v,i,j)$ (the output relation)
$C_{Op}(u',v',o,p;u,v,i,j)$ (the concedes relation)
} per $Op \in \mathrm{Ops}_{\mathrm{A}}$

Initialisation PO: $Init_{\mathrm{C}}(v') \Rightarrow (\exists\, u' \bullet Init_{\mathrm{A}}(u') \wedge G(u',v'))$ (as for refinement)

Operation PO: (per $Op \in \mathrm{Ops}_{\mathrm{A}}$)
$$G(u,v) \wedge P_{Op}(i,j,u,v) \wedge Op_{\mathrm{C}}(v,j,v',p) \Rightarrow (\exists\, u', o \bullet Op_{\mathrm{A}}(u,i,u',o) \wedge ((G(u',v') \wedge O_{Op}(o,p;u',v',u,v,i,j)) \vee C_{Op}(u',v',o,p;u,v,i,j)))$$

Remarks

- The output form; cf. $G \wedge P_{Op} \ldots \Rightarrow (\exists \ldots (G \vee C_{Op}))$ 'primitive retrenchment'.
  As just a container for properties, primitive retrenchment is fine.
  For theoretical work, the output form is much more convenient. ■■■
- The operation PO is the **definition** of retrenchment.
  Unlike refinement, in which the operation PO is **derived** from substitutivity.
- This is a partial correctness formulation. ■■■
- The operation PO defaults to the refinement operation PO;
  (set $P_{Op}(i,j,u,v) = (i = j)$; $O_{Op}(o,p;u',v',u,v,i,j) = (o = p)$; $C_{Op}(u',v',o,p;u,v,i,j) =$ **false**)
  There are other defaults. ■■■
- The operation PO is extremely expressive; it can accommodate *any* property.
  Nevertheless the PO as a whole still has to be provable.

Refinement and retrenchment compared

Refinement	Retrenchment
Guarantees concrete model is faithful to abstract model via G.	Documents model change in a structured way via G, P_{Op}, O_{Op}, C_{Op}.
Few properties addressable via G.	Many properties addressable via G, P_{Op}, O_{Op}, C_{Op}.
Forces designer to discharge PO; PO is highly constrained; in particular it is nonlinear in G.	Forces designer to discharge PO; PO is very liberal; in particular, despite being nonlinear in G, it is linear in P_{Op}, O_{Op}, C_{Op}.
Refinement is a limit of retrenchment, and so has a stronger theory than retrenchment.	Retrenchment is a generalisation of refinement, and so has wider applicability than refinement.
Fewer situations are describable using refinement.	Retrenchment has a weaker theory than refinement.

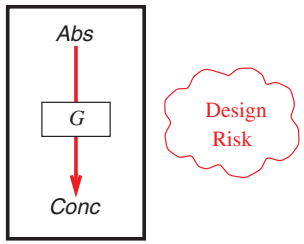
Design and risk

Refinement

Refinement tends to expel design and risk from the concretisation/ accretion process. The process can become monolithic.

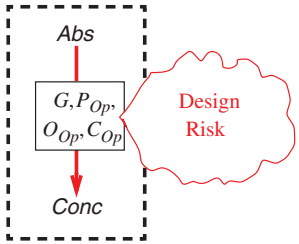
Retrenchment

Retrenchment embraces design and risk within the evolution/ concretisation process. The process can separate concerns.



Design Risk

'Black Box'

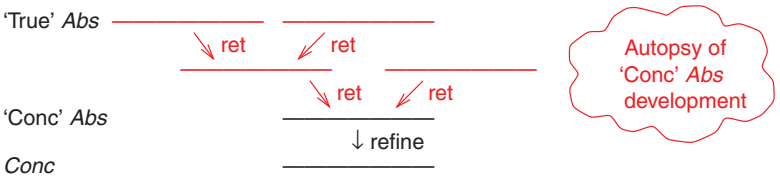


Design Risk

'Glass Box'

The real world again

'Real' world



The autopsy gives a rationalisation in domain experts' terms of the structure of the refinable abstract model.

No need to reverse engineer the true abstract model from the concrete model.

An idealisation of the process ... many variations possible.

No technical commitment to which ret steps are 'requirements engineering' and which are 'development'.

4. Retrenchment: key issues and opportunities.

- Default retrenchments 74
- Composition/decomposition 83
- Varieties of retrenchment; general retrenchment 136
- Algebraic theory of retrenchment and refinement 143
- Patterns of Retrenchment 161
- Applicability/correctness 165
- Simulation and behavioural properties 170
- Coarse grained retrenchment 185
- Probabilistic retrenchment 194
- Applications and special purpose retrenchments 211
- Tool support 213
- Methodology 215

Default retrenchments

1. Defaulting to refinement:
   $P_{Op}(i,j,u,v) := (i = j)$; $O_{Op}(o,p;u',v',u,v,i,j) := (o = p)$; $C_{Op}(u',v',o,p;u,v,i,j) :=$ **false**
2. Trivial retrenchments:
   $P_{Op}(i,j,u,v) :=$ **false** and/or $C_{Op}(u',v',o,p;u,v,i,j) :=$ **true**
3. Default retrenchments:
   Given $G(u,v)$, $P_{Op}(i,j,u,v)$, $O_{Op}(o,p;u',v',u,v,i,j)$ define:
   $P^{\mathrm{Def}}{}_{Op}(i,j,u,v) := (G(u,v) \wedge P_{Op}(i,j,u,v) \wedge (\exists\, u',o,v',p \bullet Op_{\mathrm{A}}(u,i,u',o) \wedge Op_{\mathrm{C}}(v,j,v',p)))$
   $C^{\mathrm{Def}}{}_{Op}(u',v',o,p;u,v,i,j) :=$
   $(G(u,v) \wedge P_{Op}(i,j,u,v) \wedge (\exists\, u',o,v',p \bullet Op_{\mathrm{A}}(u,i,u',o) \wedge Op_{\mathrm{C}}(v,j,v',p)) \wedge$
   $\neg((G(u',v') \wedge O_{Op}(o,p;u',v',u,v,i,j))))$

- 'Any two systems can be related via a default retrenchment.'
- 'Any jumble of words can be assembled into a meaningless phrase.'
- Retrenchment, like speech, should be undertaken with a purpose ... validation.

Default and trivial retrenchments are top and bottom of a 'lattice'.

Composition/decomposition

Vertical composition

Suppose given:

Abs/Conc retrenchment: $Op_{\mathrm{A}} \lesssim_{G_1,P_1,O_1,C_1} Op_{\mathrm{C}}$ from (u,i,u',o) to (v,j,v',p)

Conc/Imp retrenchment: $Op_{\mathrm{C}} \lesssim_{G_2,P_2,O_2,C_2} Op_{\mathrm{I}}$ from (v,j,v',p) to (w,k,w',q)

Then there is an

Abs/Imp retrenchment: $Op_{\mathrm{A}} \lesssim_{G_{(1,2)},P_{(1,2)},O_{(1,2)},C_{(1,2)}} Op_{\mathrm{I}}$ from (u,i,u',o) to (w,k,w',q)

given by:

$G_{(1,2)}(u,w) = (\exists\, v \bullet G_1(u,v) \wedge G_2(v,w))$

$P_{Op,(1,2)}(i,k,u,w) = (\exists\, v,j \bullet G_1(u,v) \wedge G_2(v,w) \wedge P_{Op,1}(i,j,u,v) \wedge P_{Op,2}(j,k,v,w))$

$O_{Op,(1,2)}(o,q;u',w'\ldots) = (\exists\, v',p,v,j \bullet O_{Op,1}(o,p;u',v'\ldots) \wedge O_{Op,2}(p,q;v',w'\ldots))$

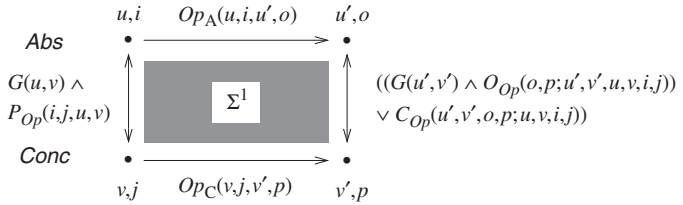
$C_{Op,(1,2)}(u',w',o,q;\ldots) = (\exists\, v',p,v,j \bullet$
$(G_1(u,v) \wedge O_{Op,1}(o,p;u',v'\ldots) \wedge C_{Op,2}(v',w',p,q;\ldots)) \vee$
$(C_{Op,1}(u',v',o,p;\ldots) \wedge G_2(v,w) \wedge O_{Op,2}(p,q;v',w'\ldots)) \vee$
$(C_{Op,1}(u',v',o,p;\ldots) \wedge C_{Op,2}(v',w',p,q;\ldots)))$

'Plain vanilla' vertical composition. Associative. There are many stronger forms. ■■■

Horizontal composition

Unrestricted horizontal composition is in general problematic. What about when the final configuration of one step cannot satisfy the $G \wedge P_{Op}$ of the next one?

The unrestricted case is best studied directly via the simulation relation:



$G(u,v) \wedge P_{Op}(i,j,u,v) \wedge Op_{\mathrm{C}}(v,j,v',p) \wedge Op_{\mathrm{A}}(u,i,u',o) \wedge$
$((G(u',v') \wedge O_{Op}(o,p;u',v',u,v,i,j)) \vee C_{Op}(u',v',o,p;u,v,i,j))$

This can capture 'retrenchment is like refinement except round the edges'.

Horizontal composition

Suppose given:

Abs/Conc retrenchment: $Op_{1,\mathrm{A}} \lesssim_{G,P_1,O_1,C_1} Op_{1,\mathrm{C}}$ from $(u,i_1,\underline{u},o_1)$ to $(v,j_1,\underline{v},p_1)$

Abs/Conc retrenchment: $Op_{2,\mathrm{A}} \lesssim_{G,P_2,O_2,C_2} Op_{2,\mathrm{C}}$ from $(\underline{u},i_2,u',o_2)$ to $(\underline{v},j_2,v',p_2)$

Then there is an *Abs/Conc* retrenchment:

$Op_{(1;2),\mathrm{A}} \lesssim_{G,P_{(1;2)},O_{(1;2)},C_{(1;2)}} Op_{(1;2),\mathrm{C}}$ from
$(u,[i_1,i_2],u',[o_1,o_2])$ to $(v,[j_1,j_2],v',[p_1,p_2])$

given by:

$Op_{(1;2),\mathrm{A}} = Op_{1,\mathrm{A}};Op_{2,\mathrm{A}} \qquad Op_{(1;2),\mathrm{C}} = Op_{1,\mathrm{C}};Op_{2,\mathrm{C}}$

$P_{Op,(1;2)}([i_1,i_2],[j_1,j_2],u,v) = G(u,v) \wedge P_{Op,1}(i_1,j_1,u,v) \wedge$
$\mathrm{wp}(\Sigma^1(Op_1),(G(\underline{u},\underline{v}) \wedge P_{Op,2}(i_2,j_2,\underline{u},\underline{v})))$

$O_{Op,(1;2)}([o_1,o_2],[p_1,p_2];\ldots) = (\exists\, \underline{u},\underline{v} \bullet O_{Op,1}(o_1,p_1;u',v'\ldots) \wedge O_{Op,2}(o_2,p_2;\underline{u},\underline{v}\ldots))$

$C_{Op,(1;2)}(u',v',[o_1,o_2],[p_1,p_2];\ldots) = (\exists\, \underline{u},\underline{v} \bullet$
$(G(\underline{u},\underline{v}) \wedge O_{Op,1}(o_1,p_1;\underline{u},\underline{v}\ldots) \wedge C_{Op,2}(u',v',o_2,p_2;\ldots)) \vee$
$(C_{Op,1}(\underline{u},\underline{v},o_1,p_1;\ldots) \wedge G(u',v') \wedge O_{Op,2}(o_2,p_2;u',v'\ldots)) \vee$
$(C_{Op,1}(\underline{u},\underline{v},o_1,p_1;\ldots) \wedge C_{Op,2}(u',v',o_2,p_2;\ldots))$

N.B. $\mathrm{wp}(\Sigma^1(Op_1),(G \wedge P_{Op,2})) = (\forall\, \underline{u},\underline{v},o_1,p_1 \bullet (u,i_1,\underline{u},o_1)\ \Sigma^1(Op_1)\ (v,j_1,\underline{v},p_1) \Rightarrow (G \wedge P_{Op,2}))$

This can capture 'bulk sequential composition' of operations, but it's rather stronger than the simulation relation.

Dataflow composition

Trivial state $*$; outputs of one step plug into inputs of next step.
Much better behaved than unrestricted horizontal composition.

Suppose given:

Abs/Conc retrenchment: $Op_{1,\mathrm{A}} \lesssim_{G,P_1,O_1,C_1} Op_{1,\mathrm{C}}$ from (i_0,o_1) to (j_0,p_1)

Abs/Conc retrenchment: $Op_{2,\mathrm{A}} \lesssim_{G,P_2,O_2,C_2} Op_{2,\mathrm{C}}$ from (i_1,o_2) to (j_1,p_2)
and $(o_1 = i_1)$ and $(p_1 = j_1)$

Then there is an *Abs/Conc* retrenchment:

$Op_{(1\rangle 2),\mathrm{A}} \lesssim_{G,P_{(1\rangle 2)},O_{(1\rangle 2)},C_{(1\rangle 2)}} Op_{(1\rangle 2),\mathrm{C}}$ from (i_0,o_2) (via $(o_1 = i_1)$) to (j_0,p_2) (via $(p_1 = j_1)$)

given by:

$Op_{(1\rangle 2),\mathrm{A}} = Op_{1,\mathrm{A}};Op_{2,\mathrm{A}} \qquad Op_{(1\rangle 2),\mathrm{C}} = Op_{1,\mathrm{C}};Op_{2,\mathrm{C}}$

$P_{(Op,1\rangle Op,2)}(i_0,j_0) = P_{Op,1}(i_0,j_0)$

$O_{(Op,1\rangle Op,2)}(o_2,p_2;i_0,j_0) = (\exists\, a,c \bullet O_{Op,2}(o_2,p_2;a,c) \wedge O_{Op,1}(a,c;i_0,j_0))$

$C_{(Op,1\rangle Op,2)}(o_2,p_2;i_0,j_0) = (\exists\, a,c \bullet (O_{Op,2}(o_2,p_2;a,c) \wedge C_{Op,1}(a,c;i_0,j_0)) \vee$
$(C_{Op,2}(o_2,p_2;a,c) \wedge O_{Op,1}(a,c;i_0,j_0)) \vee$
$(C_{Op,2}(o_2,p_2;a,c) \wedge C_{Op,1}(a,c;i_0,j_0)))$

provided $O_{Op,1} \vee C_{Op,1} \Rightarrow P_{Op,2}$

Synchronous parallel composition

Partial states.

Suppose given:

Abs/Conc retrenchment: $Op_{1,\mathrm{A}} \lesssim_{G_1,P_1,O_1,C_1} Op_{1,\mathrm{C}}$ from (u_1,i_1,u'_1,o_1) to (v_1,j_1,v'_1,p_1)

Abs/Conc retrenchment: $Op_{2,\mathrm{A}} \lesssim_{G_2,P_2,O_2,C_2} Op_{2,\mathrm{C}}$ from (u_2,i_2,u'_2,o_2) to (v_2,j_2,v'_2,p_2)

Then there is an *Abs/Conc* retrenchment:

$Op_{(1|2),\mathrm{A}} \lesssim_{G_{(1|2)},P_{(1|2)},O_{(1|2)},C_{(1|2)}} Op_{(1|2),\mathrm{C}}$ from
$((u_1,u_2),(i_1,i_2),(u'_1,u'_2),(o_1,o_2))$ to $((v_1,v_2),(j_1,j_2),(v'_1,v'_2),(p_1,p_2))$

given by:

$Op_{(1|2),\mathrm{A}} = Op_{1,\mathrm{A}} \wedge Op_{2,\mathrm{A}} \qquad Op_{(1|2),\mathrm{C}} = Op_{1,\mathrm{C}} \wedge Op_{2,\mathrm{C}}$

$G_{(1|2)}((u_1,u_2),(v_1,v_2)) = G_1(u_1,v_1) \wedge G_2(u_2,v_2)$

$P_{Op,(1|2)}((i_1,i_2),(j_1,j_2),(u_1,u_2),(v_1,v_2)) = P_{Op,1}(i_1,j_1,u_1,v_1) \wedge P_{Op,2}(i_2,j_2,u_2,v_2)$

$O_{Op,(1|2)}((o_1,o_2),(p_1,p_2);\ldots) = O_{Op,1}(o_1,p_1;u'_1,v'_1\ldots) \wedge O_{Op,2}(o_2,p_2;u'_2,v'_2\ldots)$

$C_{Op,(1|2)}((u'_1,u'_2),(v'_1,v'_2),(o_1,o_2),(p_1,p_2);\ldots) =$
$(G_1(u_1,v_1) \wedge O_{Op,1}(o_1,p_1;u'_1,v'_1\ldots) \wedge C_{Op,2}(u'_2,v'_2,o_1,p_1;\ldots)) \vee$
$(C_{Op,1}(u'_1,v'_1,o_1,p_1;\ldots) \wedge G_2(u_2,v_2) \wedge O_{Op,2}(o_2,p_2;u'_2,v'_2\ldots)) \vee$
$(C_{Op,1}(u'_1,v'_1,o_1,p_1;\ldots) \wedge C_{Op,2}(u'_2,v'_2,o_1,p_1;\ldots))$

Fusion composition

Simultaneous retrenchments. Conjunctive composition.
Equiextensional simulation relation. Add guards if necessary.

Suppose given:

Abs/Conc retrenchment: $Op_{\mathrm{A}} \lesssim_{G,P_1,O_1,C_1} Op_{\mathrm{C}}$ from (u,i,u',o) to (v,j,v',p)

Abs/Conc retrenchment: $Op_{\mathrm{A}} \lesssim_{G,P_2,O_2,C_2} Op_{\mathrm{C}}$ from (u,i,u',o) to (v,j,v',p)

Then there is an *Abs/Conc* retrenchment:

$Op_{\mathrm{A}} \lesssim_{G,P_{(1\wedge2)},O_{(1\wedge2)},C_{(1\wedge2)}} Op_{\mathrm{C}}$ from (u,i,u',o) to (v,j,v',p)

given by:

$P_{Op,(1\wedge2)}(i,j,u,v) = P_{Op,1}(i,j,u,v) \wedge P_{Op,2}(i,j,u,v)$

$O_{Op,(1\wedge2)}(o,p;u',v'\ldots) = O_{Op,1}(o,p;u',v'\ldots) \wedge O_{Op,2}(o,p;u',v'\ldots)$

$C_{Op,(1\wedge2)}(u',v',o,p;\ldots) = (G(u,v) \wedge O_{Op,1}(o,p;u',v'\ldots) \wedge C_{Op,2}(u',v',o,p;\ldots)) \vee$
$(C_{Op,1}(u',v',o,p;\ldots) \wedge G(u,v) \wedge O_{Op,2}(o,p;u',v'\ldots)) \vee$
$(C_{Op,1}(u',v',o,p;\ldots) \wedge C_{Op,2}(u',v',o,p;\ldots))$

Fusion composition

Simultaneous retrenchments. Disjunctive composition.

Suppose given:

Abs/Conc retrenchment: $Op_{\mathrm{A}} \lesssim_{G,P_1,O_1,C_1} Op_{\mathrm{C}}$ from (u,i,u',o) to (v,j,v',p)

Abs/Conc retrenchment: $Op_{\mathrm{A}} \lesssim_{G,P_2,O_2,C_2} Op_{\mathrm{C}}$ from (u,i,u',o) to (v,j,v',p)

Then there is an *Abs/Conc* retrenchment:

$Op_{\mathrm{A}} \lesssim_{G,P_{(1\vee2)},O_{(1\vee2)},C_{(1\vee2)}} Op_{\mathrm{C}}$ from (u,i,u',o) to (v,j,v',p)

given by:

$P_{Op,(1\vee2)}(i,j,u,v) = P_{Op,1}(i,j,u,v) \vee P_{Op,2}(i,j,u,v)$

$O_{Op,(1\vee2)}(o,p;u',v'\ldots) = O_{Op,1}(o,p;u',v'\ldots) \vee O_{Op,2}(o,p;u',v'\ldots)$

$C_{Op,(1\vee2)}(u',v',o,p;\ldots) = C_{Op,1}(u',v',o,p;\ldots) \vee C_{Op,2}(u',v',o,p;\ldots)$

For all proposed notions of composition, the key issues are:

- Does the notion compose to give a retrenchment?
- Does the notion compose to give a retrenchment satisfying its own definition?
- Does the notion yield a composition which is associative?
- Whenever the notion is defined by propositional manipulations of retrenchment data, then there will be stronger variants.

Decomposition (Mike Poppleton)

Suppose given:

Abs/Conc retrenchment: $Op_{\mathrm{A}} \preceq_{G,P,O,C} Op_{\mathrm{C}}$ from (u,i,u',o) to (v,j,v',p)

Suppose that the domain of Op_{A} decomposes as $\mathrm{dom}(Op_{\mathrm{A}}) = a_{Op,1} \uplus a_{Op,2} \uplus \ldots a_{Op,k}$
Suppose that the domain of Op_{C} decomposes as $\mathrm{dom}(Op_{\mathrm{C}}) = c_{Op,1} \uplus c_{Op,2} \uplus \ldots c_{Op,l}$

Let $Op_{\mathrm{A},i} = a_{Op,i} \lhd Op_{\mathrm{A}}$ and $Op_{\mathrm{C},j} = c_{Op,j} \lhd Op_{\mathrm{C}}$ and $P_{Op,ij} = a_{Op,i} \lhd P_{Op} \rhd c_{Op,j}$

Then for all pairs i,j: $Op_{\mathrm{A}} \preceq_{G,P_{ij},O,C} Op_{\mathrm{C}}$ and $Op_{\mathrm{A},i} \preceq_{G,P_{ij},O,C} Op_{\mathrm{C},j}$

The original retrenchment $Op_{\mathrm{A}} \preceq_{G,P,O,C} Op_{\mathrm{C}}$ can now be recovered by fusion composition. ■■■

Varieties of retrenchment; general retrenchment

Primitive retrenchment

$G \wedge P_{Op} \ldots \Rightarrow (\exists \ldots (G \vee C_{Op}))$

Output retrenchment

$G \wedge P_{Op} \ldots \Rightarrow (\exists \ldots ((G \wedge O_{Op}) \vee C_{Op}))$

Sharp retrenchment

$G \wedge P_{Op} \ldots \Rightarrow (\exists \ldots ((G \vee C_{Op}) \wedge V_{Op}))$

Sharp output retrenchment

$G \wedge P_{Op} \ldots \Rightarrow (\exists \ldots (((G \wedge O_{Op}) \vee C_{Op}) \wedge V_{Op}))$

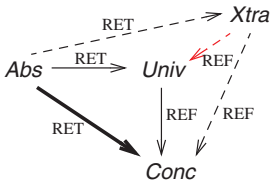
General retrenchment

$\Phi_{Op} \ldots \Rightarrow (\exists \ldots \Theta_{Op})$ ... where Φ_{Op} and Θ_{Op} are as general as possible.

Associativity!

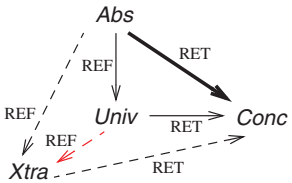
Algebraic theory of retrenchment and refinement

Factorisations of a retrenchment (Czeslaw Jeske's M.Sc.)



Why?

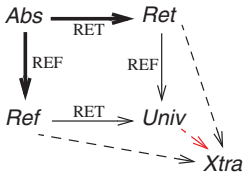
Lift new requirements to *Abs* level automatically.



Why?

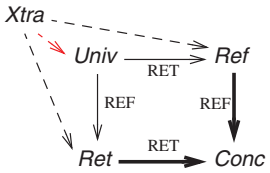
Relate *Abs* to 'Conc' *Abs* automatically.

Completion constructions I (Czeslaw Jeske's Ph.D.)



Why?

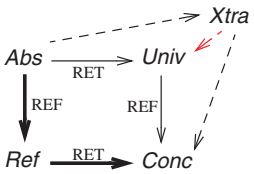
Generate refinement of new requirements automatically.



Why?

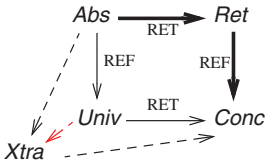
Generate abstraction of low level unpatch automatically.

Completion constructions II (Czeslaw Jeske's Ph.D.)



Why?

Generate abstraction of low level patch automatically.



Why?

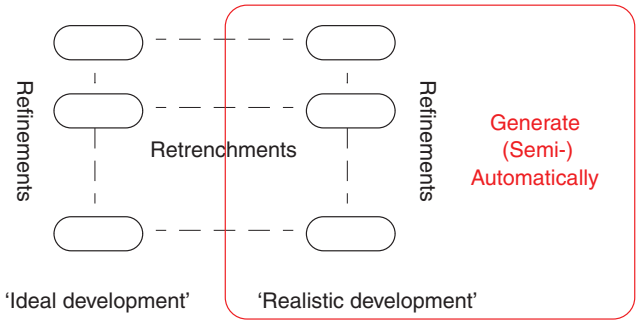
Generate refinement of new requirements automatically.

■■■

Patterns of Retrenchment

Just as software has identified commonly occurring 'patterns', we can see something similar for retrenchment, *independent of the issue dealt with in the retrenchment*.

Most evident is the '***Tower Pattern***', built top down or bottom up ...:



Applicability/correctness

Various refinement theories augment the operation PO with other criteria that a pair of operations must satisfy to be in the corresponding refinement relation.

These criteria depend strongly on the programming/systems theory used.

Examples:

- Applicability ... eg. Z methodology, ASM methodology
- Termination ... eg. Refinement calculus, B-Method
- Feasibility ... eg. Refinement calculus, B-Method

These give rise to POs that are implications '$\theta_A \Rightarrow \theta_C$' from abstract to concrete, or '$\theta_C \Rightarrow \theta_A$' from concrete to abstract, depending on the criterion and the theory.

Moreover, all the criteria known, depend on just the before states and inputs, either abstract or concrete.

Retrenchment advocates a generally applicable way of embracing all such theories.

Broad Principles:

1. Useful systems consist of successful operation steps; failure of applicability/ termination/feasibility etc. is connected with unsuccessful steps. Retrenchment should be primarily concerned with useful/successful steps. These are steps for which ALL the antecedents are valid (and thus all the consequents too).
2. Retrenchment wants to *favour* 'stepwise simulation' theorems. ■■■

The simplest solution:

Given a specific theory of correctness, and a prospective retrenchment between systems *Abs/Conc* given by G, P_{Op}, O_{Op}, C_{Op} :

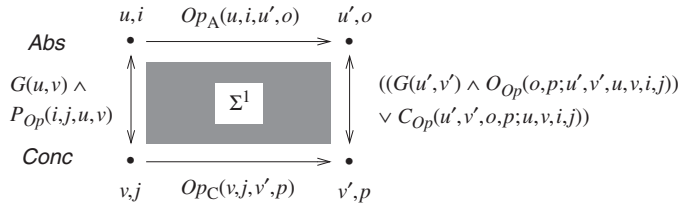
Insist that for every relevant θ of the theory in question:

$$\mathrm{dom}(G \wedge P_{Op}) \subseteq \theta_{A,Op}$$
$$\mathrm{ran}(G \wedge P_{Op}) \subseteq \theta_{C,Op}$$

This ensures that the transitions about which the retrenchment operation PO speaks, are indeed well behaved, useful/successful steps. Also the PO itself is unaffected.

Simulation and behavioural properties

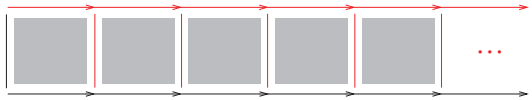
The simulation relation holds when all the facts spoken of in the PO are verified:



$$G(u,v) \wedge P_{Op}(i,j,u,v) \wedge Op_C(v,j,v',p) \wedge Op_A(u,i,u',o) \wedge$$
$$((G(u',v') \wedge O_{Op}(o,p;u',v',u,v,i,j)) \vee C_{Op}(u',v',o,p;u,v,i,j))$$

A **fragment** is a sequence of steps $\mathcal{S} = [\, u_0 \text{-}(i_0, Op_{A,0}, o_1)\text{-}\!\rightarrow u_1 \text{-}(i_1, Op_{A,1}, o_2)\text{-}\!\rightarrow u_2 \ldots \,]$
A **multifragment** is a sequence of fragments $\mathcal{S} = [\, \mathcal{S}_0, \mathcal{S}_1, \ldots \,]$

The ideal for retrenchment is the stepwise simulation relationship for fragments, i.e. an induction-like result thus:



Retrenchment can *encourage* if not guarantee this, as the final configuration of one step need not satisfy the $G \wedge P_{Op}$ of the next one (cf. horizontal composition).

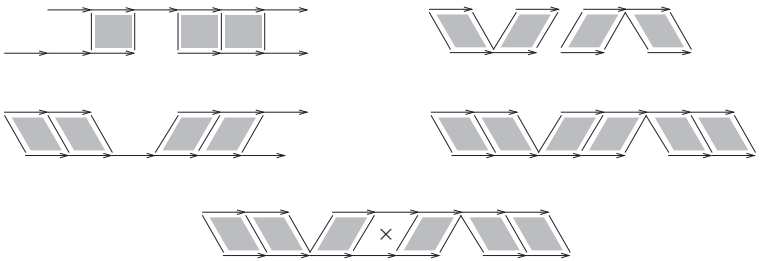
With additional (strong, effectively refinement-like) assumptions, it can be achieved.

But ... that misses the point.

Retrenchment is primarily for cases when the induction fails.

So it's best to deal with simulation directly.

Multifragment simulation permits situations $\mathcal{S}\,\Sigma\,\mathcal{T}$ like the following:



Vertical composition of multifragment simulations is *encouraged* by biretrenchment.

Biretrenchment:

$$G \wedge P_{Op} \wedge Op_C \Rightarrow (\exists \ldots\, Op_A \wedge ((G' \wedge O_{Op}) \vee C_{Op}))$$
$$G \wedge P_{Op} \wedge Op_A \Rightarrow (\exists \ldots\, Op_C \wedge ((G' \wedge O_{Op}) \vee C_{Op}))$$

&

Suitably constrained to avoid excessive fragmentation, sets of multifragments define **properties**.

The multifragment simulation relation yields simulation transformers for properties:

$$[\Sigma]\,\mathcal{SS} = \{\mathcal{T} \mid (\exists\, \mathcal{S} \bullet \mathcal{S} \in \mathcal{SS} \wedge \mathcal{S}\,\Sigma\,\mathcal{T}) \wedge (\forall\, \mathcal{S} \bullet \mathcal{S}\,\Sigma\,\mathcal{T} \Rightarrow \mathcal{S} \in \mathcal{SS})\}$$
$$\langle\Sigma\rangle\,\mathcal{SS} = \{\mathcal{T} \mid (\exists\, \mathcal{S} \bullet \mathcal{S} \in \mathcal{SS} \wedge \mathcal{S}\,\Sigma\,\mathcal{T})\}$$
$$[\Sigma]\,\mathcal{TT} = \{\mathcal{S} \mid (\exists\, \mathcal{T} \bullet \mathcal{T} \in \mathcal{TT} \wedge \mathcal{S}\,\Sigma\,\mathcal{T}) \wedge (\forall\, \mathcal{T} \bullet \mathcal{S}\,\Sigma\,\mathcal{T} \Rightarrow \mathcal{T} \in \mathcal{TT})\}$$
$$\langle\Sigma\rangle\,\mathcal{TT} = \{\mathcal{T} \mid (\exists\, \mathcal{T} \bullet \mathcal{T} \in \mathcal{TT} \wedge \mathcal{S}\,\Sigma\,\mathcal{T})\}$$

These map abstract properties to concrete properties and vice versa, and act like box and diamond operators in a modal algebra.

Unfortunately, the $[\Sigma]\,\mathcal{SS}$, $\langle\Sigma\rangle\,\mathcal{SS}$, $[\Sigma]\,\mathcal{TT}$, $\langle\Sigma\rangle\,\mathcal{TT}$ transformers leave the mapping of the non-simulable parts of multifragments completely unconstrained. This can be addressed by imposing additional constraints:

- via meta-linguistic means ...
- explicitly.

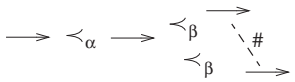
...

Coarse grained retrenchment

When a collection of individual steps is to be regarded as a whole, in effect the *only* issue is **atomicity**. (Without atomicity, what's the point?)

In coarse grained retrenchment atomicity is assumed.

In coarse grained retrenchment a partial states picture is adopted, the syntactic role is taken by relevant collections of individual steps which are encapsulated in prime event structures (ESs) $\mathcal{E} = (E, \leq, \#)$.

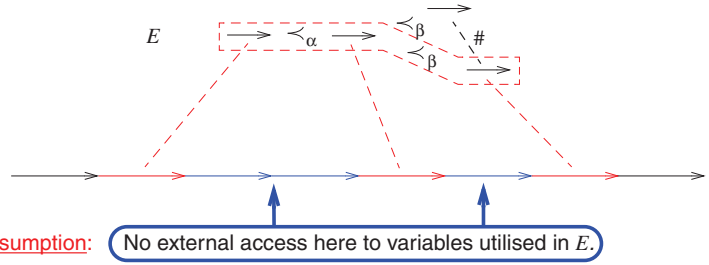


$$\leq = (\cup_\alpha \prec_\alpha)^*$$

$$e_1 \# e_2 \leq e_3 \Rightarrow e_1 \# e_3$$

Variable values at root and leaf events of E extend to the before- and after- values of the global state.

An execution structure is the corresponding (conflict free) runtime notion. It yields an execution fragment via serialisation. Fragments thus yield **deployments** of ESs.



Variable values at root and leaf events of E extend to before- and after- values of the global state.

These global state values appear in the retrenchment PO for corresponding abstract and concrete ESs.

Interaction of coarse grained retrenchment with composition/decomposition mechanisms yields a fertile infrastructure for sophisticated analyses of system behaviour.

Example: Aspect oriented programming

- Start with base language and functional aspect of top level operations.
- Identify 'weave points' for other aspects. Decompose top level operations at weave points using sequential/parallel decomposition, into finer grained actions.
- Capture the 'weaving in' of additional aspects as suitable coarse grained retrenchments of the original decomposed top level operations.
- Analyse the resulting coarse grained retrenchments. Do they yield the properties expected of the included additional aspects?

Example: Complex Systems

- Identify basic components of the complex system.
- Are they already too complicated to understand properly? If so simplify them.
- Determine the assembly of basic components to make up the global system, describe it using sequential/parallel/dataflow composition to get top level operations of the global system.
- Capture the incorporation of more of the properties of the basic components into the global model using successive coarse grained retrenchments. Can emerging properties be identified in the retrenchment analysis?
- Compare calculated retrenchment for the composed system with any ad hoc retrenchment written for it. Compare calculated retrenchments for the decomposed components with ad hoc retrenchments describing increments of basic component properties.

Probabilistic retrenchment

Fundamental *Abs* transition probabilities: $p_{Op_A}(u',o \mid u,i)$ so $\sum_{(u',o)} p_{Op_A}(u',o \mid u,i) = 1$

Given a before- distribution $p(u,i)$, the probability of achieving $\Phi(u',o)$ in one step is:

$\sum_{(u',o,u,i)} p_{Op_A}(u',o \mid u,i).p(u,i).\Phi(u',o)$ ($\Phi(u',o)$ the characteristic function of Φ)

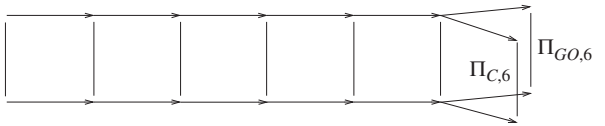
Similarly, calculate the probability of achieving a free stepwise simulation of length n, starting from (u,v) and with I/O (is,js,os,ps):

$n = 1$: $\Pi_{GO,1} = \sum_{(u_1,v_1)} \Pi_{GO,1}(u_1,v_1)$ where $\Pi_{GO,1}(u_1,v_1) =$
$p_{Op_{A,1}}(u_1,os_1 \mid u,is_0).p_{Op_{C,1}}(v_1,ps_1 \mid v,js_0).$
$G(u_1,v_1).O_{Op_1}(os_1,ps_1;u_1,v_1,u,v,is_0,js_0).G(u,v).P_{Op_1}(is_0,js_0,u,v)$
$\Pi_{C,1} = \sum_{(u_1,v_1)} \Pi_{C,1}(u_1,v_1)$ where $\Pi_{C,1}(u_1,v_1) =$
$p_{Op_{A,1}}(u_1,os_1 \mid u,is_1).p_{Op_{C,1}}(v_1,ps_1 \mid v,js_1).$
$C_{Op_1}(u_1,v_1,os_1,ps_1;u,v,is_0,js_0).G(u,v).P_{Op_1}(is_0,js_0,u,v)$

$\Pi_{GO,1} + \Pi_{C,1} - \Pi_{GOC,1} = 1$

$1 < l < n$: $\Pi_{GO,l} = \sum_{(u_l,v_l)} \Pi_{GO,l}(u_l,v_l)$ where $\Pi_{GO,l}(u_l,v_l) =$
$\sum_{(u_{l-1},v_{l-1})} p_{Op_{A,l}}(u_l,os_l \mid u_{l-1},is_{l-1}).p_{Op_{C,l}}(v_l,ps_l \mid v_{l-1},js_{l-1}).\Pi_{GO,l-1}(u_{l-1},v_{l-1}).$
$G(u_l,v_l).O_{Op_l}(os_l,ps_l;u_l,v_l,u_{l-1},v_{l-1},is_{l-1},js_{l-1}).P_{Op_l}(is_{l-1},js_{l-1},u_{l-1},v_{l-1})$

n: $\Pi_{GO,n} = \sum_{(u_n,v_n)} \Pi_{GO,n}(u_l,v_l)$ where $\Pi_{GO,n}(u_n,v_n) =$
$\sum_{(u_{n-1},v_{n-1})} p_{Op_{A,n}}(u_n,os_n \mid u_{n-1},is_{n-1}).p_{Op_{C,n}}(v_n,ps_n \mid v_{n-1},js_{n-1}).\Pi_{GO,n-1}(u_{n-1},v_{n-1}).$
$G(u_n,v_n).O_{Op_n}(os_n,ps_n;u_n,v_n,u_{n-1},v_{n-1},is_{n-1},js_{n-1}).P_{Op_n}(is_{n-1},js_{n-1},u_{n-1},v_{n-1})$
$\Pi_{C,n} = \sum_{(u_n,v_n)} \Pi_{C,n}(u_n,v_n)$ where $\Pi_{C,n}(u_n,v_n) =$
$\sum_{(u_{n-1},v_{n-1})} p_{Op_{A,n}}(u_n,os_n \mid u_{n-1},is_{n-1}).p_{Op_{C,n}}(v_n,ps_n \mid v_{n-1},js_{n-1}).\Pi_{GO,n-1}(u_{n-1},v_{n-1}).$
$C_{Op_n}(u_1,v_1,os_1,ps_1;u,v,is_0,js_0).P_{Op_n}(is_0,js_0,u,v)$



Easy variations produce:

- Probabilities for stepwise simulation of *given* concrete fragment.
- Probabilities for stepwise simulation with unrestricted I/O.
- Probabilities for coarse grained retrenchments.
- etc.

Probabilistic analysis can be mapped over the various composition/decomposition mechanisms.

Applications: special purpose retrenchments

Toy case studies (couple of pages ... discrete ones *always* doable by refinement)

- Finite precision: sets/multisets → sequences , bounded arithmetic ...
- Simple resource control
- Gas turbine control
- Control theory

Larger case studies (several pages → entire paper → several papers)

- Feature interaction
- Mondex purse
- Comparative evaluation of graphics algorithms
- More control theory
- Radiotherapy dose calculations ... Other scientific apps. ...

Special purpose retrenchments (designed for further processing to some objective)

- Fault tree generation (with Marco Bozzano):
  retrenchment for any given component changes according to its place in the complete system
- Quantum retrenchment; obvious connections with probabilistic retrenchment

There may be several different retrenchments worth contemplating for the same pair of systems for different purposes. ... Fusion composition.

Tool support

Simon Fraser's M.Phil. project incorporated retrenchment PO generation into B-Core's B-Toolkit™ environment.

- It's not public domain.
- Many aspects of the B-Toolkit are optimised for a linear B-Method development.

Simon Fraser's Ph.D. project is to design and build an open source, flexible, extensible, toolkit for experimenting with retrenchment.

- ISO-Z (extended) used as the mathematical language
- A 'wrapper' syntax for building systems out of pieces of maths
- PO generation for a variety of theorem provers:
  PVS, B-Tool, Rodin, Isabelle, Vampire, KIV, ...
  Engineering of environment information ...
- Ultimately, a meta-language for expressing myriad composition techniques etc.

Methodology

The big question is:

'How does all the formal stuff fit with engineering practice?'

Two aspects:

1. How should individual ingredients of retrenchment impact practice?
2. How should practice impact the details of retrenchment?

Re. 1., as a start, practice can systematise checklists of things to oversee during periodic reviews of a development utilising retrenchment steps (the 'glass box'), eg.:

- Adequacy/coverage of the domain/range of the within relation,
- Adequacy/coverage of the domain/range of the output and concedes relations,
- Adequacy/coverage of the abstract and concrete operations ... termination/feasibility ... etc.

Re. 2., as a start, retrenchment ought to consider how its remit could be widened to take on board more of the development's non-functional requirements.

Retrenchment strives to make quantitative, the system engineering process as it is.

5. The Retrenchment Homepage

See the Retrenchment Homepage: Google knows where it is.

`http://www.cs.man.ac.uk/retrenchment`

- Bibliographical pointers.
- This tutorial.
- Etc.