Mechanized Support for Retrenchment

Simon Fraser and Richard Banach

Dept. of Computer Science, University of Manchester, M13 9PL
sfraser@cs.man.ac.uk, banach@cs.man.ac.uk

Retrenchment is a relatively new formal technique that extends the scope of traditional formal methods, allowing more systems to be modelled and developed rigorously. Retrenchment can be considered a more liberal version of refinement that allows an abstract specification to be formally related to a more concrete specification.

Consider two abstract machines abc and def (described using B notation below). Both models represent the addition of two natural numbers. The second, however, introduces the limitation that, when implemented on a computer, the set of natural numbers cannot be infinite. It is felt that both of these models are needed within a development. The first for use at the abstract level when describing system behaviour, and the second for use during development when an exact specification of the model is necessary.

MACHINE	abc	MACHINE	def
VARIABLES	aa, bb, cc	SEES	$Bool\_TYPE$
INVARIANT	$aa \in \mathbb{N} \land bb \in \mathbb{N}$	CONSTANTS	$upperbound$
	$\land\ cc \in \mathbb{N}$	PROPERTIES	$upperbound \in \mathbb{N}$
INITIALISATION	$aa := 0 \parallel bb := 1$		
	$\parallel cc := 2$		
OPERATIONS		OPERATIONS	
$my\_plus \mathrel{\widehat{=}}$	$aa := bb + cc$	$resp, dd \longleftarrow my\_plus(ee, ff) \mathrel{\widehat{=}}$	
		PRE	$ee \in \mathbb{N} \land ff \in \mathbb{N} \land$
			$ee \leq upperbound \land ff \leq upperbound$
		THEN IF	$ee + ff \leq upperbound$
		THEN	$dd := ee + ff$
			$\parallel resp := TRUE$
		ELSE	$dd := 0 \parallel resp := FALSE$
		END	
		END	
END		END	

The abstract machines abc and def.

A frequently used method of relating models in a formal development is refinement. In this instance, however, it is clear that refinement cannot be used to relate the models. Firstly the inputs and outputs of the operations differ, and more importantly there are circumstances where the behaviour of the models differs. Retrenchment allows the relationship between the models to be expressed formally through the introduction of two relations that together restrict the set of pre-conditions in which the relationship is valid, and extend the set of post-conditions. The first is known as the 'within' relation, and the second, the 'concedes' relation. The retrenchment abc_to_def describes the relationship between our two models.

RETRENCHMENT	abc_to_def
FROM	abc
TO	def
OPERATIONS	
RAMIFICATIONS	my_plus
WITHIN	$bb = ee \wedge cc = ff$
CONCEDES	$((resp = TRUE) \Rightarrow (dd = aa))$
	$\wedge\ ((resp = FALSE) \Rightarrow (dd = 0))$
END	
END	

The retrenchment relationship between the abstract machines abc and def.

To show that the retrenchment holds, it is necessary to show that, if it is possible to take a step in the abstract model – and that both the retrieves and within relations hold – then there is an equivalent step in the concrete model that establishes either the retrieves or concedes relation. This can be expressed formally as follows.[1]

$$R(u,v) \wedge W_{Op}(i,j,u,v) \wedge stp_{Op_c}(v,j,v',p) \quad \Rightarrow \quad (\exists\, u', i, o \bullet stp_{Op_a}(u,i,u',o) \wedge (R(u',v') \vee C_{Op}(u',v',o,p;i,j,u,v)))$$

In the above example the within relation strengthens the pre-condition such that the 'input' variables of each model's operation are equal. There is no retrieves relation in this instance (as def has no state variables) so the concedes relation must be established after every execution: this can be seen to hold through observation, if the *upperbound* is exceeded then there is no relationship between the values of the variables in the two models. If, however, the sum is less than or equal to the *upperbound* then the 'output' variables of each operation are equivalent.

Detractors of formal methods have long cited the complicated specifications – and their associated long-winded proofs – as a major disadvantage of their use. In response to this criticism, tools have been created that aid with both the specification, and proof of formally developed systems. These tools provide syntactic and semantic checking to reduce mistakes in specification, and provide automated theorem provers that are able to discharge large number of proofs. It is clear from existing criticism of formal methods that retrenchment is unlikely to be accepted, or used, unless mechanical support is provided to not only specify a retrenchment step, but to prove that the retrenchment relationship between two models is valid.

As such, an initial attempt was made to provide mechanized support for retrenchment in the B-Toolkit (a suite of software engineering tools that use the B-Method to specify and refine models). A formal extension to the B-Method was proposed, and implemented within the B-Toolkit, but only limited success was achieved. The structure of the toolkit, and its proof assistant, were such that the new constructs and proof obligations required for retrenchment were not easily introduced.

It was decided, therefore, to create a new tool where the focus is not on producing code from a specification, but on examining models, and the relationships between them. This tool would allow models (and relationships) to be specified using a combination of the Z notation (ISO-13568), and a simple wrapper language. Proof obligations will then be generated in a number of formats such that a selection of automated theorem provers can be used in their discharge. It is hoped that this tool will provide a practical aid to the advancement of retrenchment theory, automating the previously manual exercise of retrenchment proof.

[1] Where u, i, o and v, j, p are the state, inputs, and outputs of the abstract and concrete model respectively.