Unifying program refinement calculi

Jim Woodcock

Department of Computer Science
University of York
Heslington, York YO10 5DD, UK
Jim.Woodcock@cs.york.ac.uk

Hoare & He's *Unifying Theories of Programming* (UTP) is a research agenda that seeks to find a common basis in which to explain a wide variety of programming paradigms. The UTP technique is to isolate important language features, and give them a denotational semantics, so allowing different languages and paradigms to be compared. These semantic models are described in an alphabetised version of Tarski's relational calculus, presented in a predicative style that is reminiscent of the schema calculus in the Z notation.

The collection of these relations forms a *theory* of the paradigm being studied, and it contains three essential parts: an alphabet, a signature, and healthiness conditions. The *alphabet* is a set of variable names that gives the vocabulary for the theory being studied, chosen for any relevant external observations of behaviour. The *signature* gives the rules for the syntax for denoting objects of the theory. *Healthiness conditions* identify properties that characterise the theory. These theories are used as a semantic model for unified languages of specification and programming. Specifications are distinguished from programs only by the fact that the latter use a restricted signature. As a consequence of this restriction, programs satisfy a richer set of healthiness conditions.

The theory of *designs* introduces healthiness conditions that deal with the phenomenon of termination, and they form the basis for the study of pre and postcondition specifications, such as those found in B, VDM, and Z. Behaviours are described as certain relations between intial and final states, and many of the techniques of assertional reasoning and the laws of sequential refinement calculi can be captured in this theory.

The theory of reactive processes is disjoint from the theory of designs, and it describes processes that interact with their environment. The behaviours of these processes can no longer be expressed as relations between intial and final states; the behaviour in intermediate states is required as well. Indeed, many reactive processes have no final states at all. A sub-theory of reactive processes is the theory of CSP.

In this paper, we show that the sub-theory of CSP is exactly the reactive image of the theory of designs. This means that all CSP operators can be expressed as reactive designs; that is, as pre and postcondition specifications that are made to satify the reactive healthiness conditions. Using this insight, reactive processes can be subjected to assertional reasoning in a style familiar from sequential programming. We show how to construct a Hoare logic, a weakest precondition calculus, and a refinement calculus for CSP.