Specification and Verification I 2004 – Paper 7 Question 6 (MJCG)

(*a*) Annotations and verification conditions are both first order formulas. Annotations are inserted by the verifier (usually a human, but could be an AI program) at appropriate points of a program to provide formal documentation of the state whenever control reaches the point (e.g. to provide invariants for loops). Annotations are also used to provide variants for termination proofs. Verification conditions are generated algorithmically from annotated programs, and constitute a sufficient condition for the program to meet its specification.

(*b*) Partial correctness ensures a postcondition is achieved *if* a program terminates. Total correctness ensures both termination and achievement of a postcondition (i.e. Total corrrectness = Partial Correctness + Termination). The proof rules for partial correctness need to be augmented for total correctness by adding extra hypotheses for looping constructs to ensure termination (e.g. variants).

(*c*) Deep and shallow embedding are methods of encoding 'guest' languages in a 'host' logic. With deep embedding the syntax and semantics are both represented in the host logic, but with shallow embedding an external translation (not in the logic) compiles the guest language constructs into host language ones. Shallow embedding requires less to be formalised inside the host logic (i.e. neither the abstract syntax not the semantic function/relation need be formalised). The benefit of deep embedding is that one can reason about the syntax and semantics, because they are represented in the host logic. With shallow embedding one can only reason about the result of compiling individual phrases, not the compilation process (i.e semantics) itself. The deep/shallow distinction is really a spectrum as one can choose to deeply embed some constructs from a language and shallowly embed others.

(*d*) First order logic only allows variables to range over values from the 'universe of discourse'. Also functions and relations can only be applied to and return such values. Higher order logics allow variables to range over functions and relations, and functions and relations to take other functions and relations as arguments and return them as results. A standard example of a higher order formula is Mathematical Induction formalised with a variable ranging over predicates on numbers. First order logic is semi-decidable, but higher-order logic (except in special cases) is completely undecidable. Higher order logic is more expressive than first order logic: it can provide higher-level and more compact specifications.

This question refers to several parts of the course: (a) refers to the part on mechanising program verification, (b) to specification methods using Floyd-Hoare logic, (c) to semantic embedding of Floyd-Hoare logic in higher order logic and (d) to pure logic.

I will give full marks for a section if the candidate shows that he/she fully understands the topics. Fewer marks will be given if there are signs of confusion or incomplete knowledge.