This paper distinguishes several different approaches to organising a Weakest Pre-condition (WP) calculus in a theorem prover. The implementation of two of these approaches for Java within the LOOP project is described. This involves the WP-infrastructures in the higher order logic of the theorem prover PVS, together with associated rules and strategies for automatically proving JML specifications for Java implementations. The soundness of all WP-rules has been proven on the basis of the underlying Java semantics. These WP-calculi are integrated with the existing Hoare logic, and together form a verification toolkit in PVS: typically one uses Hoare logic rules to break a large verification task up into smaller parts that can be handled automatically by one of the WP-strategies.

MSC:

68N15 Theory of programming languages
68T15 Theorem proving (deduction, resolution, etc.) (MSC2010)

Keywords:
Java; Program correctness; Program logics; Weakest precondition

Software:
LOOP; JML; HOL; PVS; ESC/Java; KRAKATOA

Full Text: DOI

References:
[9] Gordon, M.J.C., Mechanizing programming logics in higher order logic, (), 387-489
[17] Jacobs, B., A formalisation of java’s exception mechanism, (), 284-301 · Zbl 0977.68579


[27] Poetzsch-Heffter, A.; Müller, P., A programming logic for sequential Java, (), 162-176


[29] Loop Project, Available from <~>


[31] Extended static checker ESC/Java, Compaq System Research Center, Available from <>

This reference list is based on information provided by the publisher or from digital mathematics libraries. Its items are heuristically matched to zbMATH identifiers and may contain data conversion errors. It attempts to reflect the references listed in the original paper as accurately as possible without claiming the completeness or perfect precision of the matching.