Abstracting and counting synchronizing processes.


Summary: We address the problem of automatically establishing synchronization dependent correctness (e.g. due to using barriers or ensuring absence of deadlocks) of programs generating an arbitrary number of concurrent processes and manipulating variables ranging over an infinite domain. Automatically checking such properties for these programs is beyond the capabilities of current verification techniques. For this purpose, we describe an original logic that mixes two sorts of variables: those shared and manipulated by the concurrent processes, and ghost variables referring to the number of processes satisfying predicates on shared and local program variables. We then combine existing works on counter, predicate, and constrained monotonic abstraction and nest two cooperating counter example based refinement loops for establishing correctness (safety expressed as non reachability of configurations satisfying formulas in our logic). We have implemented a tool (Pacman, for predicated constrained monotonic abstraction) and used it to perform parameterized verification for several programs whose correctness crucially depends on precisely capturing the number of synchronizing processes.

Keywords: parameterized verification; counting logic; barrier synchronization; deadlock freedom; multi-threaded programs; counter abstraction; predicate abstraction; constrained monotonic abstraction
doi:10.1007/978-3-662-46081-8_13