

**io-port 06068920**

**Kahlon, Vineet**

**Schedule insensitivity reduction.**

de Lara, Juan (ed.) et al., Fundamental approaches to software engineering. 15th international conference, FASE 2012, held as part of the European joint conferences on theory and practice of software, ETAPS 2012, Tallinn, Estonia, March 24–April 1, 2012. Proceedings. Berlin: Springer (ISBN 978-3-642-28871-5/pbk). Lecture Notes in Computer Science 7212, 99-114 (2012).

Summary: The key to making program analysis practical for large concurrent programs is to isolate a small set of interleavings to be explored without losing precision of the analysis at hand. The state-of-the-art in restricting the set of interleavings while guaranteeing soundness is partial order reduction (POR). The main idea behind POR is to partition all interleavings of the given program into equivalence classes based on the partial orders they induce on shared objects. Then for each partial order at least one interleaving need be explored. POR classifies two interleavings as non-equivalent if executing them leads to different values of shared variables. However, some of the most common properties about concurrent programs like detection of data races, deadlocks and atomicity as well as assertion violations reduce to control state reachability. We exploit the key observation that even though different interleavings may lead to different values of program variables, they may induce the same control behavior. Hence these interleavings, which induce different partial orders, can in fact be treated as being equivalent. Since in most concurrent programs threads are loosely coupled, i.e., the values of shared variables typically flow into a small number of conditional statements of threads, we show that classifying interleavings based on the control behaviors rather than the partial orders they induce, drastically reduces the number of interleavings that need be explored. In order to exploit this loose coupling we leverage the use of dataflow analysis for concurrent programs, specifically numerical domains. This, in turn, greatly enhances the scalability of concurrent program analysis.

doi:10.1007/978-3-642-28872-2\_8