

Master Thesis Defense

Speaker:	Xiangyu Gao
Supervisors:	Drs. R. Jayakumar, H. F. Li
Examining Committee:	Drs. D. Goswami, T. Radhakrishnan, Y. Yan (Chair)
Title:	Design and Implementation of a Partial-Order Semantics Based Runtime Verification Toolset for Distributed Java Programs
Date:	Monday, September 27, 2010
Time:	14:00
Place:	EV 1.162

ABSTRACT

Testing and debugging distributed programs are known to be difficult; an anomaly that occurred during one execution may not be reproducible at other executions. Traditional approaches in formal verification and testing have limited effectiveness. In this thesis, partial-order semantics is exploited to improve the effectiveness of runtime verification. Specifically, the execution of a distributed program is modeled in partial-order semantics. The specification contains two separate parts: the necessary ordering among abstract events (called atoms) and the correctness of outcome (change of data) of each atom. An atom is a high-level abstract event originated from the design space and subsequently mapped into the code space. With this event abstraction, an execution (run) of a distributed application can be compressed into an event space much smaller than those arising from individual program statements. The partial order model avoids property requirement and checking in every state of an execution. Instead, requirements are asserted over the ordering among events and the consequences of each event. This renders both the specification and the checking more direct and effective.

This thesis follows some theoretical work reported in [LM07] and implements a set of tools for the partial order model runtime verification. Besides application specific order requirements among atoms, detection of two generic classes of error is also provided. In coding, atomicity error may arise when the corresponding code blocks actually do not exhibit globally atomic effects. Serialization (mutual exclusion) is another generic requirement among some atoms which can be segregated and checked.

The toolset is built for a Java platform. The necessary ordering among atoms is specified in the form of an extended regular expression involving three operators: choice (+), sequence (;) and concurrency (||). The outcome of an atom is specified using a global predicate on data variables. Automatic code instrumentation is provided in augmenting the source code, according to the user specification of requirements. On-the-fly monitoring and checking of the specification are performed and relevant feedback is provided to the user.