Static analysis by abstract interpretation of concurrent programs in weakly consistent memories

Internship proposal, Master 2 MPRI, year 2013–2014

Supervisor:	Antoine Miné (`mine@di.ens.fr`)
Internship location:	Département d'informatique, École normale supérieure, Paris, France
Relevant course:	M2–6: Abstract interpretation: application to verification and static analysis

Note: other internships are possible on the topic of concurrent program analysis, static analysis, and abstract interpretation in general. Please contact the internship supervisor for more information.

Motivation

The goal of the internship is to extend an existing static analysis of concurrent programs by abstract interpretation [1] in order to support thread synchronization soundly and precisely in the presence of weakly consistent memories.

Many programs are now concurrent. Such programs are much more complex to verify than sequential ones. In first approximation, the execution of a concurrent program is an interleaving of the execution of its different threads ; this is called the sequential consistency model. The inherently non-deterministic scheduling of threads causes an explosion of the number of possible interleavings and makes regular analysis techniques intractable. To solve this problem, thread-modular techniques have been proposed: such methods analyze separately each thread as well as their interferences using different, communicating abstractions. The internship will focus on such a static analysis [1] developed in the scope of the AstréeA project [4] to check for run-time errors in embedded software.

An recent difficulty is the advent of weakly memory models, which are more relaxed that sequential consistency and make the analysis even more complex. In these models, different threads may hold inconsistent views of the memory (for instance, they can execute on processors with independent, loosely synchronized memory caches). In general, an analysis designed with sequential consistency in mind will not be sound for weaker memory models. The analysis in [1] is sound in such models only under restrictive conditions that limit either its generality or its precision. More precisely, shared variables must be protected by synchronization primitives that ensure the full memory consistency (such as mutual exclusion locks as provided by POSIX Thread libraries [3]) in order to use precise abstractions of interferences; for other shared variables, a coarse, flow-insensitive abstraction must be used.

The goal of the internship is to address this limitation and makes it possible to analyze soundly and precisely programs that use weaker forms of synchronization. These include the use

of atomic (or volatile) variables, the insertion of processor-specific instructions (such as barriers), or the reliance on specific properties enforced by the weak memory model (such as the "total store order" property of some popular models).

Expected work

The intern will develop new abstractions for interferences, including all the operators and transfer functions required by a thread-modular static analyzer such as [1]. The proposed domains and operators shall be proved correct with respect to a well-defined and realistic weakly memory model, one example being the popular Total Store Ordering for x86 [2]. The abstractions shall also be motivated by their usefulness in analyzing actual program fragments and by their tractable complexity. While not required, implementing and evaluating the domains experimentally is a plus.

The internship will take place in the context of the AstréeA Project [4] which can provide realistic fragments of industrial parallel programs.

References

[1] A. Miné. Static analysis of run-time errors in embedded critical parallel C programs. In *Proc. of the 20th European Symposium on Programming (ESOP'11)*, volume 6602 of *Lecture Notes in Computer Science (LNCS)*, pages 398–418. Springer, Mar. 2011. `http://www.di.ens.fr/~mine/publi/article-mine-esop11.pdf`.

[2] P. Sewell, S. Sarkar, S. Owens, F. Zappa Nardelli, and M. Myreen. x86-TSO: A rigorous and usable programmer's model for x86 multiprocessors. *Commun. ACM*, 53, 2010.

[3] IEEE Computer Society and The Open Group. Portable operating system interface (POSIX) – Application program interface (API) amendment 2: Threads extension (C language). Technical report, ANSI/IEEE Std. 1003.1c-1995, 1995.

[4] P. Cousot, R. Cousot, J. Feret, A. Miné, and X. Rival. The AstréeA static analyzer. `http://www.astreea.ens.fr`.