Memory Models

Overview Recent consensus on memory model is a *contract between the programmer and the system*: if the programmer writes disciplined (**data-race-free**) programs, the system will provide high programmability (**sequential consistency**), and performance.

A memory model describes how programs, which contains read and/or write instructions to the memory, interact with the memory. In particular, we are more interested in multi-threaded programs since a memory model is quite simple for single-threaded programs.

If a memory model is too **weak**, it cannot force safety and security properties of multi-threaded programs. If it is too **strong**, it may severely restrict the set of applicable optimzations (at many different levels; e.g. during compiler optimization, virtual machine runtime optimization, H/W out-of-order instruction execution).

Sequential consistency Sequential consistency specifies that *memory actions (reads/writes) will appear to execute one at a time in a single total order; actions of a given thread must appear in this total order in the same order in which they appear in the program* ***prior to any optimization***.

In other words, a set T of threads, where each thread $t \in T$ is an ordered sequence of instructions, ordering of all instructions in T so that it conforms to the *program order* is **sequentially consistent**.

Sequential consistency is **too restrictive** since it restricts the use of many compiler and processor optimizations. For example, even if there are no conventional data or control dependences between two memory instructions, sequential consistency may not permit reordering of these memory instructions.

Java memory model