COMP 525V: Modeling and Verifying Cache Coherence in ACL2

due: Apr 22, 2001; 11:59pm (electronically, using assignment name *acl2cache*.)

For this assignment, model and verify the cache coherence protocol described below (the protocol is the same as in the SMV assignment). A cache coherence protocol maintains consistent views of a central memory across distributed processors. Such a protocol is considered correct if for each address in each cache, the cached value is the same as the central memory value. This assignment is designed to allow you to contrast model checking and theorem proving on a common, non-trivial example.

The System Model

The distributed system model includes some number of processors, a global memory, and a shared bus. Each processor has its own cache which is connected to the global memory via a shared bus. A cache is a set of cache lines. Each cache line is associated with a memory address and contains two fields: a data value and a flag indicating whether or not the line is valid. When the flag is true, the data value should be consistent with the value in the corresponding address of the global memory.

Each processor can send read and write requests to its cache; these requests drive the protocol. The following rules govern system-wide responses to messages:

- When a cache receives a read request from its processor, it supplies the data value if the cache line for that address is valid. If the cache line is not valid, the cache fetches the contents from the global memory, stores the value, sets the cache line to valid. and returns the value to the processor.
- When the cache receives a write request from its processor, it puts a write request out on the bus and stores the new data value.
- Every cache snoops the bus. When a cache sees a write request from another processor, it invalidates its cache line for the requested address.
- The global memory shares the bus with the caches. When the global memory sees a write request on the bus, it stores the provided data value in the provided address. When the memory sees a read request on the bus, it supplies the stored value for the requested address.

Exercises

1. Develop an ACL2 model for the cache coherence protocol.
2. Provide a high-level (english) description of the theorems that you expect to need to prove consistency. Do this before you start trying to prove the system – you'll have a much greater chance of actual getting the proof to go through this way! Don't include minor lemmas about data structures in this description – stick to the theorems that frame a proof of cache coherence.
3. Verify that the model preserves consistency between the memory and the caches.
4. Comment on the differences between doing this exercise in ACL2 and SMV. How were the models different? How was the verification process different? If you had to verify a cache coherence protocol on the job, which approach would you take and why?

Notes

- This assignment is a bit challenging, and I suspect many of you won't completely finish it. This is why the high-level description is important – it will let me evaluate your work on how you thought out the proof, rather than what you actually got ACL2 to verify.
- Reminder that ACL2 has a facility for returning multiple values from a function definition. Look for *mv* in the documentation.

- Unlike in the SMV version, do not attempt to model arbitrary bus delays with your ACL2 model (unless you feel truly ambitious). Also, treat the bus as uni-directional, with messages going from the caches to the memory. Let the caches read from memory directly.
- Here's a suggestion for the overall structure of your "single-step" function:
    1. The *step* function takes a processor p, an action to perform (read/write, with the appropriate info), and the current state of the model.
    2. Perform the action on p's cache. Capture this as a function that returns the data that the processor requested, p's new cache, and the message that performing the action would send out on the bus.
    3. Return a new model state containing the data that the processor requested, the updated set of caches (for all processors) and the updated memory; both the caches and the memory should be updated according to the message that was sent on the bus in the previous step.
- Keep in mind the sketches of microprocessor proofs that we studied in class, when deciding how to set up both your model and your proof of cache coherence.