CS 410/510spec Homework 3 – Tuesday, February 9, 2016

This homework has two parts: a pencil&paper exercise to be turned in on paper in class, and a modelling exercise in TLA+, to be turned in by zipping up your `.tla` file and `.toolbox` directory and emailing them to `tolmach@pdx.edu`. Both are due at the **beginning** of class.

1. (From Lamport's *Specifying Systems* auxiliary material):

Determine which of the following formulas are temporal tautologies. For each one that isn't, give a counterexample.

```
(a) <>[]<>F <=> []<>F

(b) []<>[]F <=> []<>F

(c) ~[](F \/ G) => <>~F /\ <>~G

(d) []([]F => G) => ([]F => []G)

(e) [](F => []G) => ([]F => []G)

(f) [][A /\ B]_v <=> [][A]_v /\ [][B]_v

(g) []<><<A /\ B>>_v <=> []<><<A>>_v  /\ []<><<B>>_v

(h) [][A => B]_v <=> [][<<A>>_v => B]_v

(i) WF_v(A) /\ WF_v(B) => WF_v(A \/ B)

(j) SF_v(A) /\ SF_v(B) => SF_v(A \/ B)

(k) ENABLED (A /\ B) => (ENABLED A) /\ (ENABLED B)
```

2. Develop an alternative implementation of `InternalMemory` (`IM`), this time using write-back caches instead of the write-through caches `WTC` that we've been examining. As with `WTC`, your system should have the same external interface as `IM`. Your write-back caches should use the MSI protocol as described at `https://en.wikipedia.org/wiki/MSI_protocol`.

To proceed, make a new directory `MSI`, copy the `MemoryInterface.tla` and `InternalMemory.tla` files into it, and create a new file `MSI.tla` modeled on `WTC.tla`. As with `WTC`, you can save a little typing by making an `INSTANCE` of `InternalMemory` within `MSI`.

(a) To start with, make the following simplifying assumptions:

- The cache line size is one word, i.e. you never need to worry about reading or writing just part of a cache line. (This considerably simplifies some of the steps described in the Wikipedia article.)
- Don't use a queue to access the underlying "backing store" (main memory); instead, assume that it can be read and written directly using atomic operations (as in `IM`).

- Assume that the entire state of all caches can be read and/or written directly as part of a single atomic action (similar to how the `DoWr` action works in `WTC`). In other words, don't worry about how the "bus snooping" or "cache dirctories" needed to communicate between caches would actually work.

(b) Test your specification in three ways, analagous to what we did for `WTC`:

- By writing and checking a suitably modified version of the `Coherence` invariant.
- By writing and checking the same `LivenessProperty` as for `WTC`, assuming a suitably modified version of the `Liveness` specification.
- By refinement checking against `IM`, using an `INSTANCE` like the one in `WTC`, with a suitably modified refinement mapping.

Hint: start with your models very small!

(c) As a further challenge, once you have all aspects of this model working as expected, try to drop the last simplification listed in part (a) that allows atomic "actions at a distance" to change one processor's cache on behalf of another processor. Instead, try to model communication among the caches explicitly, e.g. by explicitly representing a bus.

As always, feel free to send me mail at `tolmach@pdx.edu` if you have any questions.