Functional Pearl: Four slot asynchronous communication mechanism

Matthew Danish

August 16, 2013

The four slot mechanism is a fully asynchronous method of communication from a writer to a reader process. A piece of data is made available through a shared memory pool. The writer may update the pool at any time without any coordination or synchronization with the reader. The reader may obtain a fully coherent, fresh piece of data from the pool at any time. The name derives from the need for a four-element array through which data is exchanged. The four slots are necessary to maintain the desired properties of asynchronicity, coherency, and freshness.

As part of the development of a new operating system, an implementation of the four slot mechanism was created in the dependently typed programming language, ATS, in conjunction with a partial proof of safety and correctness. We believe that this implementation is an example of a dependently typed "Functional Pearl" which demonstrates how easily dependent (and linear) types can be applied in practice to prove useful properties of low-level concurrent systems programming, while leaving no traces of runtime overhead.

Several properties of the various steps of the four slot mechanism are proven by hand, and then encoded into ATS types. Those types are given to the functions which comprise the building blocks of a four slot mechanism. Then two required safety theorems are encoded into the types of the most crucial steps in the mechanism. The ATS compiler checks the consistency of the theorems automatically, while also compiling the four slot mechanism into object code, ready to be linked into other programs. The static types are erased and do not affect the runtime performance of the mechanism.