Advanced Functional Programming

Software Engineering Chair and Programming Systems Lab

Small-group work

Questions for *Embedding an interpreted language using higher-order functions and types* by Norman Ramsey, Proceedings of the 2003 workshop on Interpreters, virtual machines and emulators, pages 6–14, 2003. Pick four questions and answer them.

1. What is the problem Ramsey wants to solve, what is his contribution? Also: what do you consider noise in this paper?

2. What are the main "impedance mismatches" between the host language Objective Caml and the embedded language Lua?

3. Explain the term *type-indexed functions* from the paper. How are types in Objective Caml and the glue-code combinators related?

4. Embedding always succeeds but projection may fail. Why? Which properties should hold for embedding/projection pairs?

5. What could an Objective Caml function embedded into Lua-ML use interpreter state for? Give some examples. How could such a function be made aware of the interpreter state? Take a close look at Figure 1 and especially the types there.

6. The following lines of Lua code define some variables that could reside in a configuration file where they are easy to change for a user. (In Lua, string concatenation is denoted by two dots.) How could these values be accessed from a function written in Objective Caml? Sketch an implementation that uses **getglobal**:$state \rightarrow value \rightarrow value$ to access global Lua variables.

   ```
   prefix = "/var"
   log    = prefix .. "/log/myapp.log"  -- log files
   pid    = prefix .. "/run/myapp.pid"  -- process ID
   ```

7. What don't you like about the paper?

Homework Assignment

1. Read *The Zipper* by Gérard Huet, Journal of Functional Programming, 7(5), pages 549–554, September 1997.

2. Summarize the paper *in your own words* on one page. Put your name and student ID on your summary and drop off a printout at office 326/45 until Monday, December 12th at noon (12 am). If the door is closed, slide your printout under the door. No Emails.