Discussion questions for *ESP: A Language for Programmable Devices*

Comp 150PLD

September 22, 2015

1 Warm-ups

1. What is the domain of ESP?
2. What is a programmable device?
3. What are event-driven state machines? How do they work? Why are they used to program firmware? Why are they difficult to get right in C code?
4. What are the goals of ESP?
5. Explain what the code in Appendix B does.

2 Design Evaluation

1. What features does ESP provide? How could you tell if these features are sufficient?
2. What features of the language contribute to program brevity and/or modularity and why?
3. What features of the language (or lack thereof) facilitate efficient compilation?
4. What features (or lack thereof) of the language facilitate verification and debugging and why?
5. Describe how ESP is implemented and why.
6. How did the authors evaluate their language in terms of 1) the language design itself, 2) the verification tools, and 3) the efficiency of the language? How could these evaluations be improved?

3 Evaluating ESP as a Domain-Specific Language

1. What are the advantages and disadvantages of adopting a C-like syntax?
2. Does ESP have or could it benefit from ESP-specific tool support? Explain.
3. Describe ESP's type system. How does it contribute to the goals of the language? Is the language type safe? Why or why not?
4. Does ESP have a runtime system? If so, what does it do?
5. Does ESP have or could it benefit from ESP-specific libraries? Explain.
6. How might the design be improved?
7. Discuss to what extent ESP is a DSL.

4 More detailed questions.

These might help you answer the questions above or guide your understanding of the paper.

1. Be able to explain what the various ESP code fragments mean.
2. What does the $ indicate?
3. Why are only immutable values allowed to be sent over channels?
4. Explain why the ESP compiler doesn't have to copy objects sent as messages over channels.
5. According to the authors, why are processes a more appropriate abstraction for writing concurrent code than functions?
6. Why are link and unlink possible sources of memory unsafeness?
7. What are the advantages and disadvantages of using channels as the external interface?
8. What is SPIN and how does ESP use it to discover bugs?
9. Why does ESP translate a high-level representation of ESP programs into SPIN rather than a lower-level version that arises through the ESP compilation process?
10. Why does the ESP compiler perform standard compiler optimizations like dead-code elimination when the resulting C code is going to be passed to a C compiler?
11. Explain why the ESP team didn't implement channels as queues on which writers wait.