Rely/Guarantee-thinking and Separation Logic

Viktor Vafeiadis and Cliff Jones

MPI Kaiserslautern and Newcastle University

FM-2011 Tutorial
Limerick, Ireland
2011-06-21

Closing remarks (CBJ)

Simpson’s 4-slot (ACM) implementation

- *Asynchronous* Communication Mechanisms
  - essence is to avoid races on (four) slots
  - . . . and guarantee to deliver “freshest” element
- [JP11]
  - uses abstraction to constrain clashes
  - spec uses “;” in two repeating processes
  - . . . and “fiction of atomicity”
  - first design step reduces needed number of cells (≥ 3)
  - . . . nice new notion $\overparen{\mathit{fresh}\text{-}w}$
  - second step shows 4 slots suffice and allow communication
- R/G at abstract level although no races at concrete level
- Many have studied
  - Bornat: uses SL — adds R/G (not as in RGSep?)
  - Wang & Wang: exchange of ownership (but not freshness)

Some open questions (i)

- "auxiliary variables" (aka "ghost variables")
  - suspicion they mark abstraction failure (cf. [Jon10])
  - (as in data reification) — at least there: a precise test
- link between "linearisability" and "splitting atoms safely"?
- what does a compact notation buy in larger applications?
  - Carroll Morgan's frames where pre/post can be many lines
  - (heavy) framing notation in VDM (keywords)
  - . . . vs SL
- care with "statements" in triples
  - in development, middle of a triple is the name of something to be refined
  - VDM rarely worries about "axiom of assignment"
- partial ("conditional") vs. total correctness

Some open questions (ii)

- tool support
  - (probably) essential for wide scale deployment
  - but can have a constraining effect on thought!
- R/G built around “stack” (normal) variables
  - (SL around heap)
  - VDM would cope with heap ownership as restrictions of array
  - cf. $heap(p)$ in Viktor’s Part 1
- my aim:
  - get “under the skin of” R/G & SL
  - “back to basics” — real issues — alternative ways of tackling

References

C. B. Jones, D. Lomet, A. Romanovsky, and G. Weikum.
The atomic manifesto.
Journal of Universal Computer Science, 11(5):636–650, 2005.

C. B. Jones.
Splitting atoms safely.
Theoretical Computer Science, 375(1–3):109–119, 2007.

C. B. Jones.
The role of auxiliary variables in the formal development of concurrent programs.
In Cliff B. Jones, A. W. Roscoe, and Kenneth Wood, editors, *Reflections on the work of C.A.R. Hoare*, chapter 8, pages 167–188. Springer, 2010.

Cliff B. Jones and Ken G. Pierce.
Elucidating concurrent algorithms via layers of abstraction and reification.
Formal Aspects of Computing, 23(3):289–306, 2011.