Modelling Recursion in Event-B

Stefan Hallerstede

University of Düsseldorf, Germany

We present a method for modelling recursion using Event-B [1]. The central concepts of an Event-B model are *machines*, *events* and *invariants*: Events change the state of a machine "preserving the invariants", where the latter statement has to be proved. This simple approach has been shown to be suitable for a range of modelling problems from sequential programs to distributed systems.

We find that when modelling more complex sequential programs, we state invariants in a particular form:

$$pc = pos \;\Rightarrow\; \text{“Property-at-position } pos\text{”}, \tag{1}$$

where pc is an abstract program counter and pos a position in the program text. This means we adopt an assertion-style approach to modelling [3] for sequential programs . We can lift Event-B refinement to assertion-style modelling with the central concepts of *machines*, *events* and *assertions*. This permits us to use Event-B for proving properties about such models using the simple correspondence (1).

In assertion-style Event-B we specify programs as relations on program variables v in the form of a proof outline, e.g.,

$$P \to e \to Q, \tag{2}$$

where P and Q are assertions and e is an event. The outline (2) states that starting from assertion P event e establishes assertion Q. The resulting approach of program development resembles that of traditional program verification [2]. In this approach, we refer to a proof outline as a *machine*.

Recursion can now simply be modelled by instantiating a machine, say, M in some refinement of M. The resulting approach mixes ideas of the refinement calculus [4] and program verification. In the talk we present some examples and discuss some methodical issues.

References

1. Jean-Raymond Abrial. *Modeling in Event-B: System and Software Engineering.* Cambridge University Press, 2010.
2. K. R. Apt, , F. S. de Boer, and E.-R. Olderog. *Verification of Sequential and Concurrent Programs.* Graduate Texts in Computer Science. Springer, 3rd edition, 2009.
3. L. Lamport. *Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers.* Addison-Wesley, 2002.
4. Carroll C. Morgan. *Programming from Specifications: Second Edition.* Prentice Hall, 1994.