6 Asynchronous Multi-Agent ASMs

In this chapter the single-agent (basic or turbo) ASMs of Chaps. 3 and 4 and the multi-agent synchronous ASMs of Chap. 5 are extended to asynchronous multi-agent ASMs and shown to be useful for the design and the analysis of distributed systems. In Sect. 6.1 we define *async ASMs* and illustrate them by characteristic distributed network algorithms (for consensus, master–slave agreement, leader election, phase synchronization, load balance, broadcast acknowledgment) and a position-based routing protocol for mobile ad hoc networks. In Sect. 6.2 we show async ASMs at work in a requirements capture case study for a small embedded system (Light Control). In Sect. 6.3 we use async ASMs to model and analyze two time-constrained algorithms which support fault tolerance for a distributed computing service, namely in Sect. 6.3.1 the modem and network communication protocol *Kermit* for correct file transfer, well-known from TCP/IP installations, and in Sect. 6.3.2 a Processor Group Membership protocol. In Sect. 6.4 we use the ASM refinement method to show – adopting Lamport’s famous mutual exclusion algorithm *Bakery* as an example – how time-constrained algorithms with “atomic actions” can naturally be turned in a provably correct way to reflect also the “real-time duration” of actions. We show that it suffices to refine the global state view of atomic non-overlapping reads and writes in shared registers to a local state view of single agents whose overlapping reads and writes to the same location are governed by the constraints that async ASM runs impose on controlled, monitored and shared locations. Section 6.5 deals with event-driven ASMs. As a concrete illustration we model in Sect. 6.5.1 event-driven UML activity diagrams by async ASMs with turbo components and apply them for a compact one-page definition of an interpreter for Occam programs.

This chapter can be read independently of the preceding Chaps. 3–5 and most of Chap. 2; it suffices to know the definition of basic ASMs, of the ASM refinement concept and, for the examples in Sect. 6.5.1, of the notion of turbo ASMs.

---

1 Lecture slides can be found in *AsyncASM* (→ CD), *LightControl* (→ CD), *LightControlRequirements* (→ CD), *Kermit* (→ CD), *GroupMemberProtocol* (→ CD), *Bakery* (→ CD).

---

E. Börger et al., *Abstract State Machines*
© Springer-Verlag Berlin Heidelberg 2003
6.1 Async ASMs: Definition and Network Examples

The monitored and shared locations and functions in basic ASMs abstract from detailed modeling of the actions of the environment, thus supporting for basic ASMs the characteristic splitting of the dynamics of a system into a machine computation part and a part which describes, in a possibly declarative manner, the assumed environment properties. For a computation step of a basic ASM to happen, all locations are supposed to have well-defined values. In fact Definition 2.4.22 incorporates the environment-controller separation principle explained in Sect. 5.2.1, in the sense that the value changes for monitored locations are assumed to take place in such a way that the new value is stable each time the machine is going to perform a step. The changes of monitored locations can be viewed as resulting from “monitored” moves of “unknown” environment agents, made independently of the machine moves, which are “controlled” by the executing agent, but synchronized with the machine moves as happening either simultaneously with them or “between successive” ones.

The definition of async ASMs generalizes this situation to an arbitrary finite number of independent agents, each executing a basic or structured ASM\(^2\) in its own local state. A problem to solve for runs of such asynchronously cooperating agents originates in the possible incomparability of their moves which may come with different data, clocks, moments and duration of execution. This makes it difficult to uniquely define a global state where moves are executed to locate changes of monitored functions in an ordering of moves. The coherence condition in the definition of asynchronous multi-agent ASM runs below postulates well-definedness for a relevant portion of state in which an agent is supposed to perform a step, thus providing a notion of “local” stable view of “the” state in which an agent makes a move. The underlying synchronization scheme is described using partial orders for moves of different agents which reflect causal dependencies, determining which move depends upon (and thus must come “before”) which other move. This synchronization scheme is as liberal as it can be, restricted only by the consistency condition for the updates which is logically indispensable, and thus can be instantiated by any consistent synchronization mechanism.

**Definition 6.1.1 (Asynchronous multi-agent ASM).** An async ASM is given by a family of pairs \((a, ASM(a))\) of pairwise different agents, elements of a possibly dynamic finite set Agent, each executing its basic or structured ASM \(ASM(a)\). A run of an async ASM, also called a partially ordered run,\(^3\)

\(^2\) For the definition of async ASMs it is convenient to consider sync ASMs as given by a one-agent ASM where the synchronized subagents are viewed as a team. This allows one to let the behaviorally relevant causal dependencies and the different clocks of independent agents stand out distinctly, separate from the modularity relevant distinction of the substates of each (asynchronous or synchronous) agent.