5 Synchronous Multi-Agent ASMs

In this chapter\(^1\) the single-agent ASMs of Chaps. 3, 4 are extended to multi-agent synchronous ASMs (\textit{sync} ASM) which support modularity for the design of large systems. We illustrate this by sync ASMs for two popular benchmark case studies for the verified design of reactive control systems: a controller for the Production Cell [323] (Sect. 5.1), solving a typical industrial plant control problem, and a real-time gate controller for the Generalized Railroad Crossing [277] (Sect. 5.2), both controllers coming with to-be-verified safety, liveness and performance requirements. Although the case studies are really small (leading to roughly 1 K lines of controller code), they allow us to explain how to apply practically useful software architecture principles to modularize systems, starting from ground model ASMs and leading to verified code. The chapter can be read independently of the preceding Chaps. 4 and 3 and most of Chap. 2; it suffices to know the definition of basic ASMs and of ASM refinements.

A multi-agent synchronous ASM is defined as a set of agents which execute their own basic or turbo ASMs in parallel, synchronized using an implicit global system clock. Semantically a sync ASM is equivalent to the set of all its constituent single-agent ASMs, operating in the global states over the union of the signatures of each component. Examples \textit{par excellence} are offered by programs in synchronous programming languages [266] where runs are totally ordered sets of “logical instants” at which “events” occur (read: sets of simultaneous occurrences of possibly value-carrying signals through which the programs communicate among themselves and with the environment) to which all subprocesses of the executed program react instantaneously. The sequence of events determining a run is the sequence of states forming the run of the underlying multi-agent synchronous ASM, where the global clock tick (a built-in signal which is supposed to be present in every event) plays the role of a step counter.

The practical usefulness of sync ASMs derives from the possibility of equipping each agent with its own set of states and rules and of defining and analyzing the interaction between components using precise interfaces over common locations. To denote the instance of a function \(f\) for an agent \(a\) we write \(a.f\) and often omit mentioning \(a\) when it is clear from the context.

\(^1\) Lecture slides can be found in \textbf{ProdCell} (\(\sim\) CD), \textbf{GateController} (\(\sim\) CD).
5.1 Robot Controller Case Study

This section reviews the ASM solution [120, 332] for the Production Cell control problem [323], which together with the ASM solution [43] for the frequently used Steam Boiler case study [9] constituted the first explicit test of the integratability of the ASM method into the various phases of an industrial software development cycle (see Sect. 9.4.1 for historical details). The declared goal was to cover the following major development steps (see the V-scheme levels in Fig. 2.3):

1. elicit the given requirements by capturing them into a ground model ASM, inspectable by an application-domain expert,
2. stepwise refine this abstract model to executable (in this case C++) code whose module structure reflects the application-domain-driven component architecture of the ground model,
3. mathematically verify the required safety, performance and liveness properties (which were model checked and PVS-verified in [424, 362, 207]),
4. validate the code by extensive experimentation with the production cell simulator built at the FZI in Karlsruhe,
5. provide for maintenance purposes a transparent and complete documentation of the design (which in fact was submitted as an inspection case study to a Dagstuhl seminar on “Practical Methods for Code Documentation and Inspection” [117]).

Since in this section a production cell controller is developed to illustrate how sync ASMs enhance the modularity of specifications and their implementations, we concentrate here upon the design – ground model construction and its refinement\(^2\) – and refer the reader to the above-mentioned publications concerning the verification, validation and documentation aspects for which this book presents other more challenging examples. One example is in Sect. 5.2.2, which explains the verification of a real-time controller (there the design task is pretty obvious).

5.1.1 Production Cell Ground Model

We start by extracting from the task description [323] what are the agents and the basic objects, operations and interactions of the system.

... the production cell is composed of two conveyor belts, a positioning table, a two-armed robot, a press, and a traveling crane. Metal plates inserted in the cell via the feed belt are moved to the press. There, they are forged and then brought out of the cell via the other belt and the crane.

\(^{2}\) For a further analysis of the compositional aspects of the production cell ASM in terms of its submachines see [353].