5

Dags and Asynchronous Cellular Automata

In this chapter, we present asynchronous cellular automata (with types) as a general model of a distributed system. It subsumes the models of finite automata, asynchronous automata, and communicating finite-state machines, as introduced in Chaps. 4, 6, and 8, respectively. Nevertheless, it allows a characterization in terms of EMSO logic. The underlying domain comprises dags over a distributed alphabet, which describes the dependencies of actions provided by the system.

5.1 $(\widetilde{\Sigma}, C)$-Dags

In this section, we study classes of structures that might be appropriate to describe and model the behavior of a distributed system. We hereby come from quite general structures, which, depending on the kind of system at hand, are refined towards more specific ones.

Our starting point will be the class $\mathbb{DAG}$ of directed acyclic graphs, which generate partial orders in a natural manner. Recall that the nodes of a graph can then be seen as events, which are executed in the order imposed by the edge relation. They are partially ordered rather than queued in a total order to abstract from a concrete ordering of intrinsically independent events. However, arbitrary partial orders or, rather, their associated graphs, might be too general to model behaviors with special characteristics. For example, if we deal with finite automata as a model of a system, it suffices to consider only those graphs that represent totally ordered sets or words (cf. Chap. 4). Moreover, if the system at hand is designed for sending messages between finitely many processes each of which is sequential, only those graphs come into question whose edges reflect either a message exchange or neighboring events executed by one and the same sequential process. As the distributed behavior of a concurrent system gives rise to events that cannot be ordered, we shall furthermore exclude graphs provoking such an unnatural ordering of events. While the former model, a message-passing system, will lead us to

message sequence charts, the latter refers to *Mazurkiewicz traces*. As we will see, message sequence charts and Mazurkiewicz traces are incomparable in general, but, in some special cases, can be embedded into each other.

We fix a nonempty finite set Ag of at least two *agents*, a *distributed alphabet* $\widetilde{\Sigma}$, which is a tuple $(\Sigma_i)_{i\in Ag}$ of (not necessarily disjoint) alphabets Σ_i, and an alphabet C. In the following, let Σ stand for $\bigcup_{i\in Ag}\Sigma_i$, the set of *actions*. Elements from Σ_i are understood to be actions that are performed by agent i. So let, for $a \in \Sigma$, $loc(a) := \{i \in Ag \mid a \in \Sigma_i\}$ denote the set of agents that are involved in a. Having this in mind, we say that actions a and b are *independent* and write $a\ I_{\widetilde{\Sigma}}\ b$ if there is no common agent that controls both of them, i.e., if $loc(a) \cap loc(b) = \emptyset$. Otherwise, we say a and b are *dependent*, writing $a\ D_{\widetilde{\Sigma}}\ b$. Observe that $D_{\widetilde{\Sigma}}$ is a reflexive and symmetric binary relation on Σ. Such a relation is called a *dependence relation* over Σ. Moreover, we call the pair $(\Sigma, D_{\widetilde{\Sigma}})$ a *dependence alphabet*.

Agents may communicate with one another. They may do this by executing common actions simultaneously or via message exchange through channels. So let us denote by $Ch(Ag)$ (or just Ch) the set $\{(i,j) \in Ag \times Ag \mid i \neq j\}$ of *channels*. Thus, we assume that a "real" message exchange takes place between distinct processes only.

We now introduce the models representing the behavior of a system of communicating agents. In doing so, we combine the standard models of [29] and [55].

Definition 5.1 (Lo-Dag). *A* lo-dag *over the pair* $(\widetilde{\Sigma}, C)$ *is a structure* $(V, \{\lhd_\ell\}_{\ell\in C}, \lambda) \in \mathbb{DAG}(\Sigma, C)$ *such that, for any* $i \in Ag$, $\lambda^{-1}(\Sigma_i)$ *is totally ordered by* $\leq$.

Thus, the only restriction imposed by a lo-dag compared with a dag is the totally ordered behavior of a single process, which is thus assumed to be sequential. If, in addition, we restrict the communication between agents in the way that messages (edges) of equal type cannot cross, we obtain the class of $(\widetilde{\Sigma}, C)$-dags, which will be the main model considered in this chapter.

Definition 5.2 ($(\widetilde{\Sigma}, C)$-Dag). *A* $(\widetilde{\Sigma}, C)$-dag *is a lo-dag* $(V, \{\lhd_\ell\}_{\ell\in C}, \lambda)$ *over* $(\widetilde{\Sigma}, C)$ *such that, for any* $\ell \in C$ *and any* $(u,v), (u',v') \in \lhd_\ell$ *with* $\lambda(u) = \lambda(u')$ *and* $\lambda(v) = \lambda(v')$, *we have* $u \leq u'$ *iff* $v \leq v'$.

We conclude that, in a $(\widetilde{\Sigma}, C)$-dag $(V, \{\lhd_\ell\}_{\ell\in C}, \lambda)$, for any $u \in V$, $\ell \in C$, and $a \in \Sigma$, there is at most one vertex $v \in V$ such that both $u \lhd_\ell v$ $(v \lhd_\ell u)$ and $\lambda(v) = a$. If we require that *any* two messages between two agents must not cross, we deal with the class of fifo-dags over $(\widetilde{\Sigma}, C)$.

Definition 5.3 (Fifo-Dag). *A* fifo-dag *over the pair* $(\widetilde{\Sigma}, C)$ *is a lo-dag* $(V, \{\lhd_\ell\}_{\ell\in C}, \lambda)$ *over* $(\widetilde{\Sigma}, C)$ *such that, for any* $(u,v), (u',v') \in \lhd$ *with* $\lambda(u)\ D_{\widetilde{\Sigma}}\ \lambda(u')$ *and* $\lambda(v)\ D_{\widetilde{\Sigma}}\ \lambda(v')$, *we have* $u \leq u'$ *iff* $v \leq v'$.