Chapter 4
Supervisory Control with Partial Observations

Jan Komenda

4.1 Introduction

This chapter presents supervisory control of partially observed discrete-event systems represented as finite automata. In engineering systems it is often not realistic to assume that all events are observable. For instance, some hidden (internal) actions or failures are typically not directly observable. Sometimes it is simply too costly to install the necessary equipment (sensors) to observe all events that are occurring in the system. The partial observations are not only due to lack of some of the sensors but due to economic reasons, it is too costly to install the sensors.

Therefore, supervisory control theory has been extended to deal with systems, where only a part of the events that occur in the system is observable. Two independent papers were the first to treat the important problem of supervisory synthesis with partial observations: [6] and [10].

A closed-loop system under partial observations is defined using a supervisory feedback map that specifies the enabled actions (events) after a string of (observable) events has been observed.

Due to partial observations strings that are not distinguishable by the observations require the same control action. There are two possible control laws that satisfy this requirement: one is called permissive and the other antipermissive.

Partial observation about the state of the system is encoded by the observer automaton defined over the observable alphabet. These are helpful for implementing supervisory control laws and for many algorithms solving supervisory control problems.

An additional property, called observability, is required of a specification language to be achievable as the language of the closed-loop system (no matter which
of the two policies is chosen) in addition to controllability and relative closedness conditions (known as $L_m(G)$-closedness) needed in the case of complete observations. Basic facts about observability will be presented together with the main existential result, the existence theorem of a supervisor in terms of observability, and controllability conditions. Since observability is not in general preserved by language unions, a stronger notion called normality of a language will be described.

Finally, algorithms for checking observability and normality, and for computing supremal normal sublanguages and computing supremal normal and controllable sublanguages will be presented. The algorithm for checking observability will be of polynomial complexity, while the ones for constructing supremal sublanguages are of exponential complexity. This is because they require an observer automaton. Moreover, the representation of the specification language must have a special property called the state-partition automaton. Such a representation requires that states with indistinguishable past are partially disambiguated in the sense that different states of the observer do not overlap. For any finite automaton such a state-partition automaton always exists as a finite automaton.

Both the language-based (behavioral) [17] and the state-based framework [14] have been developed for supervisory control with partial observations. Since both frameworks are equivalent, the state-based approach does not add much to the originally developed language framework and there exist bidirectional transformations among them, only the language framework is presented in this chapter.

The concepts and results presented in this chapter will be illustrated by simple examples.

4.2 Concepts of Supervisory Control with Partial Observations

The notation for (deterministic) generators $G = (Q, E, f, q_o, Q_m)$ and controlled generators (CDFG $G_c$) is the same as in the previous chapter on supervisory control with complete observations. In addition, it is assumed that the set of events $E$ is partitioned into the disjoint union of observable events (denoted $E_o$) and unobservable events (denoted $E_{uo}$), i.e. $E = E_o \cup E_{uo}$. The concept of (natural) projection is associated with the observable event subset as the morphism of monoids $P : E^* \to E_o^*$ defined by

$$P(\varepsilon) = \varepsilon \text{ and } P(se) = \begin{cases} P(s)e, & \text{if } e \in E_o, \\ P(s), & \text{if } e \in E_{uo}. \end{cases}$$

The morphism property ensures that natural projection is catenative, i.e. $P(uv) = P(u)P(v)$ for any $u, v \in E^*$. The projection is extended to languages in a natural way, for $L \subseteq E^*$: $P(L) = \bigcup_{w \in L} P(w)$.

Given a set $Q$ we use the notation $Pwr(Q)$ to denote the set of all subsets of $Q$ including the empty set. Since the projection is not injective, the inverse projection