Controllability for Nondeterministic Finite Automata with Variables

Jasen Markovski
Department of Mechanical Engineering, Eindhoven University of Technology, Den Dolech 2, 5612MH, Eindhoven, The Netherlands

Keywords: Supervisory Control Theory, Controllability, Finite Automata with Variables, Partial Bisimulation.

Abstract: Supervisory control theory deals with automated synthesis of models of supervisory controllers that ensure safe coordinated discrete-event behavior of a given system. To increase the expressivity of the framework and provide for a greater modeling convenience, several extensions with variables have been proposed. One of the most prominent such extensions is implemented by means of extended finite automata with variables. We revisit the notion of controllability for nondeterministic finite automata with variables, which defines conditions under which a model of a supervisory controller can be synthesized. We will show that the existing notion of controllability for extended finite automata does not have desirable algebraic properties, i.e., it is not a preorder. We propose to employ an extension of controllability for nondeterministic discrete-event system based on a behavioral relation termed partial bisimulation, which we show that subsumes the existing notion of controllability for extended finite automata.

1 INTRODUCTION

Development of quality control software is becoming an increasingly difficult task due to high complexity of high-tech systems, promoting the former as an important bottleneck in the design and production process as already noted in (Leveson, 1990). Traditional techniques are not able to satisfactorily cope with the challenge due to the frequent design changes in the control requirements, which gave rise to supervisory control theory of discrete-event systems postulated in (Ramadge and Wonham, 1987; Cassandras and Lafortune, 2004). Supervisory control theory studies automatic synthesis of models of supervisory control software that provide for safe and nonblocking behavior of the controlled system by coordinating high-level discrete-event behavior of the concurrent system components.

Supervisory controllers rely on discrete-event observations made regarding the discrete-event system behavior by using sensory information, as depicted in Figure 1. Based upon the observed signals, these controllers decide which activities are allowed to be carried out safely and do not lead to potentially dangerous or otherwise undesired situations, and send back control signals to the hardware actuators. Under the assumption that the supervisory controller can react sufficiently fast on machine input, one can model this *supervisory control feedback loop* as a pair of synchronizing processes in line with (Ramadge and Wonham, 1987; Cassandras and Lafortune, 2004). The model of the uncontrolled system is typically referred to as *plant* and it is restricted by the model of the supervisory controller, which referred to as *supervisor*. The coupling of the supervisor and the plant, results in the *supervised plant*, which models the supervisory control loop, i.e., it specifies the behavior of the controlled system.

Traditionally, the activities of the machine are modeled as discrete events, whereas the supervisor is a process that synchronizes with the plant. The supervisor can enable or disable available events in the plant by synchronizing or not synchronizing with them, respectively. The events are split into *control-*

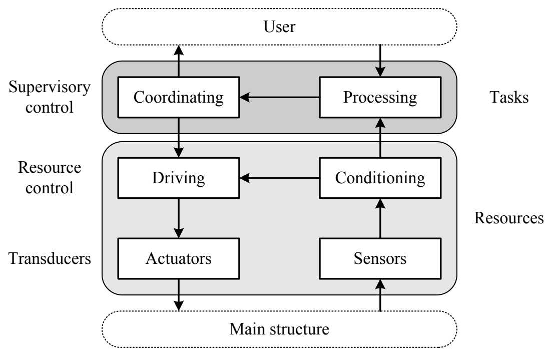

Figure 1: Supervisory control architecture.

Markovski J..
Controllability for Nondeterministic Finite Automata with Variables.
DOI: 10.5220/0004430604380446
In *Proceedings of the 8th International Joint Conference on Software Technologies* (ICSOFT-PT-2013), pages 438-446
ISBN: 978-989-8565-68-6

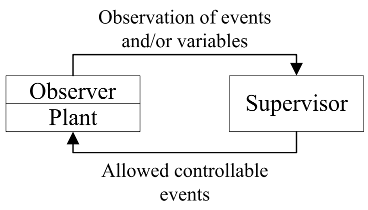
Observation of events and/or variables
Observer
Plant
Supervisor
Allowed controllable events

Figure 2: Supervisory control feedback loop with data-based observations.

lable and *uncontrollable events*, the former typically modeling interaction with actuators, whereas the latter model observation of sensory information. Therefore, the supervisor is allowed to disable controllable events, e.g., if the boiler pressure is above the safe threshold, then the heater should be switched off, but it is not allowed to disable any available uncontrollable events, e.g., by ignoring the pressure sensor of the boiler, one reaches a potentially dangerous situation.

Additionally, the supervised plant must also satisfy a given set of *control requirements*, which model the safe or allowed behavior of the machine. Furthermore, it is typically required that the supervised plant is nonblocking, meaning that it comprises no deadlock and no livelock behavior. To this end, every state is required to be able to reach a so-called *marked* or final state, following the notation of (Ramadge and Wonham, 1987; Cassandras and Lafortune, 2004), which denotes the situation that the plant is considered to have successfully completed its execution. The conditions that define the existence of such a supervisor are referred to as (nonblocking) *controllability* conditions. In the setting of this paper we will not consider in detail the process of modeling and ensuring that the (nonblocking) control requirements hold for the given plant and, instead we refer the reader to the model-based engineering framework of (Schiffelers et al., 2009; Markovski et al., 2010).

Depending on the observational power of the supervisor, we deal with event-based supervision, studied in (Ramadge and Wonham, 1987), state-based supervision as studied in (Ma and Wonham, 2005; Markovski et al., 2010), or data-based supervision along the lines of (Miremadi et al., 2008; Markovski, 2012b), respectively. The first approach relies on building a history of observed events to deduce the state of the system as suggested in (Cassandras and Lafortune, 2004), whereas the second and the third approaches employ observers and guards that directly convey the state of the system to the supervisor in the vein of (Ma and Wonham, 2005; Markovski, 2012b), as depicted in Figure 2. With respect to the control architecture of Figure 1, the second and the third approach suggest that the interface between the layers of resource and supervisory control is unified, e.g., by employing shared variables or publisher/subscriber services, which is typical for implementations in the artificial intelligence domain. The event-based approach suggests direct observation of activities of the system, which are typically triggered by the system to be supervised, relying on some input/output interface. The extensions of supervisory control theory with variables and data aim at a two-fold improvement: more concise specification due to parametrization of the systems, as suggested in (Chen and Lin, 2000; Miremadi et al., 2008) and greater expressiveness and modeling convenience, as shown in (Skoldstam et al., 2007; Gaudin and Deussen, 2007). The extensions range over the most prominent models of discrete-event systems like finite-state machines developed in (Chen and Lin, 2000), labeled transition systems, considered in (Markovski, 2012b), and automata extensions, provided in (Skoldstam et al., 2007; Gaudin and Deussen, 2007).

With the development of new models, the original notion of controllability for deterministic discrete-event systems of (Ramadge and Wonham, 1987; Cassandras and Lafortune, 2004) is subsequently extended to the corresponding settings with variables and data parameters. We note that the controllability is originally defined as a language-based property and, thus, meant for deterministic discrete-event systems. Extensions of controllability for parameterized languages are proposed in (Chen and Lin, 2000; Gaudin and Deussen, 2007). For nondeterministic discrete-event systems, there are several proposed notions, relying on commonly observed traces in (Fabian and Lennartson, 1996; Zhou et al., 2006), failure semantics in (Overkamp, 1997), or (bi)simulation semantics in (Baeten et al., 2011b). For nondeterministic extended finite automata with variables, introduced in (Skoldstam et al., 2007), the proposed notion of so-called state controllability of (Miremadi et al., 2008) relies on an extension of the work of (Fabian and Lennartson, 1996). Both works of (Overkamp, 1997) and (Baeten et al., 2011b) rely on preorder behavioral relations to formulate the notion of controllability, the former relying on failure-trace semantics, whereas the latter is (bi)simulation-based. Even though, it has been argued that refinements based on these two types of semantics have similar properties, cf. (Eshuis and Fokkinga, 2002), (bi)simulation-based refinements are finer notions that are supported by more efficient algorithms, like (Markovski, 2012a), which have already been employed in a supervisory control setting (Barrett and Lafortune, 1998).

To capture the notion of controllability, we rely

$$\frac{s \overset{a}{\longmapsto} s',\ \mathrm{v}_\delta(\gamma(s,a,s')) = \mathrm{T},\ \delta'(X) = \begin{cases} \mathrm{e}_\delta(\alpha((s,a,s'),X)), & \text{if } ((s,a,s'),X) \in \mathrm{D}(\alpha) \\ \delta(X), & \text{otherwise} \end{cases}}{(s,\delta) \xrightarrow{a} (s',\delta')}$$

Figure 3: Operational semantics of finite automata with variables

on a behavioral preorder termed *partial bisimulation*, first introduced in the co-algebraic characterization of (Rutten, 2000) and, subsequently, lifted to a process theory in (Baeten et al., 2011b). In essence, we employ this preorder to state a relation between the supervised plant and the original plant allowing controllable events to be simulated, while requiring that uncontrollable event are bisimulated. This ensures that the supervisor does not disable uncontrollable events, while preserving the branching structure of the plant. We will show that this notion subsumes the notion of state controllability for finite automata with variables. Moreover, we will show that state controllability is not a preorder and that some plants are considered as uncontrollable, even though there exist suitable supervisory controllers. Finally, by employing the proposed notion of controllability, we will show that it is possible to eliminate spurious plant nondeterminism, i.e., nondeterminism can be eliminated without sacrificing supervised plant behavior.

2 FINITE AUTOMATA WITH VARIABLES

In order to directly relate our notion of controllability with previous work, we model nondeterministic discrete-event systems by means of finite automata with variables. For a full treatment of supervisory control theory in a process-theoretic setting, we refer to (Baeten et al., 2011b; Baeten et al., 2011a; Markovski, 2012b) for event-, state-, and data-based supervision, respectively. In general, we allow arbitrary variable domains, even though variables with finite domains can be eliminated in order to employ more efficient synthesis procedures, as suggested in (Skoldstam et al., 2007). We suppose that the variables are given by the set V, where given a variable $X \in \mathsf{V}$, its domain is denoted by $\mathrm{D}(X)$. (Standard arithmetical) expressions over a set of variables $V \subset \mathsf{V}$ are denoted by $\mathsf{F}(V)$ and they are evaluated with respect to $\mathrm{e}_\delta \colon \mathsf{F}(V) \to \mathrm{D}(V)$, where $\delta \colon V \to \mathrm{D}(V)$ holds the variable assignments. We note that for the sake of clarity of presentation, we do not take into consideration the expressions that do not evaluate within the variable domain and extensions to inconsistent processes can be handled by a straightforward extension of the approach of (Baeten et al., 2011a). By $\mathsf{B}(V)$ we denote Boolean expression over the set of variables $V \subset \mathsf{V}$ where the atomic propositions are given by some set of predefined predicates, the logical constants false F and true T, and the set of standard logical operators. The obtained Boolean expressions are evaluated with respect to a given valuation $\mathrm{v}_\delta \colon \mathsf{B}(V) \to \{\mathrm{F},\mathrm{T}\}$, where again $\delta \colon V \to \mathrm{D}(V)$.

Definition 1. *A finite automaton with variables G is given by the tuple* $G = (S, A, V, \longmapsto, \gamma, \alpha, (s_0, \delta_0))$, *where*

- *S is a finite set of states;*
- *A is a finite set of event labels;*
- $V \subset \mathsf{V}$ *is a finite set of variables;*
- $\longmapsto \subset S \times A \times S$ *is a labeled transition relation;*
- $\gamma \colon \longmapsto \to \mathsf{B}(V)$ *are transition guards;*
- $\alpha \colon (\longmapsto \times V) \rightharpoonup \mathsf{F}(V)$ *is a partial updating function; and*
- (s_0, δ_0) *is the initial state* $s_0 \in S$ *and initial data assignment* $\delta_0 \colon V \to \mathrm{D}(V)$.

If the set of variables of a finite automaton with variables G, as given by Definition 1, is empty, then G is a standard automaton with labeled transitions. For the transition relations, we will employ infix notation and write $s \overset{a}{\longmapsto} s'$ for $(s,a,s') \in \longmapsto$.

The dynamics of the finite automaton with variables G is given by the transition relation $\longrightarrow \subseteq S \times (V \to \mathrm{D}(V)) \times A \times S \times (V \to \mathrm{D}(V))$, which is determined by the actual evaluation of the guards with respect to the value assignments. In order to keep track of the updated variable values, we employ the data assignment function $\delta \colon V \to \mathrm{D}(V)$. Now, the semantics of G is given by $\longrightarrow$, where initially the automaton is in state s_0 with environment δ_0, denoted by (s_0, δ_0). The dynamics of (s, δ) is captured by the operational rule depicted in Figure 3, following the notation of structural operational semantics of (Baeten et al., 2010), where the premise must hold, so that the bottom transition can be taken.

The rule states that a transition is possible if such labeled transition is defined in the automaton, the guard of that transition evaluates to true, whereas the variables are updated according to the partial updating function. It is not difficult to observe that the transition relation $\longrightarrow$ induces a labeled transition system with state space $S \times \mathrm{D}(V)$, set of labels A, and initial state (s_0, δ_0).

$$(s_1,s_2) \overset{a}{\longmapsto} \begin{cases} (s_1',s_2), & \text{if } s_1 \overset{a}{\longmapsto}_1 s_1', a \in A_1 \setminus A_2 \\ (s_1,s_2'), & \text{if } s_2 \overset{a}{\longmapsto}_2 s_2', a \in A_2 \setminus A_1 \\ (s_1',s_2'), & \text{if } s_1 \overset{a}{\longmapsto}_1 s_1', s_2 \overset{a}{\longmapsto}_2 s_2', a \in A_1 \cap A_2 \end{cases}$$

$$\gamma((s_1,s_2),a,(s_1',s_2')) = \begin{cases} \gamma_1(s_1,a,s_1'), & \text{if } s_1 \overset{a}{\longmapsto}_1 s_1', a \in A_1 \setminus A_2 \\ \gamma_2(s_2,a,s_2'), & \text{if } s_2 \overset{a}{\longmapsto}_2 s_2', a \in A_2 \setminus A_1 \\ \gamma_1(s_1,a,s_1') \wedge \gamma_2(s_2,a,s_2') \;, & \text{if } s_1 \overset{a}{\longmapsto}_1 s_1', s_2 \overset{a}{\longmapsto}_2 s_2', a \in A_1 \cap A_2 \end{cases}$$

$$\alpha(((s_1,s_2),a,(s_1',s_2')),X) =$$
$$\begin{cases} \alpha_1((s_1,a,s_1'),X), & \text{if } ((s_1,a,s_1'),X) \in \mathrm{D}(\alpha_1), ((s_2,a,s_2'),X) \notin \mathrm{D}(\alpha_2) \\ \alpha_2((s_2,a,s_2'),X), & \text{if } ((s_2,a,s_2'),X) \in \mathrm{D}(\alpha_2), ((s_1,a,s_1'),X) \notin \mathrm{D}(\alpha_1) \\ \alpha_1((s_1,a,s_1'),X), & \text{if } \begin{array}{l} ((s_1,a,s_1'),X) \in \mathrm{D}(\alpha_1), ((s_2,a,s_2'),X) \notin \mathrm{D}(\alpha_2), \\ \alpha_1((s_1,a,s_1'),X) = \alpha_2((s_2,a,s_2'),X) \end{array} \end{cases}$$

Figure 4: Definition of $\longmapsto$, γ, and α of Definition 3.

$$((s_1,\delta_1),(s_2,\delta_2)) \xrightarrow{a} \begin{cases} ((s_1',\delta_1'),(s_2,\delta_2)), & \text{if } (s_1,\delta_1) \xrightarrow{a}_1 (s_1',\delta_1'), a \in A_1 \setminus A_2 \\ ((s_1,\delta_1),(s_2',\delta_2')), & \text{if } (s_2,\delta_2) \xrightarrow{a}_2 (s_2',\delta_2'), a \in A_2 \setminus A_1 \\ ((s_1',\delta_1'),(s_2',\delta_2')), & \text{if } (s_1,\delta_1) \xrightarrow{a}_1 (s_1',\delta_1'), (s_2,\delta_2) \xrightarrow{a}_2 (s_2',\delta_2'), a \in A_1 \cap A_2 \end{cases} .$$

Figure 5: Definition of $\longrightarrow$ of Proposition 1.

Definition 2. *Given an automaton with variables $G = (S, A, V, \longmapsto, \gamma, \alpha, (s_0, \delta_0))$, we define the induced labeled transition system by $\mathrm{T}(G) = (S \times \mathrm{D}(V), A, \longrightarrow, (s_0, \delta_0))$, where:*

- *$S \times \mathrm{D}(V)$ is a set of states;*
- *A is the set of events taken over from G;*
- *$\longrightarrow \subseteq S \times (V \to \mathrm{D}(V)) \times A \times S \times (V \to \mathrm{D}(V))$ is the instantiated labeled transition relation as given by the operational rule of Figure 3; and*
- *(s_0, δ_0) is the initial state of the labeled transition system induced by the initial state of G and its initial variable valuation.*

If the set of variables is empty, i.e., $V = \emptyset$, then $\longmapsto$ and $\longrightarrow$ coincide, provided that the (then trivial) transition guards are set to be true, and G reduces to a standard automaton.

In order to define the language generated by automaton G, we extend the transition relation $\longrightarrow$ to a multistep transition relation $\longrightarrow^*$. By A^* we define the set of strings made from the labels in A that label the transitions of $\longrightarrow^*$, where ε denotes the empty string and st denotes the concatenation of the strings s and t for $s, t \in A^*$. Now, the multistep transition relation is given by the operation rules (1):

$$\frac{}{(s,\delta) \xrightarrow{\varepsilon}^* (s,\delta)}$$
$$\frac{(s,\delta) \xrightarrow{t}^* (s'',\delta''),\ (s'',\delta'') \xrightarrow{a} (s',\delta)',\ t \in A^*,\ a \in A}{(s,\delta) \xrightarrow{ta}^* (s',\delta')} . \tag{1}$$

By $(s,\delta) \xrightarrow{t}^*$ we denote that there exists (s',δ') such that $(s,\delta) \xrightarrow{t}^* (s',\delta')$. Now, the language generated by the automaton G is given by $L(G)$, where $L(G) = \{t \in A^* \mid (s_0,\delta_0) \xrightarrow{t}^* \}$.

In order to couple the plant and the supervisor, we define a synchronous composition of two automata that synchronizes on transitions with the same labels and interleaves on the other transitions. We note that, in general, the synchronous composition cannot be defined due to conflicts induced by the partial assignment functions α. A simple counterexample is the situation where two automata need to synchronize on transitions with the same label that update the same variable to two different values, as noted in (Skoldstam et al., 2007). Again, for the sake of clarity, we do not consider conflicting situations, which are easily detectable as none of the conditions for the partial updating functions in Definition 3 apply.

Definition 3. *Let $G_1 = (S_1, A_1, V_1, \longmapsto_1, \gamma_1, \alpha_1, (s_{01}, \delta_0))$ and $G_2 = (S_2, A_2, V_2, \longmapsto_2, \gamma_2, \alpha_2, (s_{02}, \delta_0))$. The synchronous composition of G_1 and G_2 is given by $G_1 \parallel G_2 = (S_1 \times S_2, A_1 \cup A_2, V_1 \cup V_2, \longmapsto, \gamma, \alpha, ((s_{01}, s_{02}), \delta_0))$, where $\longmapsto$, γ, and α are defined in Figure 4, where $\wedge$ denotes logical conjunction.*

Definition 3 is given directly in terms of automata with variables, unlike the work of (Skoldstam et al., 2007), where it is given in terms of the underlying labeled transition system. Now, given two finite automata with variables G_1 and G_2, we can derive the underlying transition systems $\mathrm{T}(G_1)$ and $\mathrm{T}(G_2)$. It is not difficult to show that $\mathrm{T}(G_1 \parallel G_2)$ coincides with $\mathrm{T}(G_1) \parallel \mathrm{T}(G_2)$, where the synchronization on the relation $\longrightarrow$ is defined as for $\longmapsto$.

Proposition 1. *Let* $G_i = (S_i, A_i, V_i, \longmapsto_i, \gamma_i, \alpha_i, (s_{0i}, \delta_0))$ *for* $i \in \{1,2\}$ *be such that* $G_1 \parallel G_2$ *is well-defined. Let* $(S_i \times \delta_i, A_i, \longrightarrow_i, (s_{0i}, \delta_0))$ *be the underlying labeled transition systems, where* $\longrightarrow_i$ *is induced by the operational rule of Figure 3 and* $\delta_i \colon V_i \to \mathrm{D}(V_i)$*, for* $i \in \{1,2\}$*. Let* $\mathrm{T}(G_1) \parallel \mathrm{T}(G_2) = ((S_1 \times \delta_1) \times (S_2 \times \delta_2), A_1 \cup A_2, \longrightarrow, ((s_{01}, \delta_0), (s_{02}, \delta_0)))$*, where* $\longrightarrow$ *is defined as in Figure 5. Then,* $\mathrm{T}(G_1 \parallel G_2)$ *is isomorphic to* $\mathrm{T}(G_1) \parallel \mathrm{T}(G_2)$.

The proof of Proposition 1 is meticulous, but straightforward, by showing that the constructions given in Definition 3 form an isomorphic transition system as the one defined in the proposition. It is worthwhile noting that the definition of $\longrightarrow$ in Proposition 1 does not impose an additional condition for the situation when $((s_1, \delta_1), (s_2, \delta_2)) \xrightarrow{a} ((s_1', \delta_1'), (s_2', \delta_2'))$ that δ_1' and δ_2' should coincide on the common updated variables. This is directly implied by the construction of α in Definition 3.

A direct corollary of Definition 3 and Proposition 1 is that the language of the synchronization is an intersection of the languages of the components of the composition, i.e., $L(G_1 \parallel G_2) = L(G_1) \cap L(G_2)$. This enables a connection with the original supervisory control theory of finite automata of (Ramadge and Wonham, 1987; Cassandras and Lafortune, 2004).

3 CONTROLLABILITY

Given an automaton with a set of labels A, we split the labels to set of controllable C and uncontrollable U labels such that $C \cap U = \emptyset$ and $C \cup U = A$. To model the plant we can take an unrestricted finite automaton with variables

$$P = (S_P, A_P, V_P, \longmapsto_P, \gamma_P, \alpha_P, (s_{0P}, \delta_0)), \qquad (2)$$

as the uncontrolled system is allowed to have every possible type of behavior. We note that the plant is typically obtained as a (well-defined) parallel composition of multiple concurrent components, which ultimately results in the process modeled by P.

The supervisor, however, is required to be a deterministic process, as it has to send unambiguous feedback to the plant and it is not allowed to alter the state of the plant, i.e., it must not comprise variable assignments, as suggested in (Markovski, 2012b). The supervisor can rely either on synchronization of events that keeps the history of the plant as in the original setting of (Ramadge and Wonham, 1987; Cassandras and Lafortune, 2004) or on data observation from the plant to make supervision decisions in the vein of (Miremadi et al., 2008; Markovski, 2012b). In both cases, we can assume that the supervisor is given as an deterministic automaton

$$S = (S_S, A_S, V_S, \longmapsto_S, \gamma_S, \emptyset, (s_{0S}, \delta_0)), \qquad (3)$$

where $C \subseteq A_S \subseteq A_P$, $V_S \subseteq V_P$, and the labeled transition function $\longmapsto_S$ is such that if $s \overset{a}{\longmapsto}_S s'$ and $s \overset{a}{\longmapsto}_S s''$, then $s' = s''$ for every $s, s', s'' \in S_S$ and $a \in A_S$.

We note that the supervisor can choose not to synchronize on some uncontrollable event from the plant, but its alphabet must comprise all controllable events as the supervisor must supply the control signals. Furthermore, the supervisor has no need of additional variables, as it does not update any variables, i.e., $\alpha_S = \emptyset$. Consequently, there is never a conflict in the synchronization between the plant and the supervisor, and the composition $P \parallel S$ is well-defined. If the supervisor does not rely on data-based observations, but employs synchronization of events to keep track of the state of the plant, then additionally $\gamma_S(s, a, s') = T$ for all $(s, a, s') \in \longmapsto$.

The composition $P \parallel S$ models the supervised plant, i.e., the behavior of the controlled system as given by the supervisory feedback loop of Figure 2. We note that the transition system

$$\mathrm{T}(P \parallel S) = (S_P \times S_S \times \delta_P, A_P, \longrightarrow, (s_{0P}, s_{0S}, \delta_0)), \quad (4)$$

where $\delta_P \colon V_P \to \mathrm{D}(V_P)$ and $\longrightarrow$ is defined by the operational rule of Figure 3.

To state that the supervisor has no control over the uncontrollable events, the language-based controllability of the original setting of (Ramadge and Wonham, 1987; Cassandras and Lafortune, 2004) is stated as:

$$L(P \parallel S)U \cap L(P) \subseteq L(P \parallel S), \qquad (5)$$

where $L(P \parallel S)U$ denotes the concatenation of the language of the supervised plant and the set of uncontrollable labels. Intuitively, the controllability relation (5) demands that all uncontrollable events available in reachable states of the original plant by traces enabled by the supervisor, must also be available in the supervised plant. This ensures that the supervisor does not disable any uncontrollable events when forming the supervised plant.

This definition has been subsequently extended to so-called state controllability in (Fabian and Lennartson, 1996; Zhou et al., 2006; Miremadi et al., 2008) for nondeterministic discrete-events systems (with variables). Given an automaton $G = (S, A, V, \longmapsto, \gamma, \alpha, (s_0, \delta_0))$ with a transition relation

$\longrightarrow$, let $\mathrm{E}(s,\delta)$ denote the set of enabled transitions of the state (s,δ) for $s \in S$ and $\delta\colon V \to \mathrm{D}(V)$, i.e., $\mathrm{E}(s,\delta) = \{a \in A \mid (s,\delta) \xrightarrow{a}\}$.

Definition 4. *Let P and S be finite automata with variables, representing the plant and the supervisor. A state $(s_P,(s_P,s_S),\delta_P)$ of the transition system $\mathrm{T}(P \parallel (P \parallel S))$ is defined as controllable, if it holds that*

$$A_S \cap U \cap \mathrm{E}(s_P,\delta_P) \subseteq \mathrm{E}((s_P,s_S),\delta_P).$$

A plant P is state controllable with respect to S if and only if all reachable states of $\mathrm{T}(P \parallel (P \parallel S))$ are state controllable.

Intuitively, the parallel composition between of the plant and the supervised plant helps identify all states in the original and the supervised plant that can be reached by the same trace. According to Definition 4, controllable states ensure that all uncontrollable events that are synchronized between the plant and the supervisor, given by $A_S \cap U$, that are also enabled in the reached plant state (s_P,δ_P) by following the same trace, must be enabled in the reached supervised plant state $((s_P,s_S),\delta_P)$. Note that both states must have the same variable assignment function δ_P as the supervisor has an empty updating function, so it does not influence the updating of the variables.

We note that the definition relies on the underlying transition system, employing it to identify the necessary control actions. It is not difficult to show that state controllability implies language controllability, as given in (5), for deterministic automata, see (Skoldstam et al., 2007). The key observation is that $P \parallel P$ coincides with P for deterministic systems, implying that $P \parallel S$ can act as a supervisor and lead to the same supervised behavior as S.

Here, we take a closer look at the state controllability condition for nondeterministic plants. Condition (4) essentially requires that all states that are reachable by the same trace, must also enable the same uncontrollable events. This proves to be too strict in some situations. Consider the automata depicted in Figure 6, where state names are given inside the circles, all guards are set to be true, there are no variables, the event labeled by c is controllable, whereas the events labeled by u_1 and u_2 are uncontrollable. Suppose that a plant is given by automaton P and a supervisor by automaton S. As the supervisor does not disable any events, we can assume that the control requirements do not restrict the behavior of the plant, i.e., the supervised plant depicted by automaton $P \parallel S$ coincides with the plant. In such reflexive situations, it is always possible to find a supervisor that simply allows all events of the plant, trivially "controlling" the plant.

Now, putting in parallel plant P and supervised plant $P \parallel S$, leads to automaton $P \parallel (P \parallel S)$ as depicted in Figure 6. This parallel composition reveals that states p_2 of P and (p_3,s_2) of $P \parallel S$ are reachable by the same trace. However, state $(p_2,\emptyset)$ of the transition system $\mathrm{T}(P)$ enables the uncontrollable transition labeled by u_1, whereas state $((p_3,s_2),\emptyset)$ of transition system $\mathrm{T}(P \parallel S)$ enables only the uncontrollable transition labeled by u_2. This directly implies that plant P is state uncontrollable with respect to $P \parallel S$, i.e., it is not state controllable with respect to itself. Thus, state controllability is not a preorder relation, as plants that have states that enable different sets of uncontrollable events in states that can be reached by the same trace are deemed uncontrollable, despite the existence of a trivial supervisor that enables all transitions.

4 PARTIAL BISIMULATION

We propose to employ the behavioral relation termed partial bisimulation to defined controllability for finite automata with variables. Partial bisimulation was first introduced in (Rutten, 2000) to capture language controllability in a coalgebraic setting. It was lifted in (Baeten et al., 2011b) to a process theory for supervisory control of nondeterministic discrete-event systems. Here, we provide an interpretation for finite automata with variables and discuss its relationship with state controllability.

Partial bisimulation is parameterized by a so-called bisimulation action set B. The relation requires that the labeled transitions of the first transition system are simulated by the second transition system, whereas only the labels of the second transition system that are in the bisimulation action set B are bisimulated back by the first one. The intuition behind this definition is that the bisimulation action set plays the role of the uncontrollable actions that must always be enabled both in the original and the supervised plant, whereas it is sufficient to only simulate controllable events, as these can be restricted by the supervisor.

Definition 5. *Let $T_i = (S_i, A_i, \longrightarrow_i, s_{0i})$ for $i \in \{1,2\}$ be two transition systems. A relation $R \subseteq S_1 \times S_2$ is said to be a partial bisimulation with respect to a bisimulation action set $B \subseteq A_2$, if for all $(s_1,s_2) \in R$, it holds that:*

1. *if $s_1 \xrightarrow{a} s_1'$ for $a \in A_1$ and $s_1' \in S_1$, then there exist $a \in A_2$ and $s_2' \in S_2$ such that $s_2 \xrightarrow{a} s_2'$ and $(s_1',s_2') \in R$;*
2. *if $s_2 \xrightarrow{b} s_2'$ for $b \in B$ and $s_2' \in S_2$, then there exist $b \in A_1$ and $s_1' \in S_1$ such that $s_1 \xrightarrow{a} s_1'$ and $(s_1',s_2') \in R$;*

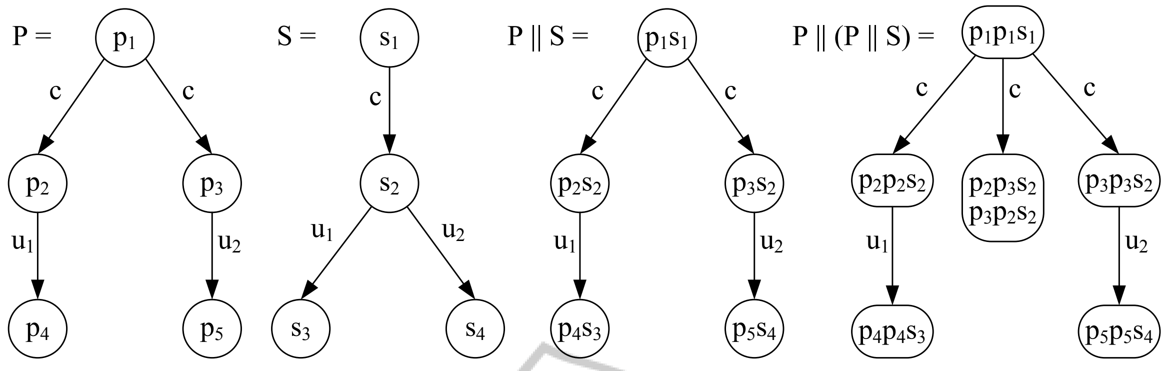

Figure 6: A nondeterministic plant P, a deterministic supervisor S, and the resulting state uncontrollable nondeterministic supervised plant $P \parallel S$.

If R is a partial bisimulation relation such that $(s_{01}, s_{02}) \in R$, then T_1 is partially bisimilar to T_2 with respect to B and we write $T_1 \leq_B T_2$. If $T_2 \leq_B T_1$ holds as well, we write $T_1 =_B T_2$.

We note that due to condition 1. of Definition 5, it must hold that $A_1 \subseteq A_2$, whereas due to condition 2. it holds that $B \subseteq A_1$ as well. It is not difficult to show that partial bisimilarity is a preorder relation (Baeten et al., 2011b). In addition, following the guidelines of (Rutten, 2000), it can be shown that $\leq_B$ is a partial bisimulation relation with respect to B. Thus, we obtain standard results for the partial bisimulation preorder and equivalence, similarly as for the simulation preorder and equivalence of (Glabbeek, 2001). Moreover, the partial bisimulation preorder is shown a precongruence for the most prominent processes operations following the guidelines of (Baeten et al., 2011b). Finally, we note that $T_1 =_{A_1 \cup A_2} T_2$ amounts to bisimulation, whereas $T_1 \leq_\emptyset T_2$ reduces to simulation preorder and $T_1 =_\emptyset T_2$ reduces to simulation equivalence, as noted in (Baeten et al., 2011b).

Now, suppose that as before, the plant is given by finite automaton with variables P, whereas the supervisor is given by S, and the supervised plant is given by $P \parallel S$. Then, the supervisor may restrict some controllable events from the plant, whereas all available uncontrollable events in the reachable states should be enabled. This can be expressed by requesting that the transition system of the supervised plant is partially bisimulated by the transition system of the original plant with respect to the uncontrollable events, i.e.,

$$\mathrm{T}(P \parallel S) \leq_U \mathrm{T}(P). \qquad (6)$$

It is immediate that $\mathrm{T}(P) \leq_U \mathrm{T}(P)$, when $P \parallel S$ coincides with P as in the example of Figure 6. It is also not difficult to show that for deterministic processes, relation (6) reduces to language controllability of (5), see (Rutten, 2000; Baeten et al., 2011b). Next, we show that controllability as defined in (6) by means of partial bisimulation is a coarser notion than state controllability of Definition 4.

Theorem 1. *Let P and S be finite automata with variables representing the plant and the supervisor. If P is state controllable with respect to S according to Definition 4, then relation (6) holds.*

Proof. Let us assume that $P = (S_P, A_P, V_P, \longmapsto_P, \gamma_P, \alpha_P, (s_{0P}, \delta_0))$ and $S = (S_S, A_S, V_S, \longmapsto_S, \gamma_S, \emptyset, (s_{0S}, \delta_0))$. We define the relation

$$R = \{(((p,s),\delta_P),(p,\delta_P)) \mid \\ \exists t \in A_P^*\colon (p_0,(p_0,s_0),\delta_0) \xrightarrow{t}{}^{*} (p,(p,s),\delta_P)\}.$$

We show that R is a partial bisimulation relation between $\mathrm{T}(P \parallel S)$ and $\mathrm{T}(P)$ with respect to the uncontrollable labels $U \subseteq A_P$. Suppose that $(((p,s),\delta_P),(p,\delta_P)) \in R$ for some states $((p,s),\delta_P) \in S_P \times S_S \times (V_P \to \mathrm{D}(V_P))$ and $(p,\delta_P) \in S_P \times (V_P \to \mathrm{D}(V_P))$.

Let $((p,s),\delta_P) \xrightarrow{a} ((p',s'),\delta'_P)$ for some $a \in A_P$. Then, according to Definition 3 and the operational rule of Figure 3, either $a \in A_P \setminus A_S$ or $a \in A_S$. In the former case, we have that $s = s'$, so $(p,\delta_P) \xrightarrow{a} (p',\delta'_P)$ and $(((p',s),\delta'_P),(p',\delta'_P)) \in R$. In the latter case, we have that $((p,s),\delta_P) \xrightarrow{a} ((p',s'),\delta'_P)$ for some $s' \in S_S$. However, since the updating function of the supervisor S is empty and the action $a \in A_S$ is synchronizing, we have that again $(p,\delta_P) \xrightarrow{a} (p',\delta'_P)$ with $(((p',s'),\delta'_P),(p',\delta'_P)) \in R$.

Now, suppose that $(p,\delta_P) \xrightarrow{u} (p',\delta'_P)$ for some $u \in U$. Again, either $u \in U \setminus A_S$ or $u \in U$. If $u \notin A_S$, then u is not a synchronizing label, implying that $((p,s),\delta_P) \xrightarrow{u} ((p',s),\delta'_P)$ with $(((p',s),\delta'_P),(p',\delta'_P)) \in R$. If u is a synchronizing label, then by the condition for controllable states of Definition 4, we have $u \in \mathrm{E}((s_P,s_S),\delta_P)$, i.e., $((p,s),\delta_P) \xrightarrow{u} ((p',s'),\delta'_P)$ for some $((p',s'),\delta'_P) \in$

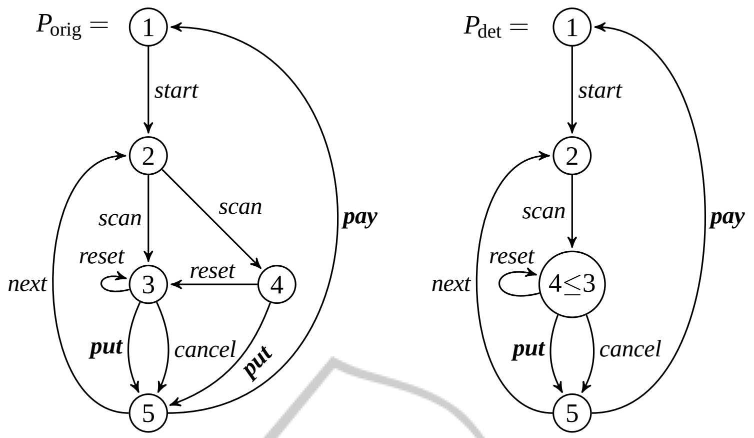

Figure 7: Checkout scanner of (Zhou et al., 2006) - A plant with spurious nondeterminism.

$S_P \times S_S \times (V_P \to \mathrm{D}(V_P))$ and $(((p', s'), \delta'_P), (p', \delta'_P)) \in R$, which completes the proof. $\square$

We have shown that every state controllable plant is also controllable with respect to condition (6). That the inclusion is strict follows immediately from the counterexample of Figure 6.

Condition (6) additionally implies that the same supervised behavior given by $P \parallel S$ is preserved for every plant P' such that $P' =_U P$, i.e., we have that $P' \parallel S =_U P \parallel S$, which is the basis of the algorithms developed in (Markovski, 2012a). This enables us to detect spurious nondeterministic behavior for which state controllability cannot be applied in general. We given an example from the literature of such nondeterministic behavior.

In Figure 7, plant P_{orig} represents a model of a faulty automated scanner that makes a shopping list of items to be purchased by the user. The scanner is faulty as sometimes it does not give an option to cancel a scanned item, e.g., when the user wants to return the product or just wants to check the price, and in that case the scanner needs to be reset. As suggested in (Zhou et al., 2006) the set of uncontrollable events is given by $U = \{pay\}$ as payment cannot be avoided, even though we also suggest to treat the event *put* as uncontrollable.

The interpretation is that if there is no cancelation of some scanned product, after a possible timeout, it should automatically be placed on the shopping list. It is easily observed that state 4 is partially bisimulated by state 3 and, thus, state 3 can be safely removed without any loss in behavior (the only situation where state 3 could not be removed arises if the event *cancel* is uncontrollable, which here is not the case). The resulting deterministic plant P_{det} reveals that P_{orig} actually contains no real nondeterministic behavior with respect to controllability. In the original setting of (Zhou et al., 2006) that employs state controllability for nondeterministic discrete-event systems, this observation was not possible and the plant P_{orig} is treated as nondeterministic.

5 CONCLUDING REMARKS

We defined a notion of controllability for finite automata with variables based on the behavioral preorder termed partial bisimulation. We showed that the proposed notion of controllability subsumes the prominent previous notion of state controllability, which was specifically tailored for nondeterministic finite automata with variables. Moreover, we showed that state controllability is not a preorder and that there exist state-uncontrollable plants for which it is possible to synthesize viable supervisory controllers. This situation was remedied by the new definition, which does not exclude the investigated cases. Moreover, we showed that the proposed setting enables detection of spurious nondeterministic behavior, i.e., it is possible to eliminate nondeterministic behavior that does not contribute to the behavior of the supervised system.

ACKNOWLEDGEMENTS

The work presented in this paper is supported by Dutch NWO project ProThOS, no. 600.065.120.11N124.

REFERENCES

Baeten, J., van Beek, D., van Hulst, A., and Markovski, J. (2011a). A process algebra for supervisory coordination. In *Proceedings of PACO 2011*, volume 60 of *EPTCS*, pages 36–55. Open Publishing Association.

Baeten, J. C. M., Basten, T., and Reniers, M. A. (2010). *Process Algebra: Equational Theories of Communicating Processes*, volume 50 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Baeten, J. C. M., van Beek, D. A., Luttik, B., Markovski, J., and Rooda, J. E. (2011b). A process-theoretic approach to supervisory control theory. In *Proceedings of ACC 2011*, pages 4496–4501. IEEE.

Barrett, G. and Lafortune, S. (1998). Bisimulation, the supervisory control problem and strong model matching for finite state machines. *Discrete Event Dynamic Systems*, 8(4):377–429.

Cassandras, C. and Lafortune, S. (2004). *Introduction to discrete event systems*. Kluwer Academic Publishers.

Chen, Y.-L. and Lin, F. (2000). Modeling of discrete event systems using finite state machines with parameters. In *Proceedings of CCA 2000*, pages 941 –946.

Eshuis, R. and Fokkinga, M. M. (2002). Comparing refinements for failure and bisimulation semantics. *Fundamenta Informaticae*, 52(4):297–321.

Fabian, M. and Lennartson, B. (1996). On nondeterministic supervisory control. *Proceedings of the 35th IEEE Decision and Control*, 2:2213–2218.

Gaudin, B. and Deussen, P. (2007). Supervisory control on concurrent discrete event systems with variables. In *Proceedings of ACC 2007*, pages 4274 –4279.

Glabbeek, R. J. v. (2001). The linear time–branching time spectrum I. *Handbook of Process Algebra*, pages 3–99.

Leveson, N. (1990). The challenge of building process-control software. *IEEE Software*, 7(6):55–62.

Ma, C. and Wonham, W. M. (2005). *Nonblocking Supervisory Control of State Tree Structures*, volume 317 of *Lecture Notes in Control and Information Sciences*. Springer.

Markovski, J. (2012a). Coarsest controllability-preserving plant minimization. In *Proceedings of WODES 2012*, pages 251–258. IFAC.

Markovski, J. (2012b). Communicating processes with data for supervisory coordination. In *Proceedings of FOCLASA 2012*, volume 91 of *EPTCS*, pages 97–111. Open Publishing Association.

Markovski, J., van Beek, D. A., Theunissen, R. J. M., Jacobs, K. G. M., and Rooda, J. E. (2010). A state-based framework for supervisory control synthesis and verification. In *Proceedings of CDC 2010*, pages 3481–3486. IEEE.

Miremadi, S., Akesson, K., and Lennartson, B. (2008). Extraction and representation of a supervisor using guards in extended finite automata. In *Proceedings of WODES 2008*, pages 193–199. IEEE.

Overkamp, A. (1997). Supervisory control using failure semantics and partial specifications. *IEEE Transactions on Automatic Control*, 42(4):498–510.

Ramadge, P. J. and Wonham, W. M. (1987). Supervisory control of a class of discrete-event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230.

Rutten, J. J. M. M. (2000). Coalgebra, concurrency, and control. In *Proceedings of WODES 2000*, pages 31–38. Kluwer.

Schiffelers, R. R. H., Theunissen, R. J. M., Beek, D. A. v., and Rooda, J. E. (2009). Model-based engineering of supervisory controllers using CIF. *Electronic Communications of the EASST*, 21:1–10.

Skoldstam, M., Akesson, K., and Fabian, M. (2007). Modeling of discrete event systems using finite automata with variables. In *Proceedings of CDC 2007*, pages 3387–3392. IEEE.

Zhou, C., Kumar, R., and Jiang, S. (2006). Control of nondeterministic discrete-event systems for bisimulation equivalence. *IEEE Transactions on Automatic Control*, 51(5):754–765.