Partially-Supervised Plants: Embedding Control Requirements in Plant Components

J. Markovski, D.A. van Beek, and J.C.M. Baeten*

Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{j.markovski,d.a.van.beek,j.c.m.baeten}@tue.nl

Abstract. Supervisory control deals with automated synthesis of controllers based on models of the uncontrolled system and the control requirements. In this paper we share the lessons learned from synthesizing controllers for a patient support system of an MRI scanner regarding the specification of the control requirements. We learned that strictly following the philosophy of supervisory control, which partitions specifications in an uncontrolled plant and control requirements, may lead to unnecessarily complex specifications and duplication of information. In such cases, the specification can be substantially simplified by embedding part of the control requirements in so-called partially-supervised plants. To formalize the new concepts, we apply a recently developed process-theoretic approach to supervisory control. The new method for analysis of the models provides a better insight into their underlying behavior, which is demonstrated by revisiting the models of the industrial study.

1 Introduction

Modern market trends dictate lower development costs and shorter time-to-market, while increasing demands for better quality, performance, safety, and ease of use. Among else, this raises the demands on the development of control software. Traditionally, software engineers write control software based on informal specification documents, amounting to a time-consuming iterative process as the control requirements constantly change during product development. This issue gave rise to supervisory control theory of discrete-event systems [9, 3], where high-level supervisory controllers are synthesized automatically based upon formal models of the hardware and control requirements.

The supervisory controller observes the discrete-event behavior of the system by receiving sensor signals from ongoing activities. Based upon these signals it makes a decision which activities are allowed to be carried out and sends back control signals to the hardware actuators. Under the assumption that the supervisory controller can react sufficiently fast on input, one can model this feedback loop as a pair of synchronizing processes. The model of the uncontrolled system, referred to as *plant*, is restricted by the model of the controller, referred to as

* This work is part of the EU project C4C, contract number FP7-ICT-223844.

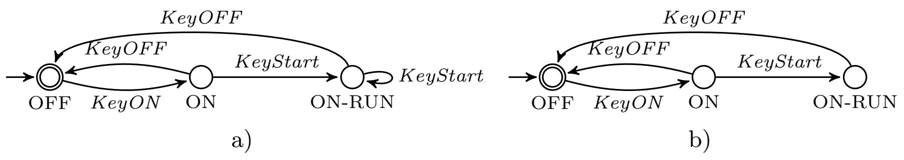

Fig. 1. Starting a car engine: a) plant component and b) control requirements.

supervisor. Traditionally, the plant is modeled as a set of observable traces of events, given as a set of synchronizing components (automata), whose joint recognized language corresponds to the observed traces. The events are split into *controllable events*, which can be disabled by the supervisor in the synchronous composition (typically actuator events), and *uncontrollable events*, which must always be allowed by the supervisor (typically sensor events). The *control requirements* specify allowed behavior again as sequences of events, leading to event-based supervisory control theory [9, 3].

In this paper, we revisit an industrial study regarding supervisory control of a patient support system for MRI scanners of Philips Healthcare [11]. We discuss the lessons learned from specifications of the plant and the control requirements. Namely, following the supervisory control paradigm, the plant should be modeled as unrestricted with respect to the controllable events, i.e., disabling of such events should be stated in the control requirements. However, following this paradigm may lead to duplicated specifications, slightly altered only to specify some controllable events that should be restricted. This situation usually occurs when one wants to make the specification of the (unsupervised) plant complete in the sense that all possible behavior is included, despite knowing it is irrelevant.

To provide a better intuition, we consider the process of starting a car engine. After turning the key in position ON, the engine can be started by turning the key to position start and releasing it (event *KeyStart*). Once it is started, by turning the key to position OFF, it is switched off. However, in position ON, there is no prohibition to turn the key again to "start" the already running engine. This is encountered (by accident) by almost everyone that drove in a car, observing strange noises produced by the engine. From a supervisory control point of view, the position of the key defines the behavior of the (unsupervised) plant component that models the starting of the car, depicted in Fig. 1a). The control requirements show the correct way of starting a car, depicted in Fig. 1b). Note that the requirements actually duplicate the plant component, omitting only the self loop in ON-RUN that specifies the turning of the key, while the engine is running. Moveover, every driver knows the correct way of starting a car, so we could argue that the *partially-supervised* behavior of the plant actually comprises the component in Fig. 1b). This makes for a more readable plant specification, while reducing a (superfluous) control requirement.

In the remainder, we formalize the notion of partially-supervised plants and we develop a method for analysis of the plant and control requirements that provides better insights into the underlying behavior. To this end, we employ a recent process-theoretic approach to supervisory control that captures the standard notion of controllability by means of a behavioral preorder termed partial bisimilarity [2, 10]. We retain the trace-based semantics by restricting to deterministic automata and we adapt partial bisimilarity to automata as used in supervisory control [3].

2 Supervisory Control Theory

We introduce some preliminary notions of automata and language theory as used in supervisory control theory [3]. Let $\mathcal{A} = \mathcal{C} \cup \mathcal{U}$ be the set of all events that can be observed in the plant, with $\mathcal{C}$ being the set of controllable events and $\mathcal{U}$ the set of uncontrollable events, such that $\mathcal{C} \cap \mathcal{U} = \emptyset$. We form traces and languages in a standard manner, i.e., $t \in \mathcal{A}^*$ is a trace and $L \subseteq \mathcal{A}^*$ is a language, where $\mathcal{A}^* \triangleq \{a_1 a_2 \ldots a_n \mid a_i \in \mathcal{A} \text{ for } 0 \leq i \leq n, n \in \mathbb{N}\}$ and ε denotes the unique empty trace $a_1 \ldots a_n$ for $n = 0$. By $t{\cdot}t'$ we denote the concatenation of the traces $t, t' \in \mathcal{A}^*$ and by $L{\cdot}L' \triangleq \{t{\cdot}t' \mid t \in L, t' \in L'\}$ the concatenation of languages. We omit $\cdot$ when clear from the context. We say that a language is prefix-closed if $L = \overline{L}$, where $\overline{L} \triangleq \{t \mid \text{there exists } t' \text{ such that } tt' \in L\}$.

We define a discrete-event automaton as a tuple $P = (\mathcal{S}_P, \mathcal{A}_P, \rightarrow_P, s_P, \mathcal{S}_P^m)$, where $\mathcal{S}_P$ is a set of states, $\mathcal{A}_P$ is the alphabet or the set of events used for synchronization, $\rightarrow_P \in \mathcal{S}_P \times \mathcal{A}_P \times \mathcal{S}_P$ the transition relation, s_P the initial state, and $\mathcal{S}_P^m$ is the set of marked states that denote successfully executed jobs. By $\mathcal{F}$ we denote the set of all finite automata. We define $\rightarrow_P^* \in \mathcal{S}_P \times \mathcal{A}_P^* \times \mathcal{S}_P$ as $s \xrightarrow{\varepsilon}{}^* s$ for all $s \in \mathcal{S}_P$, and $s \xrightarrow{at}{}_P^* s'$ for $a \in \mathcal{A}_P$ and $t \in \mathcal{A}_P^*$, if there exists $s'' \in \mathcal{S}_P$ such that $s \xrightarrow{a}{}_P s'' \xrightarrow{t}{}_P^* s'$. By $s \xrightarrow{t}{}_P^*$ we denote that there exists $s' \in \mathcal{S}_P$ such that $s \xrightarrow{t}{}_P^* s'$. Now, the recognized (prefix-closed) language of automaton P is given by $L(P) \triangleq \{t \in \mathcal{A}_P^* \mid s_P \xrightarrow{t}{}_P^* \}$. The recognized marked language additionally requests that the ending state is a marked state given by $L_m(P) \triangleq \{t \in \mathcal{A}_P^* \mid s_P \xrightarrow{t}{}_P^* s, s \in \mathcal{S}_P^m\}$.

By $P_1 \mid P_2 \triangleq (\mathcal{S}_1 \times \mathcal{S}_2, \mathcal{A}_1, \rightarrow, (s_1, s_2), \mathcal{S}_1^m \times \mathcal{S}_2^m)$ we denote the synchronous parallel composition of $P_1 = (\mathcal{S}_1, \mathcal{A}_1, \rightarrow_1, s_1, \mathcal{S}_1^m)$ and $P_2 = (\mathcal{S}_2, \mathcal{A}_2, \rightarrow_2, s_2, \mathcal{S}_2^m)$:

$$(s', s'') \xrightarrow{a} \begin{cases} (\overline{s}', \overline{s}'') & \text{if } a \in \mathcal{A}_1 \cap \mathcal{A}_2, s' \xrightarrow{a}{}_1 \overline{s}', \text{ and } s'' \xrightarrow{a}{}_2 \overline{s}'' \\ (\overline{s}', s'') & \text{if } a \in \mathcal{A}_1 \setminus \mathcal{A}_2 \text{ and } s' \xrightarrow{a}{}_1 \overline{s}' \\ (s', \overline{s}'') & \text{if } a \in \mathcal{A}_2 \setminus \mathcal{A}_1 \text{ and } s'' \xrightarrow{a}{}_2 \overline{s}''. \end{cases}$$

It is easily observed that this composition is commutative and associative [3]. Note that by increasing the alphabet of an automaton with events that occur in synchronizing automata, the parallel composition would remain the same, provided that these events were added as self loops in every state.

Suppose that the plant is given by $P = (\mathcal{S}_P, \mathcal{A}, \rightarrow_P, s_P, \mathcal{S}_P^m)$ and the control requirements by $R = (\mathcal{S}_R, \mathcal{A}, \rightarrow_R, s_R, \mathcal{S}_R^m)$. If there exists $S = (\mathcal{S}_S, \mathcal{A}, \rightarrow_S, s_S, \mathcal{S}_S^m)$ such that $L(P \mid S) = L(R)$, then we say that S is a supervisor for P that achieves R. We refer to $P \mid S$ as the *supervised plant*. We ensure that S does not

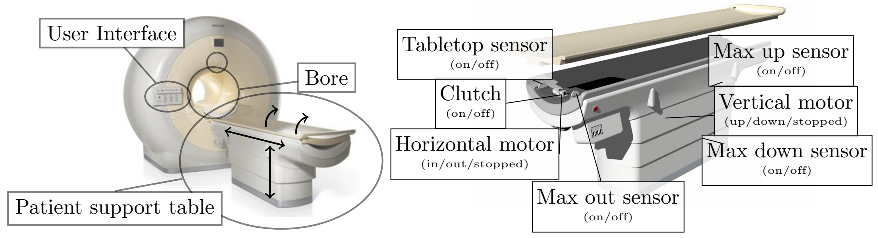

Fig. 2. Patient support system of an MRI scanner.

disable uncontrollable events by requesting that R is *controllable* with respect to P, expressed by $L(R)\mathcal{U} \cap L(P) \subseteq L(R)$ [9, 3]. Controllability is interpreted as follows. If we observe a desired trace in the plant followed by an uncontrollable event, then the control requirements cannot request that this uncontrollable event should be disabled after allowing that trace.

To assure, in addition, that the control is *nonblocking*, it is also required that $L(R) \subseteq L_m(P)$. The condition ascertains that every state can reach a marked state, guaranteeing that all jobs can be successfully finished, while preventing deadlocks and livelocks. If R is controllable with respect to P and also $L(R) \subseteq L_m(P)$, then one can guarantee the existence of a supervisor S, achieving the desired nonblocking supervised behavior R by restricting the plant P.

In general, the control requirements are not achievable and one seeks a *maximally permissive (nonblocking) supervisor*, its prefix-closed language uniquely defined for deterministic plants and control requirements as

$$M = \bigcup\{K \subseteq L(R) \cap L_m(P) \mid K \text{ is controllable with respect to } P\}.$$

In other words, the maximally permissive supervisor enables the greatest achievable nonblocking behavior that is controllable with respect to P and bounded by R. Consequently, if S is a supervisor for the plant P with respect to the control requirements R, then $L(S) \subseteq L(M) \subseteq L(R)$ and $L_m(S) \subseteq L_m(M) \subseteq L_m(R)$.

Next, we revisit the supervisor synthesis for a patient support system [11].

3 Supervisor Synthesis for a Patient Support System

The patient support system positions a patient inside an MRI scanner, see Fig. 2. The system comprises a vertical axis, a horizontal axis, and a user interface. Due to page limitations, we present only a part of the system [11]. The vertical axis consists of a lift with a motor drive and end sensors. The horizontal axis contains a removable tabletop, which can be moved in and out of the bore either by a motor drive, when the clutch is on, or by hand, otherwise. It contains sensors to detect the presence of the tabletop and its end positions. A tumble switch controls table movement and the clutch is controlled by a manual button.

The control should accomplish multiple control objectives. When the operator operates the tumble switch, the table should move up and down, or in and

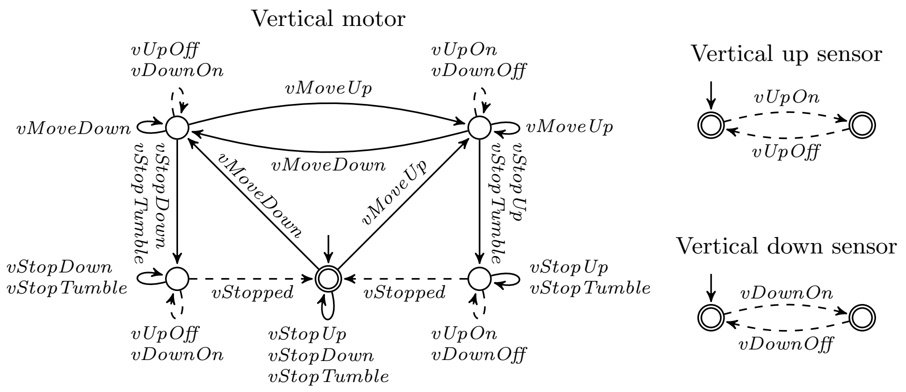

Fig. 3. Plant components for the vertical motor and end sensors.

out of the bore. This depends on the current position of the table and the position of the tumble switch. When the manual button is pushed, the clutch should be released such that the table can be moved manually by hand. Finally, the table should not move beyond its end positions, and it should not collide with the magnet. Note that we do not consider faulty behavior in this paper.

This system is more difficult to control then it might appear at first sight. It contains several complex interactions of components, and the overall finite state model of the uncontrolled system contained $6.3 \cdot 10^9$ states ($64 \cdot 10^6$ states without user interface). Recall, that here we show just a part of this system. Nonetheless, the manufacturer estimated one week for manual adaptation of the control software to meet a change in the control requirement, while adapting them using supervisor synthesis took merely four hours [11].

We model the plant and the control requirements using automata as given in Section 2. To visualize automata, we use circles for states, full and dashed labeled arrows for controllable and uncontrollable events, respectively, incoming arrows for initial states, and doubly-lined circles for marked states. The plant and control requirements are composed out of synchronizing models for each of the components. The alphabets of the automata are comprised of the transition labels. We assume full observation of the sensors and the actuators.

Vertical Axis The table moves up and down along the vertical axis, which comprises one actuator and two end sensors, see Fig. 3. The system should never be required to move beyond the maximally up and down position. We name the events such that their purpose becomes clear from the context. Initially, the motor is stopped and after any movement it should be able to return to its marked state. Movement is started via events *vMoveUp* and *vMoveDown*. If the motor is moving and a stop event, *vStopUp*, *vStopDown*, or *vStopTumble* is triggered, the motor slows down. When it comes to a halt, the event *vStopped* is emitted. The maximally up and down sensors are active if the table is at the end sensor position, otherwise the sensors are inactive. They are modeled by means of automata

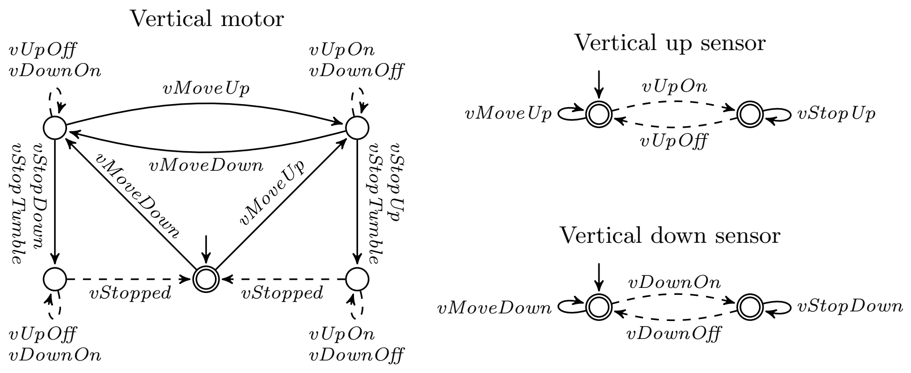

Fig. 4. Control requirements for the vertical motor and end sensors.

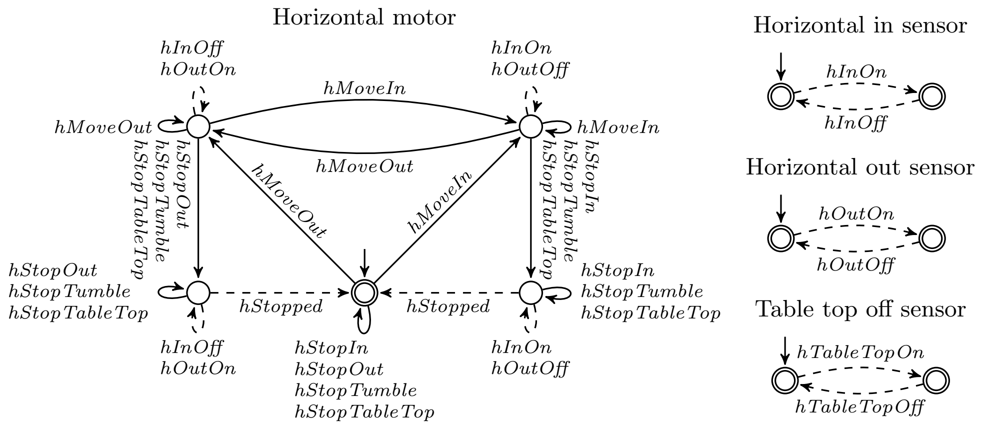

Fig. 5. Plant components for the horizontal motor, end sensors, and table top sensor.

with corresponding (uncontrollable) sensor events *vDownOn* and *vDownOff*, for the down sensor, and *vUpOn* and *vUpOff*, for the up sensor.

The sensors only change state when the table moves vertically. Only when the motor drive is moving the table up, the maximally down sensor can turn off, and the maximally up sensor can turn on, and vice versa. Although the end positions must be reachable, movement beyond them is not allowed. This implies that up movement is only allowed when the table is not maximally up and likewise for the down movement. Furthermore, up movement must be stopped when the table is maximally up and likewise for the down movement. In addition, once a stop event has been issued, there is no need to issue it again. These requirements lead to the models depicted in Fig. 4.

Note that when we add new events to automata, we are increasing its alphabet, and unless we add self loops in every state, we are actually restricting events in the parallel composition, cf. Section 2.

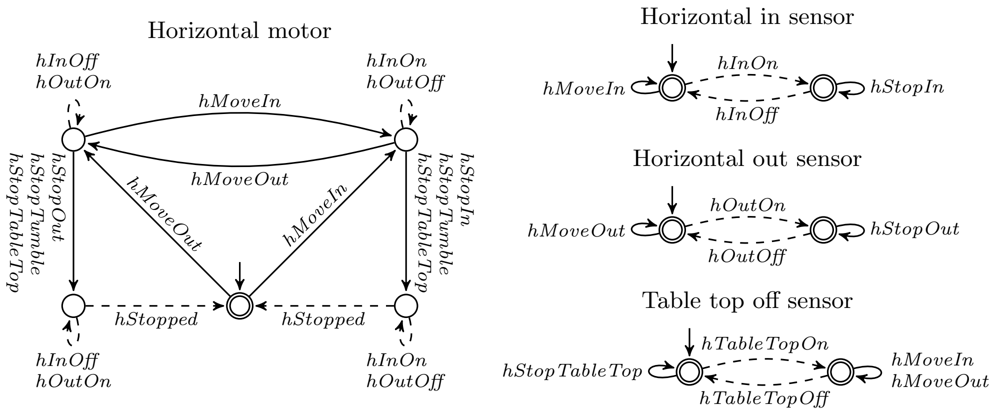

Fig. 6. Control requirements for horizontal motor, end sensors, and table top sensor.

Horizontal Axis The movement along the vertical axis is analogous to the one for the vertical axis, with the exception that the table top may be taken off by the operator, which is detected using an additional sensor. The plant components dealing with horizontal movement are depicted in Fig. 5, whereas the control requirements are depicted in Fig. 6.

User interface The user interface consists of a tumble switch that controls the table movement and a manual button that controls the operation mode via the clutch. In motorized mode, the tumble switch controls the movement of the table. In manual mode, the operator can move the table top by hand. When the manual button is pushed, the clutch is either applied or released, if allowed by the safety requirements of Fig. 10, leading to motorized or manual mode, respectively. The manual button push is associated to a safety timeout as manual operation is allowed only when the table top is on and it is in the topmost position, and the motors are stopped. As this takes some time, the button push might be forgotten by the operator. Fig. 7 depicts the plant components modeling the user interface.

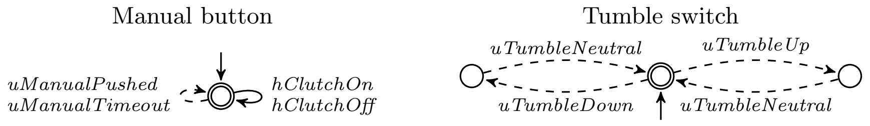

Fig. 7. Plant components for the manual button and the tumble switch.

When the manual button is pressed, either the clutch is applied or released, or a timeout occurs, that invalidates the button push. When the tumble switch is down, then either downward or outward movement is allowed, whereas when

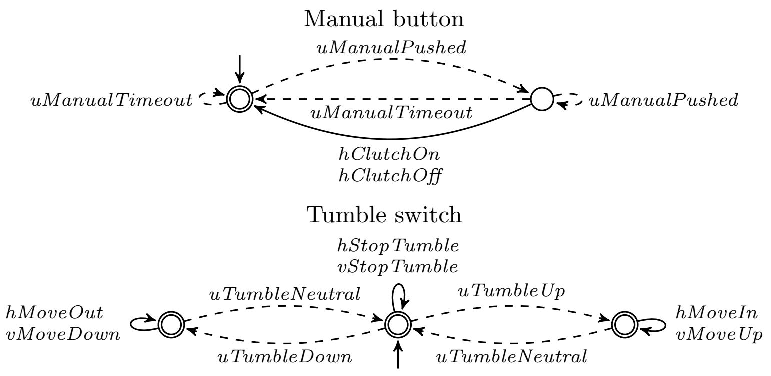

Fig. 8. Control requirements for the manual button and the tumble switch.

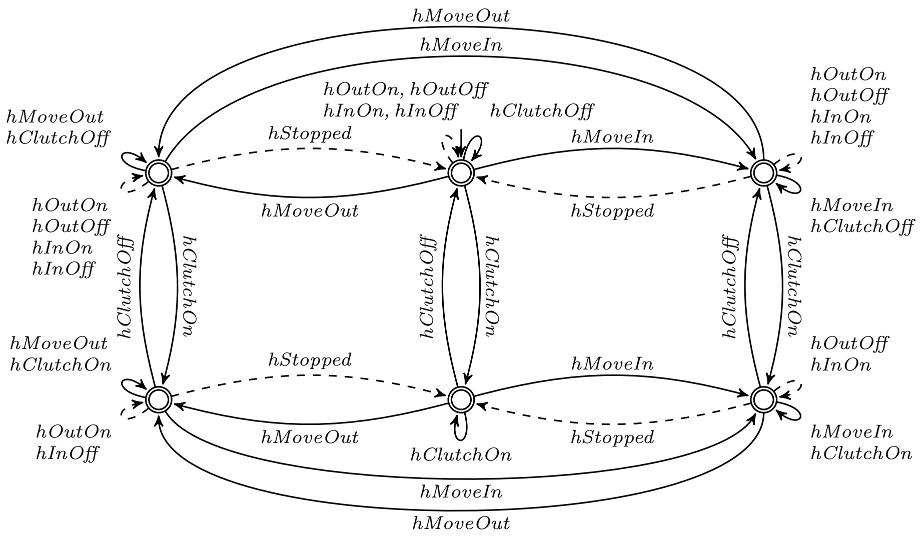

Fig. 9. Plant component relating horizontal actuator and sensor events.

the switch is up, either upward or inward movement is allowed. This is captured by the control requirements depicted in Fig. 8.

Finally, in Fig. 9, we capture the relationship between the horizontal motor and sensors. We note that when the clutch is off, every activation/deactivation of horizontal end sensors is possible, as the operator can move the table top unrestrictedly. We also note that the component depicted in Fig. 9 results from an interleaving (nonsynchronizing) parallel composition of a component that gives the relation between the clutch and horizontal sensor events and a component that describes motorized horizontal table movement [11].

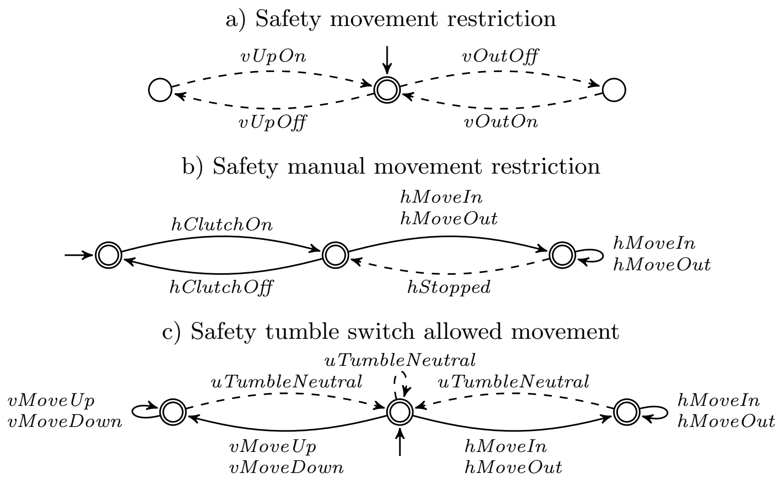

Fig. 10. Control requirements for safe patient support table movement.

To guarantee safe operation of the patient support system, the following safety movement restrictions apply. To ensure that the patient support table does not collide with the magnet of the MRI machine, it is required that the table can enter the magnet only if it is maximally up, and the table can move vertically only if it is fully retracted (Fig. 10a). The table top can be manually operated only if the horizontal movement of the motor is stopped (Fig. 10b). In addition, to change the axis of movement, the tumble switch must be first placed in neutral position, while the motors can be activated only if the tumble switch is not in neutral position (Fig. 10c).

One cannot help to notice the duplication of information in Fig. 3 and Fig. 4, Fig. 5 and Fig. 6, and Fig. 7 and Fig. 8. Moreover, for modelers involved in this study, the plant components depicted in Fig. 3, Fig. 5, and Fig. 7, are overly simplified and actually produce the countereffect of making the plant specification unclear. This is most obvious in Fig. 9, where one can hardly deduce anything about the relationship between the horizontal actuators and sensors.

In the next chapter, we alleviate some of these issues by embedding a part of the control requirements into the plant components.

4 Process-Theoretic Approach to Controllability

We present a process-theoretic approach to supervisory control theory that enables us to manipulate more easily with the underlying notions. We define controllability from a process-theoretic perspective in terms of a so-called partial bisimilarity preorder. This preorder is meant to capture that uncontrollable events should not be disabled by the supervisor and it gives the relation between the original and the supervised plant. It requires that the unrestricted plant simulates, i.e., it is enabled to perform, every event of the supervised plant, but it is required that the supervised plant only simulates back uncontrollable events. We note that in the original process-theoretic setting of [2] we employed labeled transition systems, whereas here we adjust the behavioral (semantic) relation to accommodate deterministic automata in the vein of [3].

Definition 1. *Let $P_1 = (\mathcal{S}_1, \mathcal{A}_1, \rightarrow_1, s_1, \mathcal{S}_1^m)$ and $P_2 = (\mathcal{S}_2, \mathcal{A}_2, \rightarrow_2, s_2, \mathcal{S}_2^m)$ be two finite automata with $\mathcal{A}_1 = \mathcal{A}_2 = \mathcal{A}' \subseteq \mathcal{A}$. A relation $Q \subseteq \mathcal{S}_1 \times \mathcal{S}_2$ is a partial bisimulation between P_1 and P_2 with respect to the bisimulation action set $B \subseteq \mathcal{A}'$ if for all $p_1 \in \mathcal{S}_1$ and $p_2 \in \mathcal{S}_2$ such that $(p_1, p_2) \in Q$ it holds that:*

1. *if $p_1 \in \mathcal{S}_1^m$, then $p_2 \in \mathcal{S}_2^m$;*
2. *for all $p_1' \in \mathcal{S}_1$ and $a \in \mathcal{A}'$ such that $p_1 \xrightarrow{a}_1 p_1'$, there exists $p_2' \in \mathcal{S}_2$ such that $p_2 \xrightarrow{a}_2 p_2'$ and $(p_1', p_2') \in Q$;*
3. *for all $p_2' \in \mathcal{S}_2$ and $b \in B$ such that $p_2 \xrightarrow{b}_2 p_2'$, there exists $p_1' \in \mathcal{S}_1$ such that $p_1 \xrightarrow{a}_1 p_1'$ and $(p_1', p_2') \in Q$;*

We say that P_1 is partially bisimilar to P_2 with respect to the bisimulation action set B, notation $P_1 \preceq_B P_2$, if there exists a partial bisimulation Q with respect to B such that $(s_1, s_2) \in R$. If $P_2 \preceq_B P_1$ holds as well, then P_1 and P_2 are mutually partially bisimilar with respect to B and we write $P_1 \leftrightarrow_B P_2$.

Note that $\preceq_B$ is a preorder relation, making $\leftrightarrow_B$ an equivalence relation for all $B \subseteq \mathcal{A}$ [10]. If $B = \emptyset$, then $\preceq_\emptyset$ coincides with strong similarity preorder and $\leftrightarrow_\emptyset$ coincides with strong similarity equivalence [5, 1]. When $B = \mathcal{A}$, both $\preceq_\mathcal{A}$ and $\leftrightarrow_\mathcal{A}$ turn into strong bisimilarity [5, 1].

By adopting partial bisimilarity as a behavioral (semantic) relation, we replace the original language-based equivalence, which also permits the use of nondeterministic automata in the specification. We note that partial bisimilarity also accounts for controllability of nondeterministic plants and control requirements, for more details see [2]. Nonetheless, in the setting of this paper we only consider deterministic plants and control requirements, as they guarantee existence of a unique maximally permissive supervisor. The uniqueness is a prerequisite for the proof of main theorem that enables the embedding of the control requirements. Moreover, our experience, during execution of several other industrial studies [8, 4, 6, 7], points out to no particular need to employ nondeterministic automata for modeling purposes.

Definition 2. *Automaton $P = (\mathcal{S}_P, \mathcal{A}_P, \rightarrow_P, s_P, \mathcal{S}_P^m)$ is deterministic if for all $s, s_1, s_2 \in \mathcal{S}_P$ and $a \in \mathcal{A}_P$ it holds that if $s \xrightarrow{a} s_1$ and $s \xrightarrow{a} s_2$ then $s_1 = s_2$.*

We denote the set of deterministic automata by $\mathcal{D}$. We give properties of partial bisimilarity that help to relate it to the standard notion of controllability.

Proposition 1. *Let $P_1, P_2 \in \mathcal{D}$ with $P_1 = (\mathcal{S}_1, \mathcal{A}_1, \rightarrow_1, s_1, \mathcal{S}_1^m)$ and $P_2 = (\mathcal{S}_2, \mathcal{A}_2, \rightarrow_2, s_2, \mathcal{S}_2^m)$. Then, the following holds:*

1. *if $P_1 \preceq_B P_2$, then $P_1 \mid P \preceq_B P_2 \mid P$ for every $P \in \mathcal{D}$;*
2. *if $P_1 \preceq_B P_2$, then $P_1 \preceq_C P_2$ for every $C \subseteq B$;*

3. *$P_1 \preceq_\emptyset P_2$ if and only if $L(P_1) \subseteq L(P_2)$ and $L_m(P_1) \subseteq L_m(P_2)$;*
4. *if $\mathcal{A}_2 \subseteq \mathcal{A}_1$, then $P_1 \mid P_2 \preceq_\emptyset P_1$ and $P_2 \mid P_1 \preceq_\emptyset P_1$; and*
5. *if $P_1 \preceq_\mathcal{U} P_2$ then $L(P_1)\mathcal{U} \cap L(P_2) \subseteq L(P_1)$.*

Proof. Property *1.* states that the partial bisimilarity preorder is a precongruence for the parallel composition, as shown in [2].

Property *2.* is straightforward, by following Definition 1 [2].

Property *3.* follows from Definition 1 and the definitions of recognized and marked languages, and it has been given explicitly for simulation in [5].

Property *4.* follows directly from the definition of the parallel composition and the fact that $\mathcal{A}_2 \subseteq \mathcal{A}_1$ implies that P_2 can only restrict the transitions of P_1 and, therefore, $L(P_1 \mid P_2) = L(P_2) \cap L(P_1)$ [3] implying $P_1 \mid P_2 \preceq_\emptyset P_1$ and $P_2 \mid P_1 \preceq_\emptyset P_1$ by property *2.*

Property *5.* has been previously stated in [10] in a slightly different context, but having in mind its significance, we will give another proof in this setting. Suppose that $P_1 \preceq_\mathcal{U} P_2$ holds. By Definition 1, there exists a partial bisimulation Q such that $(s_1, s_2) \in Q$. We show that $L(P_1)\mathcal{U} \cap L(P_2) \subseteq L(P_1)$ holds by contradiction. Suppose that there exists a trace $t \in L(P_1)$ such that $tu \in L(P_2)$ for some $u \in \mathcal{U}$, but $tu \notin L(P_1)$. As $t \in L(P_1)$, we have that $s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_n} s_n$ for $s_i \in \mathcal{S}_1$ and $t = a_1 \ldots a_n$. Then, we also have that $s_2 \equiv q_1 \xrightarrow{a_1} \ldots \xrightarrow{a_n} q_n$ for $q_i \in \mathcal{S}_2$. Following Definition 2 we have $(s_i, q_i) \in Q$ for $1 \leq i \leq n$. However, according to Definition 1, $s_n \xrightarrow{u} s'_n$ for some $s'_n \in \mathcal{S}_1$ as $q_n \xrightarrow{u}$, leading to a contradiction. □

As in Section 2, we have that $P, R, S \in \mathcal{D}$ denote the plant, the control requirements, and the supervisor, respectively. Again, the supervised plant is given by $P \mid S$. Intuitively, controllability requires that the uncontrollable transitions of P should be bisimilar to those of $P \mid S$, so that the reachable uncontrollable parts of P and $P \mid S$ are indistinguishable. The controllable transitions of the supervised plant may only be simulated by the ones of the original plant, since some controllable transitions are suppressed by the supervisor. Then, we have that R is controllable, if $R \preceq_\mathcal{U} P$. For nonblocking behavior, we still need to ascertain that $L(R) \subseteq L_m(P)$. In case the behavior defined by the control requirements cannot be achieved, then we have that $P \mid S \preceq_\mathcal{U} P$ and $P \mid S \preceq_\emptyset R$ for some supervisor S. Note that $P \mid S \leftrightarrow_\emptyset S$, i.e., the achievable behavior is identified by the supervisor [3, 9]. Also, note that for deterministic systems this is equivalent to $P \mid S \leftrightarrow_B S$ for every $B \in \mathcal{A}$ [5]. Finally, if M is the maximally permissive supervisor for P and R, then $S \preceq_\emptyset M$ for every other supervisor S of P.

Next, we employ this approach to directly manipulate plant and control requirement components, without having to unravel their recognized languages.

5 Partially-Supervised Plants

We assume that, as in Section 3, the plant and the control requirements are given as sets of parallel synchronizing components and restrictions, respectively. The following theorem states when it is possible to embed a control requirement component in the definition of the plant, without affecting the supervised behavior of the plant.

Theorem 1. *Let $P, R, X, S, T \in \mathcal{D}$ such that $P \mid X \preceq_{\mathcal{U}} P$, S is the maximally permissive supervisor for P with respect to $R \mid X$, and T is the maximally permissive supervisor for $P \mid X$ with respect to R. Then $S \leftrightarrow_{\emptyset} T \mid X$.*

Proof. As S and T are a supervisors for P and $P \mid X$, respectively, we have that

$$P \mid S \preceq_{\mathcal{U}} P \qquad \text{and} \qquad (P \mid X) \mid T \preceq_{\mathcal{U}} (P \mid X).$$

From the assumptions, we have $P \mid X \preceq_{\mathcal{U}} P$. Thus,

$$(P \mid X) \mid T \leftrightarrow_{\mathcal{U}} P \mid (T \mid X) \preceq_{\mathcal{U}} P \mid X \preceq_{\mathcal{U}} P,$$

implying that $T \mid X$ is a supervisor for P. As S is the maximally permissive supervisor for P, we have that $T \mid X \preceq_{\emptyset} S$. On the other hand, from $P \mid S \preceq_{\mathcal{U}} P$ we derive that

$$(P \mid S) \mid X \leftrightarrow_{\mathcal{U}} (P \mid X) \mid S \preceq_{\mathcal{U}} P \mid X$$

implying that S is a supervisor for $P \mid X$. As T is the maximally permissive supervisor for $P \mid X$, we have that $S \preceq_{\emptyset} T$. We show that $T \preceq_{\emptyset} T \mid X$, which implies that $S \leftrightarrow_{\emptyset} T \mid X$. Using that $(P \mid X) \mid T \leftrightarrow_{\emptyset} T$, since T is a supervisor for $P \mid X$, we derive:

$$\begin{array}{ll}
(P \mid X) \mid T \preceq_{\emptyset} P \mid X & \text{implies} \\
((P \mid X) \mid T) \mid (T \mid X) \preceq_{\emptyset} (P \mid X) \mid (T \mid X) & \text{implies} \\
(P \mid (X \mid X)) \mid (T \mid T) \preceq_{\emptyset} ((P \mid X) \mid T) \mid X & \text{implies} \\
(P \mid X) \mid T \preceq_{\emptyset} ((P \mid X) \mid T) \mid X) & \text{implies} \\
T \preceq_{\emptyset} T \mid X. &
\end{array}$$

As $S \leftrightarrow_{\emptyset} T \mid X$, we conclude that S and T deliver the same supervised behavior for P and $P \mid X$ with respect to $R \mid X$ and R, respectively. We validate that the requirements are satisfied accordingly. We have the following derivation:

$$\begin{array}{ll}
(P \mid X) \mid T \preceq_{\emptyset} R & \text{implies} \\
((P \mid X) \mid T) \mid X \preceq_{\emptyset} R \mid X & \text{implies} \\
(P \mid (X \mid X)) \mid T \preceq_{\emptyset} R \mid X & \text{implies} \\
(P \mid X) \mid T \preceq_{\emptyset} R \mid X & \text{implies} \\
P \mid (T \mid X) \preceq_{\emptyset} R \mid X, &
\end{array}$$

i.e., $T \mid X$ satisfies the control requirements for P. Also, we derive:

$$(P \mid X) \mid T \leftrightarrow_{\emptyset} (P \mid (X \mid X)) \mid T \leftrightarrow_{\emptyset} (P \mid X) \mid (T \mid X) \leftrightarrow_{\emptyset} (P \mid X) \mid S \preceq_{\emptyset} R,$$

implying that the requirements are satisfied for both supervisors.

Finally, we show that the marked behavior of $P \mid S$ and $(P \mid X) \mid T$ is equivalent, i.e., $L_m(P \mid S) = L_m((P \mid X) \mid T)$. First, note that for every $P_1, P_2 \in \mathcal{D}$, if $t \in L_m(P_1 \mid P_2)$, then $t \in L_m(P_1)$ and $t \in L_m(P_2)$. Suppose that $t \in L_m(P \mid S)$, implying that $t \in L_m(P)$ and $t \in L_m(S)$. Then, by Proposition 1, $t \in L_m(R \mid X)$ and, thus, $t \in L_m(R)$ and $t \in L_m(X)$. Now, it is not difficult to observe that $t \in L_m(P \mid T \mid X)$, as $S \leftrightarrow_{\emptyset} T \mid X$. The other direction is analogous, which completes the proof. $\square$

Using the result of Theorem 1 we can define the notion of partially-supervised plants as specifications that embed a portion of the control requirements in them. Such practice removes trivial and intuitive control requirements that require duplication of information and increase both the readability and meaningfulness of both the plant and control requirements.

The essential requirement of Theorem 1 is that $P \mid X \preceq_{\mathcal{U}} P$ must hold. However, such a requirement may prove difficult to check. We note that it is not necessary to take the whole plant into account, but it is sufficient to prove the claim for a portion of it. To show this, assume that $P \leftrightarrow_{\mathcal{A}} P_1 \mid P_2$ and X is such that $P_1 \mid X \preceq_{\mathcal{U}} P_1$. Then, according to Proposition 1, $(P_1 \mid X) \mid P_2 \preceq_{\mathcal{U}} P_1 \mid P_2$ holds as well, implying that $P \mid X \preceq_{\mathcal{U}} P$. Next, we characterize two cases that can be easily checked by visual inspection.

Let $E \in \mathcal{D}$ represent a totally unrestricted behavior given by $E = (\{s_E\}, \mathcal{A}, \rightarrow_E, s_E, \{s_E\})$, where $s_E \xrightarrow{a} s_E$ for every $a \in \mathcal{A}$. Then, $P \mid E \leftrightarrow_{\mathcal{A}} P$ for every $P \in \mathcal{D}$. Now, if $X \preceq_{\mathcal{U}} E$, then X is a suitable control requirement component for embedding. Visually, one needs only to verify that X does not disable any uncontrollable events in its alphabet.

A typical situation arises when the control requirements need to restrict the occurrence of controllable self loops as shown in Section 3. This requires a duplication of the plant component, while selectively adding transitions with controllable self-loop events (if not already present in the alphabet of the automaton at hand) and/or restricting their occurrences as desired, compare, e.g., Fig. 3 and Fig. 4. We show that in that case the control requirement component can be taken as a plant component.

Let $P \mid K$ be the plant, with $K = (\mathcal{S}_K, \mathcal{A}_K, \rightarrow_K, s_K, \mathcal{S}_K^m)$ a plant component. Let $L = (\mathcal{S}_K, \mathcal{A}_L, \rightarrow_L, s_K, \mathcal{S}_K^m)$ be a control requirement corresponding to this component with $\mathcal{A}_L = \mathcal{A}_K \cup C$, where $C \subseteq \mathcal{C} \setminus \mathcal{A}_K$ and $\rightarrow_L = \rightarrow_K \cup \{(s, c, s') \in \mathcal{S}_K \times C \times \mathcal{S}_K \mid s = s'\}$. Augment the transition relation of K with self loops of events in C for every state in $\mathcal{S}_K$ given by $U = \{(s, c, s) \mid s \in \mathcal{S}_K,\ c \in C\}$. Denote the augmented automaton as K'. It is straightforward that $P \mid K \leftrightarrow_{\mathcal{A}} P \mid K'$, cf. Section 2. Moreover, it is easy to see that $L \mid K' \preceq_{\mathcal{U}} K'$, so that we can replace the updated plant component K' by L, as shown above for P_1 and X.

Using the results above, we can now adapt the specifications of the plant and the control requirements given in Section 3. First, we directly replace the plant components of Fig. 3, Fig. 5, and Fig. 7 with their corresponding control requirements of Fig. 4, Fig. 6, and Fig. 8. These control requirements restrict controllable self-loop events, so we can safely omit them, following the above analysis. Thus, we are eliminating duplication of information without losing essential behavior as the plant specification leaves the control loops for the sake of being consistent with the paradigm of supervisory control theory. As directly witnessed, this leads to unnecessary complication, whereas the partially-supervised plant behavior as given by Fig. 4, Fig. 6, and Fig. 8 is intuitive and would be directly modeled by modelers with insight to the matter at hand.

This leaves us only with the control requirements regarding the safety of movement, depicted in Fig. 10, which can be considered as the only "meaning-

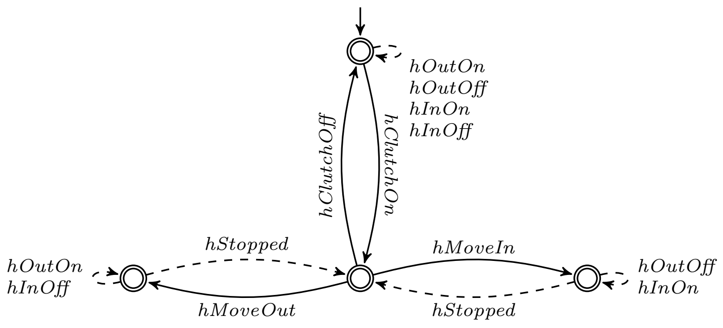

Fig. 11. Embedding the behavior of the clutch in the actuator-sensor relation.

ful" set of control requirements. Here, we embed the requirement regarding the behavior of the clutch, i.e., "Safety manual movement restriction" of Fig. 10, with the plant component describing actuator-sensor relationship of Fig. 9. The result of such an embedding is depicted in Fig. 11, which clearly shows the relation between the horizontal actuators and sensors. Namely, the horizontal motor must be stopped in order to allow manual operation and in that case every sensor event is possible. In case the patient support system is in motorized mode, i.e., the clutch is applied, the sensor events can only occur consistently with the horizontal movement of the table. This behavior cannot be easily deduced from the "crowded" plant component depicted in Fig. 9, so the analysis reveals clearly the intended behavior of the system.

We conclude that partially-supervised plants contribute to clarity and meaningfulness of supervisory control specifications, when they are employed to eliminate trivial and cluttered control requirements. We also demonstrated that they can help better understand the underlying behavior. However, we must warn that ad-hoc modeling should not replace supervisory control, i.e., the conditions of Theorem 1 must be verified before applying the method presented in this section. Furthermore, checking that the preconditions of the theorem hold, actually amounts to supervisory control synthesis in some cases. For that purpose, we characterized two simple instances of the Theorem 1, that are often found and applied in practice without formalizing the underlying process of thought.

6 Concluding Remarks

We introduced the notion of partially-supervised plants that embed control requirements in their components. The main motivator for such an embedding is that by strictly following the principles of supervisory control, we sometimes end up with cluttered and "redundant" plant components and control requirements. Moreover, we noticed that embedding of control requirements actually occurs ad hoc during the modeling of the plant. We have shown under which conditions this embedding is safe and does not alter the supervised behavior of the plant.

We also gave a simple characterization, which can be verified visually, of the two most intuitive and most applied situations. We have demonstrated the new concept in an industrial study involving supervision of a patient support system for an MRI scanner. We have shown that there is considerable improvement of the readability and the meaningfulness of the specifications of the plant and control requirements.

To show that the embedding of the control requirements does not alter the supervised behavior of the plant, we employed a process-theoretic approach that captures the notion of controllability by means of a behavioral preorder. The preorder, termed partial bisimilarity, has been adapted for deterministic automata as employed in standard supervisory control. By analyzing the proof of the main theorem, one also observes the ease of manipulation with the underlying notions, which further validates our approach to supervisory control theory.

As future work, we intend to deepen our understanding of nondeterministic partially-controlled plants, as there does not exist a unique maximally permissive supervisor. We will also investigate state-based supervisory control, where the control requirements refer to states, instead of supplying traces of events.

References

1. Baeten, J.C.M., Basten, T., Reniers, M.A.: Process Algebra: Equational Theories of Communicating Processes, Cambridge Tracts in Theoretical Computer Science, vol. 50. Cambridge University Press (2010)
2. Baeten, J.C.M., van Beek, D.A., Luttik, B., Markovski, J., Rooda, J.E.: A process-theoretic approach to supervisory control. In: Proceedings of ACC 2011. IEEE (2011), to appear. See `http://se.wtb.tue.nl`.
3. Cassandras, C., Lafortune, S.: Introduction to discrete event systems. Kluwer Academic Publishers (2004)
4. Forschelen, S.T.J.: Supervisory control of theme park vehicles. Master's thesis, Systems Engineering Group, Eindhoven University of Technology (2010)
5. Glabbeek, R.J.v.: The linear time–branching time spectrum I. Handbook of Process Algebra pp. 3–99 (2001)
6. Leijenaar, J.F.: Supervisory Control of Document Processing Machines. Master's thesis, Systems Engineering Group, Eindhoven University of Technology (2009)
7. Markovski, J., Jacobs, K.G.M., van Beek, D.A., Somers, L.J.A.M., Rooda, J.E.: Coordination of resources using generalized state-based requirements. In: Proceedings of WODES 2010. pp. 300–305. IFAC (2010)
8. Petreczky, M., van Beek, D.A., Rooda, J.E.: Supervisor for toner error handling: a case study in supervisory control of Océ printers. SE Report 2008-011, Eindhoven University of Technology, Systems Engineering Group (2008), available from `http://se.wtb.tue.nl/sereports`.
9. Ramadge, P.J., Wonham, W.M.: Supervisory control of a class of discrete event processes. SIAM Journal on Control and Optimization 25(1), 206–230 (1987)
10. Rutten, J.J.M.M.: Coalgebra, concurrency, and control. SEN Report R-9921, Center for Mathematics and Computer Science, Amsterdam, The Netherlands (1999)
11. Theunissen, R., Schiffelers, R., van Beek, D., Rooda, J.: Supervisory control synthesis for a patient support system. In: Proceedings of 10th European Control Conference. pp. 1–6. EUCA (2009)