



Control of Discrete-Event Systems with Partial Observations Using Coalgebra and Coinduction

JAN KOMENDA

komenda@ipm.cz

Institute of Mathematics-Brno Branch, Czech Academy of Sciences, Zirkova 22, 616 62 Brno, Czech Republic

JAN H. VAN SCHUPPEN

J.H.van.Schuppen@cwil.nl

CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract. Control of discrete-event systems with partial observations is treated by concepts and results of coalgebra and coinduction. Coalgebra is part of abstract algebra and enables a generalization of the computer science concept of bisimulation. It can be applied to automata theory and then provides a powerful algebraic tool to treat problems of supervisory control. A framework for control of discrete-event systems with partial observations is formulated in terms of coalgebra. The contributions to control theory are besides the framework, algorithms for supremal normal and supremal normal and controllable sublanguages of the plant.

Keywords: supervisory control, coalgebra, bisimulation, coinduction, partial observations

1. Introduction

The purpose of the paper is to present for control of discrete-event systems with partial observations a coalgebraic framework, concepts for supervisory control, theorems for the existence of supervisors, and algorithms for the computation of supremal normal and controllable sublanguages.

Supervisory control is motivated by the need for control algorithms for communication networks, automated cars, railway control systems, failure diagnosis in heating and ventilation systems, etc. For this purpose the logical operation of engineering systems is described by automata or related models. Since the early 1980's, W.M. Wonham and many other researchers have developed supervisory control theory, see (Ramadge and Wonham, 1989; Wonham and Ramadge, 1987; Lin and Wonham, 1988, 1990; Brandt et al., 1990; Brandt et al., 1990; Yoo et al., 2001; Cho and Marcus, 1989a; Overkamp and van Schuppen, 2000), etc. both for finite and infinite strings. Control with partial observations is motivated by the engineering experience that not all events can be observed and hence cannot be used for control.

Coalgebra is a topic of abstract algebra. The term was already used in the 1960's. Algebraists did not consider the concept useful until the appearance of the theorem on the existence of a final coalgebra, see Aczel and Mendler (1989). Coalgebra has been used by R. Milner and other computer scientists in the form of bisimulation. The main results of coalgebra used are a coinduction proof and definition principle and the implication that the existence of a bisimulation between two elements of final coalgebras implies their equality.

The computer scientist J.J.M.M. Rutten has formulated a framework for control of discrete-event systems with complete observations using coalgebra (Rutten, 1999). This paper is an extension of that framework to control with partial observations.

The contributions of this paper to supervisory control are the following. Primary is the coalgebraic framework for control of discrete-event systems with partial observations. This includes the concepts of a weak transition and that of an observability relation which result in a characterization of observability of a sublanguage. A theorem states an equivalent condition for the existence of a supervisor based on partial observations in terms of a controllability relation and of an observability relation. These results are the coalgebraic equivalence of the theorems published in (Lin and Wonham, 1988). Because supremal sublanguages with partial observations do not exist, the concept of a normality relation is defined as in the approach of Lin and Wonham (1988). A characterization of a normal relation is stated. New algorithms are stated for the computation of supremal normal (and normal and controllable) sublanguages. Their computational complexity is compared with that of the currently available algorithms.

The advantages of the coalgebraic framework for supervisory control with partial observations compared with the Ramadge-Wonham framework are primarily: (1) the conceptual clarity of the framework; (2) the simplified proofs; and (3) the algorithms for supremal languages mentioned above. In particular, the natural algorithmic character of the concepts and algorithms and the proof technique of working with relations, are advantages. Since the first draft of this paper several papers were written on modular control of discrete-event systems in which new results were obtained and which establish the usefulness of this framework. Results for decentralized control using coalgebra have also been obtained (Komenda and van Schuppen, 2003).

The contribution of this paper when compared to the literature is twofold. Firstly, the full framework of discrete-event control with partial observations introduced in Lin and Wonham (1988) has been reformulated using the concepts from coalgebra and concurrency theory. Secondly, our approach provides a refinement of the existing theory and yields new algorithms and useful concepts. Among the new results, novel algorithms for computation of supremal normal and supremal normal and controllable sublanguages have been proposed. Moreover, Algorithms 4 and 5 are *single-step*, which can be by itself considered as an important result. Unlike the known algorithms for supremal normal and controllable sublanguages, in particular those developed by Cho and Marcus (1989a) and by Yoo and Lafortune in Yoo et al. (2001) which are iterations of two separate algorithms, our Algorithm 5 is 'single-step.' The concept of supervised product is central in our framework, because its coinductive definition describes in fact an event by event action of the supervisor. This considerably simplifies the study of properties of closed-loop languages compared to the classical algebraic approach.

The paper is organized as follows. The introductory Section 2 presents a verbal motivation and short description of our approach to the supervisory control which is accessible to a reader who is not familiar with abstract algebra. Section 3 presents concepts from automata theory, bisimulation, category theory, coalgebra, and coinduction. It also recalls the partial automata from Rutten (1999) as the coalgebraic framework for DES represented by automata. The reader interested in more details about the key notions like bisimulation, coinduction, and finality should consult Rutten (2000),

Gumm (2003) or Rutten (1999). In Section 4 weak transition structures are defined on partial automata, powerset, projected, and observer automata are introduced using a deterministic notion of weak transitions. Observability relations are introduced in Section 5 and normality relations in Section 6. These relational characterizations are then used in Section 7 to derive new algorithms for computing the supremal normal (and normal and controllable) sublanguages. In Section 4 necessary and sufficient conditions for a given language to be exactly achieved are captured in a relation called partial bisimulation. Finality is used to define the language of the closed-loop system as well as the infimal-closed observable superlanguage. In Section 5 an algorithm is presented for the computation of an observable sublanguage that contains the supremal normal sublanguage. In Section 6 simple ‘single-step’ algorithms for the computation of supremal normal (resp. normal and controllable) sublanguages are presented: Algorithm 4 (resp. Algorithm 5). Preliminary results of this paper have been presented in Komenda (2002a,b) without proofs. The results of the Sections 4, 5, and 6 complete the algebraic results of (Lafortune and Chen (1990), Rudie and Wonham (1990), and Cho and Marcus (1989b)). In Section 7 the distributivity of the supervised product with respect to basic language operations is studied.

2. Problem formulation and approach

The problem of control of discrete-event systems with partial observations is formulated in and related to coalgebra. The presentation in this section is exclusively verbal, a mathematical framework is developed from Section 3 onwards. For readers not previously familiar with the material used in this paper, iteratively re-reading Sections 2 and 3 may prove to be useful.

2.1. Discrete-event systems and supervisory control

For engineering control problems there is an increased interest to control the logical operations of engineering systems. For the order of the distinct operations mathematical and computer science models of discrete-event systems are used such as automata, Petri nets, and process algebras. This differs from classical control theory where the times are an essential part of the model.

Supervisory control developed by P. Ramadge and W.M. Wonham and coworkers is now a well established theory for control of discrete-event systems. Control problems address the control objectives of safety, resource allocation, liveness, and fault diagnosis (Cassandras and Lafortune, 1999). A supervisor restricts the behavior of a discrete-event system such that the control objectives (safety, resource allocation, liveness) are met or such that faults are diagnosed.

The supervisor affects the plant by disabling or enabling a subset of the available events, the subset of controllable events. The interconnection of the system and the

supervisory is called the closed-loop system. The supervisory control problem is then to synthesize a supervisor such that the closed-loop system meets the prespecified control objectives.

2.2. *Supervisory control with partial observations*

A discrete-event system is said to have partial observations if not all events are observed and hence not all events are available to the supervisor or controller. The events which are observed are called the observable events. There are also events that are not observed, unobservable events. Examples of such events are failures of a machine or operations in a communication network where the local events are not communicated to a distant observer station. In general, the events are not observable because this is impossible or costly (requires extra sensors and communication capabilities). Supervisory control with partial observations is then to synthesize a supervisor based on partial observations only such that the closed-loop system meets the prespecified control objectives. Control with partial observations is highly relevant to engineering because not all events are observed. The problem is one of the most difficult one of all control problems and there are many concepts and results missing, not only for control of discrete-event systems with partial observations.

2.3. *Coalgebra*

The term coalgebra was used by S. Eilenberg in a 1965 paper (Eilenberg and Moore, 1965). Algebraists did not consider the concept useful until the appearance of a proof on the existence of a final coalgebra, see Aczel and Mendler (1989). The computer scientist R. Milner has used bisimulation for automata and process algebras since 1980 and this is a special case of coalgebra. Since then bisimulation and coalgebra are extensively used in computer science. Coalgebra has been used on other parts of control and system theory since about 1990, see the papers by R. Grossman (Grossman and Larson, 1992).

Briefly, an algebra can be considered, in terms of category theory, as a map from a functor of a set to the corresponding set. A coalgebra is then defined as a map from a set to a functor of the set. A coalgebra is called final if there exists a unique map from every coalgebra to the final coalgebra.

A theorem of coalgebra is that if the coalgebra is final then any bisimulation on the product of two sets implies equality of the sets. Another useful theorem is that one can prove existence of an object via coinduction on final coalgebras. Coinduction corresponds to induction as coalgebra corresponds to algebra.

In this paper coalgebra is used as a framework for control of discrete-event systems. To keep the paper elementary, no category theory is used at all except for the introduction of the concept of coalgebra in Section 3. The reader need not have a background in coalgebra to read the paper. The only results used are that the existence of a bisimulation on lan-

guage subsets implies equality of the subsets and that new objects (operations) on such subsets can be constructed by coinduction.

2.4. A sample of the results

As was mentioned in the introduction, one of the contributions of the paper is the reformulation of control of discrete-event systems with partial observations using coalgebra and concurrence theory.

For example, consider the control with partial observations as treated in the Sections 4 and 5. The tuples of marked and corresponding prefix-closed languages of automata are called partial languages. Consider an automaton, a partial language L representing the plant, and a partial language K representing the closed-loop language. One defines the closed-loop language of the plant under supervision by coinduction, see Definition 8.4 where the definition is stepwise as in the definition of an automaton. Next one defines a relation between two subsets of partial languages, they are called *partially bisimilar* if the conditions of Definition 8.5 hold. Theorem 8.6 then states that the closed-loop language K equals the supervised product of K and L if and only if the two sublanguages K and L are partially bisimilar. The proof of this theorem consists only of proving that a particular relation defined in the proof is a partial bisimulation relation.

A by product of the coalgebraic framework is that the definitions suggest new algorithms for the supremal controllable sublanguage, the infimal controllable superlanguage, and the infimal observable superlanguage, etc. An example is Algorithm 5 of Section 6 for supremal normal and controllable sublanguages, which normal sublanguages. The coinductive definitions of supervised product, supremal controllable sublanguages and infimal (prefix-closed) observable superlanguages also simplify the study of these important concepts. As an example the distributivity of the supervised product with language unions is studied. It can be seen as the generalization of the known fact that if all controllable events are observable, then observability is preserved by language union, which is a special case of our Theorem 11.1 in view of is ‘single-step’ and simplifies considerably the study of supremal controllable and Theorem 8.6. Indeed, it is sufficient to consider the result of Theorem 11.1 in the special case, where the specification (partial) languages are partially bisimilar with respect to the plant (partial) language. Then these specifications are in particular observable. As a consequence of Theorem 11.1, the union of these specifications is also partially bisimilar with respect to the plant, i.e., in particular observable.

3. Automata and coalgebra

In this section automata are first defined as is done classically. Then coalgebra is defined mathematically and it is shown how automata can be formulated in coalgebraic terms. With every automaton can be associated the partial language which it generates. The subset of partial languages is then given the structure of an automaton. Finally the concept of coinduction is introduced.

3.1. Automata

Automata were used as models of computation from about the 1950's on. For a long while they were taught to computer science students but knowledge of automata theory is currently limited. Textbooks on automata theory include (Hopcroft and Ullman, 1979; Eilenberg, 1974; Sipser, 1997). System theory, the basis of control theory, has been inspired by automata theory. Therefore control theory and automata theory have the same basis.

An automaton is a collection of sets and functions,

$$(Q, E, f, q_0, Q_m),$$

where Q is a finite set called the *state set*, E is a finite set called the *event set*, $f: Q \times E \rightarrow Q$ is a function called the *transition function*, $q_0 \in Q$ is the *initial state*, and $Q_m \subseteq Q$ is the subset of *marked states*. An automaton operates on a string of events and produces a sequence of states, also called a state trajectory:

$$(q_0, q_1, q_2, \dots), q_{i+1} = f(q_i, e_i), \forall i \in \mathbb{N}_+ = \{1, 2, \dots\}.$$

The state trajectory need not stop at a marked state, a marked state signifies only that a subtask has been properly completed.

Instead of an automaton one also defines a generator. Recall that the transition function of an automaton, f , is defined for all discrete states, those in Q , and all events, those in E . It is therefore also called a *total function*. A *generator* is a collection as an automaton above but the transition function is a partial function, for every $q \in Q$ there exists a subset $E(q) \subseteq E$, in general not equal to E , such that $f(q, e)$ is defined for all $q \in Q$ and $e \in E(q)$. Most examples of engineering systems are actually generators rather than automata. In this paper a generator is also called a partial automaton in line with the terminology used by J.J.M.M. Rutten in Rutten (1999).

Automata with outputs can be either Moore automata or Mealy automata. Below Moore automata are defined because they will be used. A Moore automaton is collection of sets and functions,

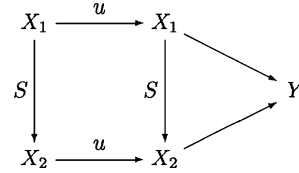
$$(Q, Y, E, f, h, q_0),$$

where, in addition to the definition of a generator, $f: Q \times E \rightarrow Q \cup \{\emptyset\}$ and $f(q, e) = \emptyset$ if there is no transition defined at state q ; Y is a finite set called the *output set*; and $h: Q \rightarrow Y$ is a function called the *output function*. An example is $Y = \{0, 1\}$ and $h(q) = 1$ if the trajectory terminates in q .

3.2. Bisimulation

The concept of a bisimulation has been popularized in computer science by Robin Milner (1989) but is due to D.M.R. Park (1980) for behavioral equivalence of concurrent processes. It is extensively used in computer science. After P. Aczel (1988) published a proof principle for final coalgebras, coalgebras became more popular. A categorical definition of bisimulation is due to P. Aczel and N. Mendler (1989).

To introduce bisimulation for readers who are familiar with control and system theory, consider two finite-dimensional linear systems. If both systems are minimal realizations of the same impulse response function then it is known from the work of R.E. Kalman that there exists a state-space isomorphism S such that the following diagram commutes



A *bisimulation between two automata*

$(X_1, E, f_1, x_{1,0})$, $(X_2, E, f_2, x_{2,0})$, f_1, f_2 total functions,
 is defined as a relation $R \subseteq X_1 \times X_2$ such that
 $\{(x_1, x_2) \in R \text{ and } e \in E\} \Rightarrow (f_1(x_1, e), f_2(x_2, e)) \in R$.

A bisimulation between automata is related to the concept of congruence of a unary algebra. Consider set X . A *congruence* on X of a unary function $f: X \rightarrow X$ is an equivalence relation $R \subseteq X \times X$ such that,

$$(x_1, x_2) \in R \Rightarrow (f(x_1), f(x_2)) \in R.$$

A bisimulation between two generators is defined slightly differently because the transitions do not always exist. A *bisimulation between two generators*

$(X_1, E, f_1, x_{1,0})$, $(X_2, E, f_2, x_{2,0})$, f_1, f_2 , partial functions,
 is defined as a relation $R \subseteq X_1 \times X_2$ such that,

- (1) if $x_{1,e} = f_1(x_1, e)$,
 then there exists $x_{2,e} = f_2(x_2, e)$ and $(x_{1,e}, x_{2,e}) \in R$; equivalently,
 $x_1 \xrightarrow{e} x_{1,e} \Rightarrow \{x_2 \xrightarrow{e} x_{2,e} \text{ and } (x_{1,e}, x_{2,e}) \in R\}$;
- (2) $x_2 \xrightarrow{e} x_{2,e} \Rightarrow \{x_1 \xrightarrow{e} x_{1,e} \text{ and } (x_{1,e}, x_{2,e}) \in R\}$.

A *bisimulation between two Moore automata* is defined as,

$$\begin{aligned}
 M_i &= (X_i, Y, E, f_i, h_i, x_{i,0}), \quad i = 1, 2, \\
 R &\subseteq X_1 \times X_2, \text{ a relation such that,} \\
 (x_1, x_2) \in R &\Rightarrow (1) \forall e \in E, x_1 \xrightarrow{e} \Rightarrow \{x_2 \xrightarrow{e} \text{ and } (x_{1,e}, x_{2,e}) \in R\}; \\
 &\quad (2) \forall e \in E, x_2 \xrightarrow{e} \Rightarrow \{x_1 \xrightarrow{e} \text{ and } (x_{1,e}, x_{2,e}) \in R\}; \\
 &\quad (3) h_1(x_1) = h_2(x_2).
 \end{aligned}$$

A bisimulation on a Moore automaton M is a bisimulation on (M, M) . A *bisimilarity relation* on a Moore automaton M is defined as,

$$\begin{aligned}
 \sim &\subseteq X \times X, \\
 x_1 \sim x_2 &\quad \exists \text{a bisimulation } R \subseteq X \times X, \text{ bisimulation such that } (x_1, x_2) \in R.
 \end{aligned}$$

The following statements can then be proven: (a) Unions and compositions of bisimulations on Moore automata are bisimulations. (b) Bisimilarity, \sim , is the largest bisimulation on Moore automaton. (c) Bisimilarity, \sim , is an equivalence relation. (d) $k : X_1 \rightarrow X_2$ is a homomorphism between Moore automata if and only if its graph, $\{(x, k(x)) \in X \times X\}$, is a bisimulation on a Moore automaton. Note that the existence of a bisimulation implies that the two Moore automata are realizations of the same input–output map but not conversely. There is a problem related to nondeterministic Moore automata, where there is not a unique minimal (up to isomorphism) realization. As a consequence, there is no final object among all Moore automata in general and the concept of coinduction fails. Therefore we consider in the sequel only deterministic Moore automata, which are more suitable for our coalgebraic treatment.

3.3. Category theory

Bisimulation can now be applied directly. However, more is learned if the concept is seen from the view point of category theory. Category theory is not used in the remainder of the paper and the reader not interested in the topic may skip this subsection. Category theory was formulated by S. MacLane and S. Eilenberg with as motivation to have a rather general mathematical structure into which many algebraic structures could be embedded.

A *category* consists of a class of objects \mathbf{C} ; $\forall A, B \in \mathbf{C}$, a set $\text{hom}(A, B)$ of morphisms, $f : A \rightarrow B$; and $\forall A, B, C \in \mathbf{C}$, a map $\text{comp} : \text{hom}(A, B) \times \text{hom}(B, C) \rightarrow \text{hom}(A, C)$, $\text{comp}(f, g) = g \circ f$, such that

- (a) *existence of unit*: $\forall A \in \mathbf{C}$ there exists $i_A \in \text{hom}(A, A)$ such that $f \circ i_A = f$ and $i_A \circ g = g$;
- (b) *associativity* holds: if $f \in \text{hom}(A, B)$, $g \in \text{hom}(B, C)$, $h \in \text{hom}(C, D)$, then $h \circ (g \circ f) = (h \circ g) \circ f$;
- (c) if $(A, B) \neq (C, D)$ then $\text{hom}(A, B) \cap \text{hom}(C, D) = \emptyset$.

[Jacobson (1980), Volume 2, Chapter 1].

An example of a category is that of the category of sets, say *(Sets, Maps)*, formed by a collection of sets with as morphisms the maps between sets, say for the sets $A, B \in \text{Sets}$, $\text{hom}(A, B) = \{f : A \rightarrow B\}$. The composition of maps then forms the composition of the category and the identity maps on sets form the identities of the categories.

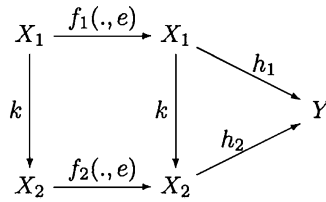
Category theory works with functors that operate between categories. Consider sets U, V , the set of maps $H(U, V) = \{f : U \rightarrow V\}$, and the operation $F : U \rightarrow V$.

The operation F is said to be *functor* if (1) when the operation can be lifted to morphisms, e.g., to functions on sets, there exists a function $F_H : H(U, U) \rightarrow H(V, V)$ and (2) the function F_H preserves identity maps and compositions. Such an operation on sets is also called functorial. A special case is where $F : U \rightarrow U$, e.g., $U = \text{Set}$ and F is called a set functor.

For the remainder of this subsection, attention is restricted to the category of Moore automata. The final category will play a major role for the interpretation of bisimulation.

A *homomorphism* k of Moore automata

$M_1 = (X_1, Y, E, f_1, h_1, x_{1,0})$, $M_2 = (X_2, Y, E, f_2, h_2, x_{2,0})$,
 is a map $k : X_1 \rightarrow X_2$, such that,
 $k(f_1(x_1, e)) = f_2(k(x_1), e)$, $h_1(x_1) = h_2(k(x_1))$;
 equivalently, the following diagram commutes,



A system theoretic interpretation is that the existence of a homomorphism between Moore automata M_1, M_2 implies that M_2 is realization of the input–output behavior of M_1 and $\text{card}(M_2) \leq \text{card}(M_1)$. One may call M_2 a realization of lower order than M_1 .

It is a theorem that the set of Moore automata with the set of homomorphisms of Moore automata forms a category

$$\begin{aligned}
 & (\mathbf{M}(\mathbf{E}), \{\text{hom}(X, X'), X, X' \in M(E)\}), \\
 \mathbf{M}(\mathbf{E}) &= \{M = (X, Y, E, f, h, x_0) \mid M \text{ is Moore automaton}\} \\
 \text{hom}(\mathbf{X}, \mathbf{X}') &= \{k : X \rightarrow X' \mid \text{homomorphism of } X, X'\}, \forall X, X' \in M(E).
 \end{aligned}$$

3.4. Algebras and coalgebras

An F -algebra is a tuple (U, c) consisting of,

$$\begin{array}{ll}
 U & \text{a set, called the carrier set,} \\
 c : F(U) \rightarrow U & \text{the operation of the algebra.}
 \end{array}$$

A F -coalgebra is a tuple (U, c) consisting of,

$$\begin{array}{ll}
 U & \text{a set, called the carrier set,} \\
 c : U \rightarrow F(U) & \text{the operation of the coalgebra.}
 \end{array}$$

Let us denote $1 = \{\emptyset\}$. As an example consider the functor $F = (f, h)$

$$Q \mapsto F(Q) = (Q + 1)^E \times Y.$$

Then a Moore automaton (X, Y, E, f, h) can be viewed as $(X, (f, h))$, i.e., as an F -coalgebra.

Consider functor F . (a) A *homomorphism of F -coalgebras* from a F -coalgebra (X_1, c_1) to a F -coalgebra (X_2, c_2) is a function $f: X_1 \rightarrow X_2$ such that $c_2 \circ f = F(f) \circ c_1$, or commutativity holds in the diagram,

$$\begin{array}{ccc}
 X_1 & \xrightarrow{f} & X_2 \\
 c_1 \downarrow & & \downarrow c_2 \\
 F(X_1) & \xrightarrow{F(f)} & F(X_2)
 \end{array}$$

(b) A *final F -coalgebra* (X_f, c_f) is a F -coalgebra such that for every F -coalgebra (X, c) there exists a unique homomorphism $f: X \rightarrow X_f$ of F -coalgebras.

It is then a theorem that (a) The identity map of a F -coalgebra (X, c) is a homomorphism of F -coalgebras. (b) Compositions of homomorphisms of F -coalgebras are homomorphisms of F -coalgebras. (c) A final F -coalgebra is unique up to isomorphism. It is now also possible to define a bisimulation between coalgebras but this concept will not be defined in this paper because it will not be used directly.

3.5. Automata in terms of coalgebra

Below generators introduced above are formulated in a coalgebraic framework. This was first done by J.J.M.M. Rutten, who called them *partial automata*, and his framework will be used in this paper. The transition function can be viewed as a coalgebraic map together with the output function that determines the subset of marked states.

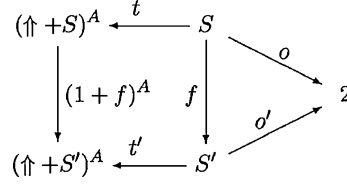
Now we recall from Rutten (1999) partial automata as coalgebras of a special functor in the category of sets with functions as morphisms. Let A be an arbitrary set (usually finite and referred to as the set of inputs or events). The free monoid of words (strings) over A is denoted by A^* . The empty string will be denoted by ε . Denote by $\uparrow = \{\emptyset\}$ the one element set and by $2 = \{0, 1\}$ the set of Booleans.

A *partial automaton* is a pair $S = (S, \langle o, t \rangle)$, where S is a set of states, and a pair of functions $\langle o, t \rangle: S \rightarrow 2 \times (\uparrow + S)^A$, consists of an output function $o: S \rightarrow 2$ and a transition function $S \rightarrow (\uparrow + S)^A$. The output function o indicates whether a state $s \in S$ is accepting (or terminating) : $o(s) = 1$, denoted also by $s \downarrow$, or not: $o(s) = 0$, denoted by $s \uparrow$. The transition function t associates to each state s in S a function $t(s): S \rightarrow (\uparrow + S)$. The set $\uparrow + S$ is the disjoint union of S and \uparrow . The meaning of the state transition function is that $t(s)(a) = \emptyset$ iff $t(s)(a)$ is undefined, which means that there is no a -transition from the state $s \in S$. $t(s)(a) \in S$ means that the a -transition from s is possible and we define in this case $t(s)(a) = s_a$, which is denoted mostly by $s \xrightarrow{a} s_a$. This notation can be extended by induction to arbitrary strings in A^* . Assuming that $s \xrightarrow{w} s_w$ has been defined, define $s \xrightarrow{wa} s_{wa}$ iff $t(s_w)(a) \in S$, in which case $s_{wa} = t(s_w)(a)$, also denoted by $s \xrightarrow{wa} s_{wa}$. It is easy to see that partial automata are coalgebras of the set functor $F = 2 \times (\uparrow + (\cdot))^A$.

A *homomorphism* between partial automata $S = (S, \langle o, t \rangle)$ and $S' = (S', \langle o', t' \rangle)$ is a function $f: S \rightarrow S'$ with, for all $s \in S$ and $a \in A$:

$$o'(f(s)) = o(s) \text{ and } s \xrightarrow{a} s_a \text{ iff } f(s) \xrightarrow{a} f(s_a),$$

in which case: $f(s)_a = f(s.1_a)$.



A partial automaton $S' = (S', \langle o', t' \rangle)$ is a *subautomaton* of $S = (S, \langle o, t \rangle)$ if $S' \subseteq S$ and the inclusion function $i: S' \rightarrow S$ is a homomorphism. It is important to notice that the coalgebraic concept of subautomaton corresponds to the notion of strict subautomaton in Cho and Marcus (1989a). In the sequel we use always subautomata in the coalgebraic sense defined above, i.e., strict subautomata are meant.

A *simulation* between two partial automata $S = (S, \langle o, t \rangle)$ and $S' = (S', \langle o', t' \rangle)$ is a relation $R \subseteq S \times S'$ with, for all $s \in S$ and $s' \in S'$:

$$\text{if } \langle s, s' \rangle \in R \text{ then } \begin{cases} (i) & o(s) \leq o(s'), \text{ i.e., } s \downarrow \Rightarrow s' \downarrow, \text{ and} \\ (ii) & \forall a \in A : s \xrightarrow{a} \Rightarrow (s' \xrightarrow{a} \text{ and } \langle s_a, s'_a \rangle \in R), \end{cases}$$

A *bisimulation* between two partial automata $S = (S, \langle o, t \rangle)$ and $S' = (S', \langle o', t' \rangle)$ is a relation $R \subseteq S \times S'$ with, for all $s \in S$ and $s' \in S'$:

$$\text{if } \langle s, s' \rangle \in R \text{ then } \begin{cases} (i) & o(s) = o(s'), \text{ i.e., } s \downarrow \text{ iff } s' \downarrow \\ (ii) & \forall a \in A : s \xrightarrow{a} \Rightarrow (s' \xrightarrow{a} \text{ and } \langle s_a, s'_a \rangle \in R), \text{ and} \\ (iii) & \forall a \in A : s' \xrightarrow{a} \Rightarrow (s \xrightarrow{a} \text{ and } \langle s_a, s'_a \rangle \in R). \end{cases}$$

We write $s \sim s'$ whenever there exists a bisimulation R with $\langle s, s' \rangle \in R$. This relation is the union of all bisimulations, i.e., the greatest bisimulation also called bisimilarity. It is immediate from the definition of bisimulation that two states are bisimilar iff they can make the same transitions and they give rise to the same outputs:

PROPOSITION 3.1 For any partial automaton $S = (S, \langle o, t \rangle)$ and any $s, s' \in S$:

$$s \sim s' \text{ iff } \forall w \in A^* : s \xrightarrow{w} \Leftrightarrow s' \xrightarrow{w}, \text{ in which case } o(s_w) = o'(s'_w).$$

3.6. Final automaton of partial languages

In this subsection an automaton is defined which is a final automaton in the sense of category theory defined above. For the remainder of the paper it is important that the automaton considered is final. This makes then available the theorems of coinduction and

of proofs of equality of partial languages by existence of a bisimulation. The final automaton is not the standard one described earlier but one in which the main object is the language generated by the automaton.

Below a partial automaton of a partial languages is defined over an alphabet (input set) A , denoted by $\mathcal{L} = (\mathcal{L}, \langle o_{\mathcal{L}}, t_{\mathcal{L}} \rangle)$. More formally, $\mathcal{L} = \{\Phi : A^* \rightarrow (\uparrow + 2) \mid \text{dom}(\Phi) = \{w \in A^* \mid \Phi(w) \in 2\} \neq \emptyset \text{ is prefix-closed}\}$. To each partial language Φ a pair $\langle V, W \rangle$ can be assigned: $W = \text{dom}(\Phi)$ and $V = \{w \in \text{dom}(\Phi) \mid \Phi(w) = 1\{\in 2\}\}$. Conversely, to a pair $\langle V, W \rangle \in \mathcal{L}$, a function Φ can be assigned: $\Phi(w) = 1$ if $w \in V$, $\Phi(w) = 0$ if $w \in W$ and $w \notin V$, and $\Phi(w)$ is undefined if $w \notin W$. Therefore we can write:

$$\mathcal{L} = \{\langle V, W \rangle \mid V \subseteq W \subseteq A^*, W \neq \emptyset, \text{ and } W \text{ is prefix-closed}\}.$$

The transition function $t_{\mathcal{L}} : \mathcal{L} \rightarrow (1 + L)^A$ is defined using input derivatives. Recall that for any partial language $L = (L^1, L^2) \in \mathcal{L}$, $L_a = (L_a^1, L_a^2)$, where $L_a^i = \{w \in A^* \mid aw \in L^i\}$, $i = 1, 2$. If $a \notin L^2$ then L_a is undefined. Given any $L = (L^1, L^2) \in \mathcal{L}$, the partial automaton structure of \mathcal{L} is given by:

$$o_{\mathcal{L}}(L) = \begin{cases} 1 & \text{if } \varepsilon \in L^1 \\ 0 & \text{if } \varepsilon \notin L^1 \end{cases} \text{ and } t_{\mathcal{L}}(L)(a) = \begin{cases} L_a & \text{if } L_a \text{ is defined} \\ \emptyset & \text{otherwise} \end{cases}.$$

Notice that if L_a is defined, then $L_a^1 \notin L_a^2$, $L_a^2 \neq \emptyset$, and L_a^2 is prefix-closed. The following notational conventions will be used: $L \downarrow$ iff $\varepsilon \in L^1$, and $L \xrightarrow{w} L_w$ iff L_w is defined (iff $w \in L^2$).

3.7. Induction and coinduction

Induction is taught to undergraduate students in courses of algebra. The student learns that all elements of a sequence indexed by the natural numbers satisfy a specified property if (1) the first element of the sequence satisfies it and (2) if element $n \in \mathbb{N}$ satisfies the property then so does element $n + 1$. This can be put in a more abstract setting as the proper definition of a function from the natural numbers to a set corresponding to the property concerned, and illustrated by a commutative diagram. Most of mathematics students only remember the simple sufficient condition and not the abstract setting.

Coinduction is a dual concept to induction. Many people use induction without bearing in mind its abstract (categorical or universally algebraic) meaning. Coinduction in its full generality must be put into a general framework of universal coalgebra that uses the category theory. Finality of a coalgebra enables coinductive definitions and proofs in a similar way as initiality of an algebra enables definitions and proofs by induction. In order to make the paper more accessible to a reader not very familiar with category theory we have preferred to introduce the coinduction only in its special form: on final coalgebra of partial languages. It is the same as with mathematical induction that is by many people understood only on the initial algebra of natural numbers with the (unary algebraic) structure given by the successor operation: $\forall n \in \mathbb{N} : \text{succ}(n) = n + 1$. Here definitions of functions by induction correspond to giving the successor on functions, hence yielding

recursive formulas. Proofs by induction correspond to the very well known two-steps procedure, which amounts to verify that a relation is a congruence relation with respect to the successor operation. This is possible because natural numbers with the successor operation is the initial algebra in the category of all unary algebras, i.e., there is a unique morphism from the initial algebra of natural numbers to any unary algebra.

Similarly, a definition by coinduction amounts to give the corresponding structure, here output and derivatives on operations to be defined, and a proof by coinduction consists in verifying the conditions of bisimulation relation. We believe that giving a general categorical definition of coinduction would go far beyond the scope of the paper, the purpose of the paper is primarily control of discrete-event systems with coalgebra. Coinduction has been well covered by the existing literature on universal coalgebra (Rutten, 2000, 2003).

Coinduction is used as a proof and definition principle throughout this paper. The use of coinduction is limited to final coalgebras. Behavior equivalence of two elements of final coalgebra means that these are equal. Also notice that the elements of final coalgebras are equal to their behaviors (the identity is the unique behavior homomorphism). This feature is sometimes paraphrased as ‘being is doing,’ because these elements behave as they are.

Proofs by coinduction consist in constructing appropriate relations: for instance a proof of equality of two elements of a final coalgebra consists in finding a bisimulation relation that relates them. Definition by coinduction of an operation on elements of a final coalgebra consists in defining the same coalgebraic structure on the operation (for instance we define binary operations on partial languages by defining derivatives and output functions further in this paper). More details about coinduction and finality can be found in Rutten (2000) or Rutten (1999). Various supervisory control and observation problems will be tackled using coinduction. It offers more than just an insight to some well known solutions of these problems, it leads to some new algorithms and results.

3.8. Final automata and coinduction

Most of the rest of this subsection is recalled from Rutten (1999).

THEOREM 3.2 \mathcal{L} satisfies the principle of coinduction: for all K and L in \mathcal{L} , if $K \sim L$ then $K = L$.

Proof: It follows from Proposition 3.1. Indeed, if $K \sim L$ then for any $w \in A^* : K \xrightarrow{w} \Leftrightarrow L \xrightarrow{w}$, i.e., $w \in K^2$ iff $w \in L^2$, in which case $o(K_w) = o'(K'_w)$ i.e., $w \in K^1$ iff $w \in L^1$. It follows that $K = L$. The converse implication is also true. ■

THEOREM 3.3 The partial automaton $\mathcal{L} = (\mathcal{L}, \langle o_{\mathcal{L}}, t_{\mathcal{L}} \rangle)$ is final among all partial automata: for any partial automaton $S = (S, \langle o, t \rangle)$ there exists a unique homomorphism $l : S \rightarrow \mathcal{L}$. This homomorphism identifies bisimilar states: for $s, s' \in S : l(s) = l(s')$ iff $s \sim s'$.

Proof: For the existence part of the theorem, we define the homomorphism l by putting for $s \in S$:

$$\text{dom}(l(s)) = \{w \in A^* : s \xrightarrow{w}\}$$

and

$$l(s) = \left((l(s))^1, (l(s))^2 \right) = (\{w \in A^* \mid s \xrightarrow{w} \text{ and } s_w \downarrow\}, \{w \in A^* \mid s \xrightarrow{w}\}).$$

Uniqueness of l follows from the fact that for any two homomorphisms $l, l' : S \rightarrow \mathcal{L}$ the relation

$$R = \{\langle l(s), l'(s) \rangle \in \mathcal{L} \times \mathcal{L} \mid s \in S\}$$

is a bisimulation. Therefore $l = l'$ follows from Theorem 3.2. The last statement is immediate from the definition of l and Proposition 3.1. ■

We adopt the notation from Rutten (1998), page 9, easily extended from automata to partial automata, and denote the minimal (in size of the state set) representation of a partial language L by $\langle L \rangle$. Hence, $\langle L \rangle = (DL, \langle o_{\langle L \rangle}, t_{\langle L \rangle} \rangle)$ is a subautomaton of \mathcal{L} generated by L . This means that $o_{\langle L \rangle}$ and $t_{\langle L \rangle}$ are uniquely determined by the corresponding structure of \mathcal{L} . The carrier set of this minimal representation of L is denoted by DL , where $DL = \{L_u \mid u \in L^2\}$. Let us call this set the set of derivatives of L . Inclusion of partial languages that corresponds to a simulation relation is meant componentwise. The prefix closure of an (ordinary) language L is denoted by \bar{L} . Some further notation from Rutten (1999) is used, e.g., ‘zero’ (partial) language is denoted by 0 , i.e., $0 = (\emptyset, \{\varepsilon\})$.

There is yet another important concept that will be needed in this paper. Namely, given an (ordinary) language L , the suffix closure of L is defined by $\text{suffix}(L) = \{s \in A^* \mid \exists u \in A^* \text{ with } us \in L\}$. For partial languages, the suffix closure is defined in the same way as the prefix closure, i.e., componentwise. There is the following relation between the transition structure of L and its suffix closure operator.

Observation 3.4: For any (partial) language L : $\text{suffix}(L) = \cup_{u \in L^2} L_u$.

Proof: It is immediate from the fact that $L_u = (\{s \in A^* \mid us \in L^1\}, \{s \in A^* \mid us \in L^2\})$. ■

4. Weak transition structures

Control with partial observations implies that the observed traces are different from those with complete observations. This motivates the concept of weak transitions.

In the following definition we introduce the notion of weak derivative (transition). Roughly speaking it disregards unobservable steps, which correspond to so called internal moves in the framework of process algebras (Milner, 1989b). Let $A = A_o \cup A_{uo}$ be a

partition of A into observable events (A_o) and unobservable (A_{uo}) events with the natural projection $P : A^* \rightarrow A_o^*$. Recall that $P(a) = \varepsilon$ for any $a \in A_{uo}$, $P(a) = a$ for $a \in A_o$, and P is catenative.

DEFINITION 4.1 (*Nondeterministic weak transitions.*) For an event $a \in A$ define $L \xRightarrow{P(a)}$ iff $\exists s \in A^* : P(s) = P(a)$ and $L \xrightarrow{s} L_s$. Denote in this case $L \xRightarrow{P(a)} L_s$.

Remark 4.1: According to this notation for unobservable events $L \xRightarrow{\varepsilon}$ is an abbreviation for $\exists \tau \in A_{uo}^*$ such that $L \xrightarrow{\tau}$. We admit $\tau = \varepsilon$, hence $L \xRightarrow{\varepsilon}$ is always true. For $a \in A_o$ our notation means that there exist $\tau, \tau' \in A_{uo}^*$ such that $L \xrightarrow{\tau a \tau'} L_{\tau a \tau'}$. This definition can be extended to strings (words in A^*) in the following way:

$$L \xRightarrow{P(s)} \text{ iff } \exists t \in A^* : P(s) = P(t) \text{ and } L \xrightarrow{t} L_t. \text{ Denote in this case } L \xRightarrow{P(s)} L_t.$$

There may exist two or more $s \in A^*$ satisfying the condition in the definition of weak transition. Hence, the weak transition structure introduced above is not deterministic. We introduce deterministic weak transition structure on \mathcal{L} in the following definition.

DEFINITION 4.2 (*Deterministic weak transitions.*) Define for $a \in A_o : L \xrightarrow{a} L_{\hat{a}}$ iff $L \xRightarrow{P(a)}$ and $L_{\hat{a}} := \cup_{\{s \in L^2 | P(s) = a\}} L_s$.

To avoid any confusion we must distinguish between both concepts. Let us introduce the convention that for nondeterministic weak transition we say that $L \xRightarrow{P(a)} L'$ for some L' and for deterministic concept we denote always the unique weak a -derivative by $L_{\hat{a}}$. For ε -weak transitions we introduce the notation $L \xRightarrow{\varepsilon} L_{uo}$, where $L_{uo} = \cup \{L_{\tau} : \tau \in A_{uo}^* \text{ such that } L_{\tau} \text{ exists}\}$, the latter set being nonempty ($\varepsilon \in A_{uo}^*$). Sometimes it will be denoted for notational convenience also by $L_{\hat{\varepsilon}}$, i.e., $L_{\hat{\varepsilon}} = L_{ou}$ is the so called unobservable reach of the partial language L . Notice that for any $L \in \mathcal{L}$, L_{uo} has the pleasant property that for $a \in A_o : L_{uo} \xRightarrow{P(a)} L_{uo} \xrightarrow{a}$.

The concept of deterministic weak transitions can be extended to observable strings by induction. It should be clear that for $s \in A_o^* : L \xrightarrow{s} L_{\hat{s}}$ iff $L \xRightarrow{P(s)}$ with $L_{\hat{s}} = \cup_i \{L_t \mid t \in L^2 \text{ and } P(t) = s\}$. Otherwise stated $L_{\hat{s}} = \cup \{L' \mid L \xRightarrow{P(s)} L'\}$.

There is the following relation between the deterministic weak transitions of a language L and the (strong) transition structure of the projected language $P(L)$ over alphabet A_o .

PROPOSITION 4.2 For any (partial) language L and $s \in A_o^* : P(L) \xrightarrow{s} P(L)_s$ iff $L \xrightarrow{s} L_{\hat{s}}$, in which case $P(L)_s = P(L)_{\hat{s}}$.

Proof: The first part is easy. Indeed, $P(L) \xrightarrow{s} P(L)_s$ iff $s = P(s) \in P(L)^2$ iff $\exists u \in L^2 : P(u) = P(s)$ iff $L \xRightarrow{P(s)}$, which is equivalent to $L_{\hat{s}}$ exists, i.e., $L \xrightarrow{s} L_{\hat{s}}$. In order to see the second part, observe that $t \in P(L_{\hat{s}})$ iff $\exists w \in L_{\hat{s}}$ with $t = P(w)$ iff $\exists w \in L_r$ for $P(r) = s$ and $w \in P^{-1}(t)$ iff $\exists r w \in L$ with $r \in P^{-1}(s)$ and $w \in P^{-1}(t)$ iff $P^{-1}(s)P^{-1}(t) \cap L = P^{-1}(st) \cap L \neq \emptyset$. On the other hand $t \in P(L)_s$ iff $st \in P(L)$ iff $\exists u \in L : P(u) = st$ iff $\exists u \in L \cap P^{-1}(st)$ iff

$P^{-1}(st) \cap L \neq \emptyset$. Since this is valid for both components of the languages involved, this achieves the proof of the proposition. ■

Notice that deterministic weak derivatives can be generated by strong derivatives and ε -weak derivatives.

PROPOSITION 4.3 *For any partial language L and $a \in A_o : L_{\hat{a}} = ((L_{\hat{\varepsilon}})_a)_{\hat{\varepsilon}}$.*

Proof: It is immediate from Definition 4.2 and the fact that $P^{(-1)}(a) = \{\tau a \tau' \mid \tau, \tau' \in A_{uo}^*\}$. ■

Much more can be said about the topic of weak transition. In particular, our notion of deterministic weak transition gives rise to another concept of weak bisimulation, where usual nondeterministic weak transitions (Milner, 1989b) are replaced by the deterministic ones. However, due to Proposition 5 it is not difficult to show that the new concept would coincide with the equality of projections, i.e., observable trace equivalence. The notion of weak bisimulation can be defined unlike deterministic weak transitions for any partial automaton using the concept of powerset automaton introduced in the next subsection. But this goes beyond the aimed scope of the present paper. On the other hand weak bisimulation is not a congruence, but only a weak congruence, and therefore does not provide (strong) quotients. The same problem can be encountered in the framework of process algebras (Milner, 1989a).

4.1. Weak transitions and observers for partial automata

Weak transitions in the final automaton of partial languages have been introduced. Let us extend our definition to arbitrary partial automata. As for the nondeterministic concept of weak transitions, the definition is straightforward (simply for a state s in partial automaton $S = (S, \langle o, t \rangle)$ and $a \in A$ we put $s \xrightarrow{P(a)} s'$ if there exists $u \in A^*$ such that $P(u) = P(a)$ and $s \xrightarrow{u} s' = s_u$). As for the deterministic concept of weak transitions, the corresponding definition does not make sense in general, however it can be defined if the set of states is a powerset. This motivates the following construction, where we denote the set of nonempty subsets of S by $Pwr^+(S) (= Pwr(S) \setminus \emptyset)$.

DEFINITION 4.3 (Powerset automaton.) *To any partial automaton $S = (S, \langle o, t \rangle)$ we assign a powerset automaton, a partial automaton denoted by $Pwr(S) = (Pwr^+(S), \langle o_S, t_S \rangle)$, where for any $Q \subseteq S$; $Q \neq \emptyset$ we put*

$$t_S(Q)(a) = \cup_{q \in Q} t(q)(a) \text{ and } o_S(Q) = \max(o(q), q \in Q).$$

Notice that in the definition of transition function in a powerset automaton there is no necessity to consider separately the case when $t(q)(a)$ is not defined for some $q \in Q$, because according to the definition of partial automata for such a case there is $t(q)(a) =$

$\emptyset \in \uparrow$. Therefore the above compact way of defining t_S is correct. If we denote by $l_S : Pwr^+(S) \rightarrow \mathcal{L}$ and $l : S \rightarrow \mathcal{L}$ the unique homomorphisms defined by finality of \mathcal{L} , then clearly $l_S(Q) = \bigcup_{q \in Q} l(q)$. This enables us to use the same notation for l and l_S , i.e., the subscript S can be dropped.

In order to implement the projections we define the projected automaton.

DEFINITION 4.4 (Projected automaton). *The projected automaton is a partial automaton over A_o :*

$$P(S) = (Pwr^+(S), \langle o_P, t_P \rangle) \text{ with } t_P(Q)(a) = \bigcup_{\{w \in A^* \mid P(w)=a\}} t_S(Q)(w), \quad a \in A_o$$

and $o_P(Q) = o_S(Q_{uo}) = \max\{o(q), q \in Q_{uo}\}$, where Q_{uo} is the unobservable reach set of Q , i.e., $Q_{uo} = \{q' \in S \mid \exists q \in Q \text{ with } q \Rightarrow q'\}$.

If we denote by $l_P : Pwr^+(S) \rightarrow \mathcal{L}_o$ the unique homomorphism defined by finality of \mathcal{L}_o (automaton of partial languages over A_o), then clearly $l_P(Q) = P(l(Q))$.

Deterministic weak transitions can now be defined in powerset automata: for any $\emptyset \neq Q \subseteq S$ and $a \in A_o$: $Q \xrightarrow{a} Q_a = \bigcup_{\{u \in P^{-1}(a)\}} t_S(Q)(u)$. Notice in particular that deterministic weak transitions in the powerset automaton $Pwr(S)$ correspond exactly to strong transitions in the projected automaton $P(S)$, i.e., for any $\emptyset \neq Q \subseteq S$: $t_P(Q)(a) = Q_a$.

The projected automaton $P(S)$ of a given automaton S is related to the observer automaton [see e.g., Cassandras and Lafortune (1999)], but its state space is in general much larger than that of the observer automaton. In control theory, the observer automaton is defined by induction starting from the initial state. However, partial automata as defined above have no initial state. Note that it is natural to consider $L \in \mathcal{L}$ itself as the initial state of the minimal recognizer $\langle L \rangle$ of $L \in \mathcal{L}$. For general S , if we are given an initial state, the usual construction of observer has its coalgebraic meaning. We define for each partial automaton with designated initial state $S = (S, \langle o, t \rangle)$ with $s_0 \in S$ the observer automaton as the subautomaton of $P(S)$ generated by $\{s_0\}_{uo}$ (accessible from $\{s_0\}_{uo}$):

DEFINITION 4.5 (Observer automaton.) *The observer automaton is denoted by $Obs(S) = (S_{obs}, \langle o_{obs}, t_{obs} \rangle)$ with carrier set $S_{obs} \subseteq Pwr^+(S)$ defined as follows:*

- 1) $\{s_0\}_{uo} \in S_{obs}$ ($\{s_0\}_{uo}$ —unobservable reach set is that of $Pwr(S)$).
- 2) If $Q \in S_{obs}$ then $\forall a \in A_o$: $t_P(Q)(a) \in S_{obs}$.

The structure of the observer automaton is given by the structure of $P(S)$ restricted to S_{obs} : $\forall Q \in S_{obs}$: $o_{obs}(Q)$ and $\forall a \in A_o$: $t_{obs}(Q)(a) = t_P(Q)(a)$.

Remark 4.4: Notice that the definition above implies that the states of the observer are isomorphic to, i.e., can be identified with, different deterministic weak derivatives of its initial state $\{s_0\}_{uo}$, i.e., we have $S_{obs} = \{(s_0)_d : t_P(\{s_0\}_{uo})(d) \text{ is defined}\}$. Note that if it happens that two deterministic weak derivatives are equal, they determine a single state of the observer automaton.

5. Observability relation

In the supervisory control of DES with partial observations the observability of a (specification) language with respect to the plant and projection (to observable events) is necessary for achieving this language as a desirable behavior of the closed-loop system. We assume that $A = A_c \cup A_{uc}$ is a partition of A into controllable events (A_c) and uncontrollable (A_{uc}) events. The observability condition has been first introduced in Lin and Wonham (1988) using a slightly different, but equivalent formulation. This notion of observability is very different from the observability of linear systems. The first attempt to capture this type of condition in an abstract setting goes back to Wonham (1976). There has been yet another approach to the observability of DES, based on an automata theoretic framework. A necessary condition for a given specification represented by an automaton to be achieved has been formulated in Bergeron (1993) using automata framework.

DEFINITION 5.1 (*Observability.*) *A partial language K is said to be observable with respect to another partial language L (with $K \subseteq L$) and projection P if for all $s \in K^2$ and $a \in A_c$ the following implication holds true:*

$$sa \in L^2, s'a \in K^2, \text{ and } P(s) = P(s') \Rightarrow sa \in K^2.$$

Our aim is to find a relational characterization of observability. Unlike (Komenda, 2002b) we present first definitions on automata representations of K and L without specialization to relations on \mathcal{L} . The following auxiliary relation is needed.

DEFINITION 5.2 (*Observational indistinguishability relation on S .*) *A binary relation $Aux(S)$ on S , called observational indistinguishability relation is the smallest relation satisfying:*

- (i) $\langle s_0, s_0 \rangle \in Aux(S)$
- (ii) *If $\langle s, t \rangle \in Aux(S)$ then $\forall a \in A : (s \xrightarrow{P(a)} s' \text{ for some } s' \text{ and } t \xrightarrow{P(a)} t' \text{ for some } t') \Rightarrow \langle s', t' \rangle \in Aux(S)$*

From the definition of weak transitions it follows that (ii) is equivalent to (ii)' and (iii)' below:

- (ii)' *If $\langle s, t \rangle \in Aux(S)$ then: $(s \xrightarrow{\varepsilon} s' \text{ for some } s' \text{ and } t \xrightarrow{\varepsilon} t' \text{ for some } t') \Rightarrow \langle s', t' \rangle \in Aux(S)$*
- (iii)' *If $\langle s, t \rangle \in Aux(S)$ then $\forall a \in A_o : (s \xrightarrow{a} s_a \text{ and } t \xrightarrow{a} t_a) \Rightarrow \langle s_a, t_a \rangle \in Aux(S)$.*

$Aux(S)$ can be characterized by the following lemma.

LEMMA 5.1 *For any $s, s' \in S : \langle s, s' \rangle \in Aux(S)$ iff there exist two strings $w, w' \in K^2$ such that $P(w) = P(w')$ and $s = (s_0)_w$ and $s' = (s_0)_{w'}$.*

Proof: (\Leftarrow) Let $s, s' \in S$ such that there exist two strings $w, w' \in K^2$ such that $P(w) = P(w')$ and $s = (s_0)_w$ and $s' = (s_0)_{w'}$. Let $w = w_1 \dots w_n$ and $w' = t_1 \dots t_m$. Let $P(w) = P(w') =$

$a_1 \dots a_k$. Then $n > k$ and $m \geq k$ and there exists two increasing sequences of integers (indices) $u_i \geq i, i = 1, \dots, k$ and $v_i \geq i, i = 1, \dots, k$ such that $a_i = w_{u_i}$ and $a_i = t_{v_i}$. Since all a_i are observable events we can write $s_0 \xrightarrow{P(a_1) \dots P(a_k)} s$ and $s_0 \xrightarrow{P(a_1) \dots P(a_k)} s'$, whence by (ii) inductively applied $\langle s, s' \rangle \in Aux(S)$.

(\Rightarrow) Let $\langle s, s' \rangle \in Aux(S)$. Then by the construction of $Aux(S)$ there exist $a_1, \dots, a_k \in A$ such that $s_0 \xrightarrow{P(a_1) \dots P(a_k)} s$ and $s_0 \xrightarrow{P(a_1) \dots P(a_k)} s'$. Therefore there exist by definition of nondeterministic weak transitions two strings w, w' with the same projection such that $s = (s_0)_w$ and $s' = (s_0)_{w'}$. ■

Remark 5.2: Remark that $Aux(S)$ is not in general an equivalence relation, because it might be nontransitive as is shown in the following example. However it is always symmetric and reflexive. Such a relation is sometimes called a tolerance relation.

Example 1: Take $S = DK$, where $K = (\emptyset, \{\bar{a}\tau, \tau\})$. Then $DK = \{K, K_a, K_\tau\}$. Then $Aux(DK)$ is not an equivalence relation, because $K_\tau = K_{a\tau} = \{\varepsilon\}$ means that $\langle K, K_\tau \rangle \in Aux(DK)$ and $\langle K_\tau, K_a \rangle \in Aux(DK)$, while $\langle K, K_a \rangle \notin Aux(DK)$.

A natural question arises under which conditions $Aux(S)$ is an equivalence relation. Recall the concept of state-partition automaton from Cho and Marcus (1989a) and Cassandras and Lafortune (1999).

DEFINITION 5.3 (State-partition automaton). Let $S = (S, \langle o, t \rangle)$ be a partial automaton and let $Obs(S) = (S_{obs}, \langle o_{obs}, t_{obs} \rangle)$ be its observer automaton. Then S is said to be a state-partition automaton if for all $Q_1, Q_2 \in S_{obs} \subseteq Pwr(S)$ we have: $Q_1 \neq Q_2 \Rightarrow Q_1 \cap Q_2 = \emptyset$.

A partial automaton S with initial state s_0 is a state-partition automaton if any two different states of the observer are disjoint (as subsets of S). In the case, where all states of the automaton S are accessible from s_0 , this condition is equivalent to the statement that the states of the observer automaton form a partition of S . In our coalgebraic framework, the property of state-partition automaton can be described in terms of deterministic weak derivatives (in the sense of $Pwr(S)$). Namely, S is a state-partition automaton if $\forall d, d' \in P(l(s_0)^2) : (s_0)_d \neq (s_0)_{d'} \Rightarrow (s_0)_d \cap (s_0)_{d'} = \emptyset$. It is easy to prove that this condition is sufficient for $Aux(S)$ to be an equivalence relation.

PROPOSITION 5.3 *If S is a state-partition automaton then $Aux(S)$ is an equivalence relation.*

Proof: Let S be a state-partition automaton. Let us show that $Aux(S)$ is transitive. Take s, s', s'' such that $\langle s, s' \rangle \in Aux(S)$ and $\langle s', s'' \rangle \in Aux(S)$. Let us show that $\langle s, s'' \rangle \in Aux(S)$. There exist strings v, v', w, w' such that $P(v) = P(v'), P(w) = P(w'), s = (s_0)_v, s' = (s_0)_{v'}, s' = (s_0)_w, s'' = (s_0)_{w'}$. Denote $d = P(v)$ and $d' = P(w)$. Then $s' \in (s_0)_d \cap (s_0)_{d'}$. This means that $(s_0)_d \cap (s_0)_{d'} \neq \emptyset$ and by definition of state-partition automaton $(s_0)_d = (s_0)_{d'}$. In particular, there exists w'' with $P(w'') = P(w)$ and $s = (s_0)_{w''}$. Thus $\langle s, s'' \rangle \in Aux(S)$. ■

However the opposite statement does not hold as the following example shows.

Example 2: Put $S = \{s_0, s_1, s_2\}$ with the transition function $t(s_0)(a) = s_2$, $t(s_0)(\tau) = s_1$, $t(s_1)(\tau) = s_2$, $t(s_2)(\tau) = s_1$, the other transitions are undefined, and the output function can be arbitrary. Then $Aux(S) = S^2$ is trivially an equivalence relation on S , but S is not a state partition automaton, because the sets $(s_0)_{uo} = S$ and $(s_0)_{\hat{a}} = \{s_1, s_2\}$ violate the condition for S to be a state-partition automaton.

Lemma 5.3 implies that $\langle s, s' \rangle \in Aux(S)$ iff there exists $d \in P(L)$ ($d = P(w) = P(w')$) such that $s \in (s_0)_{\hat{a}}$ and $s' \in (s_0)_{\hat{a}}$, i.e., there exists a state $Q \subseteq S$ of the observer automaton $Obs(S)$ such that $s \in Q$ and $s' \in Q$. This motivates the following definition.

DEFINITION 5.4 Define for any $s \in S$:

$$[s]_{Aux(S)} = \{s' \in S : \langle s, s' \rangle \in Aux(S)\}.$$

Let us now use a simpler notation $ls]_{Aux}$ if automaton S is supposed to be fixed. It is to be expected that

Observation 5.4: The following properties are equivalent:

- (i) $Aux(S)$ is an equivalence relation
- (ii) $\forall s, s' \in S : \langle s, s' \rangle \in Aux(S) \Rightarrow ls]_{Aux} = ls']_{Aux}$.
- (iii) $\forall s, s' \in S : ls]_{Aux} \cap ls']_{Aux} \neq \emptyset \Rightarrow ls]_{Aux} = ls']_{Aux}$.

Proof: (i) \Rightarrow (ii) Let $Aux(S)$ be an equivalence relation and take arbitrary $s, s' \in S$ such that $ls]_{Aux} \neq ls']_{Aux}$. Assume that $\exists q \in S : q \in ls]_{Aux} \setminus ls']_{Aux}$, the other case can be treated in a symmetric way. By definition of $ls]_{Aux}$, $\langle s, q \rangle \in Aux(S)$ and $\langle q, s' \rangle \notin Aux(S)$. Let us show that $\langle s, s' \rangle \notin Aux(S)$. Suppose by contradiction that $\langle s, s' \rangle \in Aux(S)$, then using the fact that $Aux(S)$ is symmetric and transitive, $\langle s', q \rangle \in Aux(S)$, hence also $\langle q, s' \rangle \in Aux(S)$, a contradiction. Therefore $\langle s, s' \rangle \in Aux(S)$.

(ii) \Rightarrow (iii) Let the implication (ii) hold true. We show (iii) : if for $s, s' \in S : ls]_{Aux} \cap ls']_{Aux} \neq \emptyset$, then there exists $q \in S$ such that $q \in ls]_{Aux} \cap ls']_{Aux}$, i.e., $\langle s, q \rangle \in Aux(S)$ and $\langle q, s' \rangle \in Aux(S)$. Thus from (ii) we have $ls]_{Aux} = ls']_{Aux} = ls']_{Aux}$, which was to be shown.

(iii) \Rightarrow (i) Let the implication (iii) hold true. Take $\langle s, s' \rangle \in Aux(S)$, and $\langle s', s'' \rangle \in Aux(S)$. Then $s' \in ls]_{Aux} \cap ls'']_{Aux} \neq \emptyset$. It follows that $ls]_{Aux} = ls'']_{Aux}$. But $s \in ls]_{Aux} \cap ls'']_{Aux}$, i.e., $\langle s, s'' \rangle \in Aux(S)$ according to the definition of $ls'']_{Aux}$. This proves the transitivity of $Aux(S)$. ■

Note that in fact $ls]_{Aux} = \cup_{\{d: s \in (s_0)_{\hat{a}}\}} (s_0)_{\hat{a}}$. It follows that $\langle s, s' \rangle \in Aux(S)$ iff $\{s, s'\} \subseteq Q$, for some $Q \in S_{obs}$, which is equivalent by definition of the observer to $\exists d \in A_o^* : \{s, s'\} \subseteq (s_0)_{\hat{a}}$.

One could ask for conditions that ensure that $Aux(S)$ is an equivalence relation without the use of $Aux(S)$ itself. Let $M = l(s_0)^2$ be the closed behavior generated by s_0 . There is the following condition using the states of $Obs(S)$:

LEMMA 5.5 *Aux(S) is an equivalence relation iff $\forall d_1, d_2 \in P(M) : (s_0)_{\tilde{d}_1} \cap (s_0)_{\tilde{d}_2} \neq \emptyset \Rightarrow (\forall s_1 \in (s_0)_{\tilde{d}_1} \text{ and } \forall s_2 \in (s_0)_{\tilde{d}_2}) \exists d \in P(M) : \{s_1, s_2\} \subseteq (s_0)_{\tilde{d}}$.*

Proof: (\Rightarrow) Let $Aux(S)$ be an equivalence relation and suppose by contradiction that $\exists d_1, d_2 \in P(M) : (s_0)_{\tilde{d}_1} \cap (s_0)_{\tilde{d}_2} \neq \emptyset$ and $\exists s_1 \in (s_0)_{\tilde{d}_1}$ and $\exists s_2 \in (s_0)_{\tilde{d}_2} : \forall d \in P(M) : \{s_1, s_2\} \not\subseteq (s_0)_{\tilde{d}}$. It means that there exists $q \in S : q = (s_0)_{v_1} = (s_0)_{v_2}$, where $P(v_1) = d_1$ and $P(v_2) = d_2$, i.e., $\langle s_1, q \rangle \in Aux(S)$ and $\langle q, s_2 \rangle \in Aux(S)$. By the transitivity of $Aux(S)$, $\langle s_1, q \rangle \in Aux(S)$ and $\langle q, s_2 \rangle \in Aux(S)$ implies $\langle s_1, s_2 \rangle \in Aux(S)$, i.e., there exists $d \in P(M) : s_1 \in (s_0)_{\tilde{d}}$ and $s_2 \in (s_0)_{\tilde{d}}$, a contradiction with $\forall d \in P(M) : \{s_1, s_2\} \not\subseteq (s_0)_{\tilde{d}}$. Thus the implication holds true.

(\Leftarrow) Assume the implication on the right hand side holds true, $\langle s, s' \rangle \in Aux(S)$, and $\langle s', s'' \rangle \in Aux(S)$. By Lemma 5.1 there exists strings v, v', w, w' such that $s = (s_0)_v$, $s' = (s_0)_{v'} = (s_0)_w$, $s'' = (s_0)_w$ where $P(v) = P(v')$, and $P(w) = P(w')$. Denote by $d_1 = P(v)$ and $d_2 = P(w)$. Then $s \in (s_0)_{\tilde{d}_1}$. Therefore for $s'' \in (s_0)_{\tilde{d}_1}$ and $s \in (s_0)_{\tilde{d}_2}$ there exists $d \in P(M) : s \in (s_0)_{\tilde{d}}$ and $s'' \in (s_0)_{\tilde{d}}$, i.e., $\langle s, s'' \rangle \in Aux(S)$ using Lemma 5.1, and $Aux(S)$ is an equivalence relation. ■

Notice that the condition of Lemma 5.5 is similar but somewhat weaker than the condition required for S to be a state-partition automaton, which is only a sufficient condition for $Aux(S)$ to be an equivalence relation.

Our aim now is to provide a coalgebraic characterization of observability. Since observability is a property of the second (closed) components of K and L , we can assume that $S_1 = (S_1, \langle o_1, t_1 \rangle)$ is a partial automaton with initial state $s_0 \in S$ that represents K in the sense $K = l_1(s_0)$, $l_1 : S_1 \rightarrow \mathcal{L}$ being the unique behavior homomorphism defined by finality of \mathcal{L} . Moreover, since $K \subseteq L$, we can assume that S_1 is a subautomaton of $S = (S, \langle o, t \rangle)$ with $L = l(s_0)$ ($l : S \rightarrow \mathcal{L}$ is the behavior homomorphism) and s_0 their common initial state. Let the transition function of S be denoted by \rightarrow , i.e., $s \xrightarrow{a} s_a$ means $s_a = t(s)(a)$ and similarly the transition function t_1 of S_1 is denoted by \rightarrow_1 , i.e., $s \xrightarrow{a}_1 s_a^1$ means $s_a^1 = t_1(s)(a)$. Notice also that due to the requirement that S_1 is a subautomaton of S , we have in fact $s_a^1 = s_a \in S_1$. It means that the superscript ¹ can be dropped here. Let us introduce observability relations, in which the observational indistinguishability relation is involved.

DEFINITION 5.5 (Observability relation.) *A binary relation $O(S_1, S)$ on $S_1 \times S$ is called the observability relation if for any $\langle s, t \rangle \in O(S_1, S)$ the following items hold:*

- (i) $\forall a \in A : s \xrightarrow{a}_1 s_a \Rightarrow t \xrightarrow{a} t_a \text{ and } \langle s_a, t_a \rangle \in O(S_1, S)$
- (ii) $\forall a \in A_c : t \xrightarrow{a} t_a \text{ and } (\exists s' : \langle s, s' \rangle \in Aux(S_1) : s' \xrightarrow{a}_1 s'_a) \Rightarrow s \xrightarrow{a}_1 s_a \text{ and } \langle s_a, t_a \rangle \in O(S_1, S)$.

Remark that (ii) can be expressed using the set $s_0 \lfloor s \rfloor Aux(s_1)$ introduced above: the condition $\exists s' : \langle s, s' \rangle \in Aux(S_1) : s' \xrightarrow{a}_1 s'_a$ can be replaced by the simpler one $\langle s \rangle_{Aux(S_1)} \xrightarrow{a}_1$, where \rightarrow_1 is now to be interpreted in $Pwr(S_1)$. For $s \in S_1$ and $s' \in S_1$ we write $s \approx_{O(S_1, S)} s'$ whenever there exists an observability relation $O(S_1, S)$ on $S_1 \times S$ such that $\langle s, s' \rangle \in O(S_1, S)$. Now we are ready to prove:

THEOREM 5.6 *A (partial) language K is observable with respect to $L(K \subseteq L)$ and P iff $s_0 \approx_{O(S_1, S)} s_0$.*

Proof: (\Rightarrow) Let K be observable with respect to L and P such that $K \subseteq L$. Denote

$$O(S_1, S) = \{ \langle (s_0)_u, (s_0)_v \rangle \in S_1 \times S \mid u \in K^2 \} \subseteq S_1 \times S.$$

Note that some of the pairs in $O(S_1, S)$ can be equal. Indeed, it is possible that there exists $u, v \in K^2 : (s_0)_u = (s_0)_v$. But we will show that this is not a problem for our proof. Let us show that $O(S_1, S)$ is an observability relation. Let $\langle q, r \rangle \in O(S_1, S)$. Because of the definition of $O(S_1, S)$ we can assume that $q = (s_0)_s$ for some $s \in K^2$ and $r = q$. We must show that conditions (i) and (ii) are satisfied.

- (i) Let $q \xrightarrow{a}_1$ for $a \in A$. Clearly $q \xrightarrow{a}$, because S_1 is a subautomaton of S and it is immediate from the definition of $O(S_1, S)$ that $\langle q_a, q_a \rangle \in O(S_1, S)$.
- (ii) Let $q \xrightarrow{a}$ for $a \in A_c$ and $\exists q' : \langle q, q' \rangle \in Aux(S_1) : q' \xrightarrow{a}_1$. Then by Lemma 5.1 there exist two strings $s', s'' \in K^2$ such that $P(s'') = P(s')$, $q = (s_0)_{s''}$, and $q' = (s_0)_{s'}$. Now $q' \xrightarrow{a}_1$ implies that $s'a \in K^2$. Recall that $s' \in K^2$, because $q' = (s_0)_{s'}$. From $q \xrightarrow{a}$ and $q = (s_0)_{s''}$ follows $s''a \in L^2$ and by application of the observability of K with respect to L and P we deduce $s''a \in K^2$, i.e., $a \in l((s_0)_{s''})^2$, which means that $q = (s_0)_{s''} \xrightarrow{a}_1$. It follows from (i) that $\langle q_a, q_a \rangle \in O(S_1, S)$. We see now that considering s'' instead of s , where $q = (s_0)_s = (s_0)_{s''}$ did not make any difference.

(\Leftarrow) Let $s_0 \approx_{O(S_1, S)} s_0$. Let us show that K is observable with respect to L and P . For this purpose, let $s \in K^2$, $s'a \in K^2$ for $a \in A_c$, $sa \in L^2$, and $P(s) = P(s')$. Then $s \in K^2 \cap L^2$, i.e., $(s_0) \xrightarrow{s}_1$ and $(s_0) \xrightarrow{s}$, whence from (i) of Definition 5.5 inductively applied $(s_0)_s \approx_{O(S_1, S)} (s_0)_s$. Since K^2 is prefix closed, $s' \in K^2 = l_1(s_0)^2$, we have $s_0 \xrightarrow{s'}_1 (s_0)_{s'}$ and according to Lemma 5.1 we have $\langle (s_0)_s, (s_0)_{s'} \rangle \in Aux(S_1)$. Now we have $(s_0)_s \xrightarrow{s}$ and $(s_0)_{s'} \xrightarrow{s}_1$, where recall $\langle (s_0)_s, (s_0)_{s'} \rangle \in Aux(S_1)$. By (ii) of the definition of observability relation we obtain that $(s_0)_s \xrightarrow{a}_1$, i.e., $(s_0) \xrightarrow{sa}_1$, which means that $sa \in l_1(s_0)^2 = K^2$. ■

Recall that the tests for observability proposed in Cho and Marcus (1989b) or Cassandras and Lafortune (1999) are to be made for all states of observer automaton $Obs(S_1)$ that are in fact different weak derivatives $(s_0)_{\hat{d}}$, $d \in P(l_1(s_0)^2)$. It would mean that the test for observability requires the construction of the observer. We have just shown that a test for observability might not rely on the observer, but on $Aux(S_1)$ instead. The test for condition (ii) of observability relation can be made for different $\lfloor s \rfloor_{Aux(S_1)}$. Recall that $\lfloor s \rfloor_{Aux(S)} = \cup_{\{d: s \in (s_0)_{\hat{d}}\}} (s_0)_{\hat{d}}$. However, it is known (Tsitsiklis, 1989) that polynomial tests for observability exist.

6. Normal relations

In this section we show an application of the above introduced notion of weak transition to the characterization of normality of languages introduced in supervisory control of DES with partial observations. For the sake of completeness, the concept of normality (Lin and Wonham, 1988; Cieslak et al., 1988; Cho and Marcus, 1989a; etc.) is stated.

DEFINITION 6.1 (Normality). Let $K, L \in \mathcal{L} : K \subseteq L$. K is said to be (L, P) —normal if $K^2 = L^2 \cap P^{-1}(P(K^2))$.

PROPERTY 6.1 K is (L, P) —normal iff $s \in K^2$, $s' \in L^2$, and $P(s) = P(s') \Rightarrow s' \in K^2$.

Proof: Since $K \subseteq L$, normality is equivalent to $L^2 \cap P^{-1}(P(K^2)) \subseteq K^2$, which is equivalent to the statement. ■

From the definitions of strong and weak transitions it follows:

COROLLARY 6.2 $K \subseteq L$ is (L, P) —normal iff $\forall w \in A^* : (L \xrightarrow{w} \text{ and } K \xrightarrow{P(w)} \Rightarrow K \xrightarrow{w})$.

Normality is preserved by the unobservable reach sets.

PROPOSITION 6.3 If a language K is (L, P) —normal then K_{uo} is (L_{uo}, P) —normal.

Proof: Using Corollary 6.2 it is sufficient to show that $\forall w \in A^* : L_{uo} \xrightarrow{w}$ and $K_{uo} \xrightarrow{P(w)}$ implies that $K_{uo} \xrightarrow{w}$. Recall that $L_{uo} = \cup \{L_\tau \mid \tau \in L^2 \text{ and } P(\tau) = \varepsilon\}$. Assume that $L_{uo} \xrightarrow{w}$ and $K_{uo} \xrightarrow{P(w)}$, i.e., there exist $\tau, \tau' \in A_{uo}^*$ such that $L_\tau \xrightarrow{w}$ and $K_{\tau'} \xrightarrow{P(w)}$. Hence, $L \xrightarrow{\tau w}$ and $K \xrightarrow{P(\tau' w)}$, thus $K \xrightarrow{P(\tau w)}$, because $P(\tau' w) = P(\tau w)$. It follows that $K \xrightarrow{\tau w}$ after the application of (L, P) —normality of K . But it means that $K_\tau \xrightarrow{w}$. Since $K_\tau \subseteq K_{uo}$ we obtain finally that $K_{uo} \xrightarrow{w}$, which was to be shown. ■

The following fact will be useful.

PROPERTY 6.4 Normality is preserved by (strong) transitions, i.e., if K is (L, P) —normal and $a \in A$ such that $K \xrightarrow{a}$ and $L \xrightarrow{a}$ then K_a is (L_a, P) —normal.

Proof: It is easily seen from Corollary 6.2. Indeed, if $L_a \xrightarrow{w}$ then $L \xrightarrow{aw}$ and if $K_a \xrightarrow{P(w)}$ then clearly $K \xrightarrow{P(aw)}$, whence by (L, P) —normality of K we deduce that $K \xrightarrow{aw}$, i.e., $K_a \xrightarrow{w}$. ■

Remark 6.5: It is interesting to notice that (L, P) —normality is preserved by deterministic weak transitions. It follows from Property 6.4, Proposition 6.3 and Proposition 4.3.

Now we introduce a binary relation that corresponds to the normality.

DEFINITION 6.2 (*Normal relation.*) Given two (partial) automata $S_1 = (S_1, \langle o_1, t_1 \rangle)$ and $S = (S, \langle o, t \rangle)$ as in Section 5 with initial state $s_0 \in S$, a binary relation $N(S_1, S)$ on $S_1 \times S$ is called a normal relation if for any $\langle s, t \rangle \in N(S_1, S)$ the following items hold:

- (i) $\forall a \in A : s \xrightarrow{a}_1 s_a \Rightarrow t \xrightarrow{a} t_a$ and $\langle s_a, t_a \rangle \in N(S_1, S)$
- (ii) $\forall a \in A_o : t \xrightarrow{a} t_a$ and $(\exists s' : \langle s, s' \rangle \in Aux(S_1) : s' \xrightarrow{a} s'_a) \Rightarrow s \xrightarrow{a}_1 s_a$.
- (iii) $\forall u \in A_{uo} : t \xrightarrow{u} t_u \Rightarrow s \xrightarrow{u}_1 s_u$.

Remark 6.6: Recall that (ii) can be expressed using the set $[s]_{Aux(S_1)}$: the condition $\exists s' : \langle s, s' \rangle \in Aux(S_1) : s' \xrightarrow{a}_1 s'_a$ can be replaced by the simpler one $[s]_{Aux(S_1)} \xrightarrow{a}_1$.

For $s \in S_1$ and $t \in S$ we write $s \approx_{N(S_1, S)} t$ whenever there exists a normal relation $N(S_1, S)$ on $S_1 \times S$ such that $\langle s, t \rangle \in N(S_1, S)$. Now we can prove:

THEOREM 6.7 A (partial) language K is (L, P) —normal iff $s_0 \approx_{N(S_1, S)} s_0$.

Proof: (\Rightarrow) Let K be (L, P) —normal. Denote

$$R = \{ \langle (s_0)_u, (s_0)_u \rangle \mid u \in K^2 \} \subseteq S_1 \times S.$$

Let us show that R is indeed a normality relation. Assume that $\langle q, r \rangle \in R$. From the form of R it follows that we can assume that $q = r = (s_0)_s$ for some $s \in K^2$. The same remark as in the proof of Theorem 5.6 applies. Namely, $s \in K^2$ such that $q = r = (s_0)_s$ might not be uniquely determined. Again we can show that the choice is not important. Nevertheless, since the argument is the same as in the proof of Theorem 5.6, we assume that s has been chosen in the way Lemma 5.1 in (ii) below can be correctly applied.

- (i) This part is the same as above in the proof of Theorem 5.6.
- (ii) Let $a \in A_o$ be such that $r = q \xrightarrow{a}$ and $\exists q' : \langle q, q' \rangle \in Aux(S_1)$ with $q' \xrightarrow{a}$. Then by Lemma 5.1 there exists a string $s' \in K^2 = l_1(s_0)^2$ such that $P(s) = P(s')$ and $q' = (s_0)_{s'}$. Thus we have $(s_0)_{s'} \xrightarrow{a}_1$, whence $s'a \in K^2$ and by normality we deduce $sa \in K^2$, because also $sa \in L^2 = l(s_0)^2$ (from $q = (s_0)_s \xrightarrow{a}$).
- (iii) Let $u \in A_{uo}$ and $q = (s_0)_s \xrightarrow{u}$. Thus we have and $su \in l(s_0)^2 = L^2$, where $P(su) = P(s)$, and recall that $s \in K^2$, whence by normality (Property 6.1) $su \in K^2$, i.e., $q = (s_0)_s \xrightarrow{u}_1$.

(\Leftarrow) Now let R be a normal relation on $S_1 \times S$ and $\langle s_0, s_0 \rangle \in R$. It will be shown that K is (L, P) —normal using Corollary 6.2. Let us prove by induction on the structural complexity of strings that for each $w \in A^* : L \xrightarrow{w}$ and $K \xrightarrow{P(w)}$ implies $K \xrightarrow{w}$. For $w = \varepsilon$ it is trivially true, because K^2 is prefix closed (i.e., $K \xrightarrow{\varepsilon}$). Suppose now that for $w \in A^*$ the above implication holds true. Let $L \xrightarrow{wq}$ and $K \xrightarrow{P(wq)}$. This implies in particular that $L \xrightarrow{w}$ and $K \xrightarrow{P(w)}$, hence by the induction hypothesis $K \xrightarrow{w}$. Since $\langle s_0, s_0 \rangle \in R$, by inductive application of (i) of the definition of normal relation we obtain that $\langle (s_0)_w, (s_0)_w \rangle \in R$.

Now suppose first $a \in A_o$ and $L_w \xrightarrow{a}$ and $K \xrightarrow{P(wa)}$. The latter means by definition of nondeterministic weak transition that there exists $v \in A^*$: $P(v) = P(w)$ and $K \xrightarrow{v} K' \xrightarrow{P(a)}$, where $K' = K_v$. Moreover, v (and K') can be chosen such that $K' \xrightarrow{a}$. Indeed, $K' \xrightarrow{P(a)}$ means by definition there exists $\tau, \tau' \in A_{uo}^*$ such that $K' \xrightarrow{\tau a \tau'}$. Thus, it is sufficient to consider $K = K'_\tau$ and $v\tau$ rather than v , because now $K = K'_\tau \xrightarrow{a}$. But $P(v\tau) = P(v) = P(w)$. Now $L_w \xrightarrow{a}$ gives $(s_0)_w \xrightarrow{a}$ and $v \in K^2$ with $K_v \xrightarrow{a}$ implies $va \in K^2$, i.e., $(s_0)_v \xrightarrow{a}_1$. By Lemma 5.1 $\langle (s_0)_w, (s_0)_v \rangle \in Aux(S_1)$, i.e., by application of (ii) of normal relation we obtain $(s_0)_w \xrightarrow{a}_1$, i.e., $K \xrightarrow{wg}$. In the case $a \in A_o$ the property (iii) gives the same result. Indeed, we simply obtain that $L_w \xrightarrow{a}$, i.e., $(s_0)_w \xrightarrow{a}$ implies by (iii) of the definition of normal relations that $(s_0)_w \xrightarrow{a}_1$. But this means that $K_w \xrightarrow{a}$. ■

Let us recall here the concept of control relation introduced in Rutten (1999). Let A_{uc} be the subset of uncontrollable events. We use the following stronger version of control relations with condition (i) strengthened to inclusion.

DEFINITION 6.3 (Control relation.) Given two partial automata $S_1 = (S_1, \langle o_1, t_1 \rangle)$ and $S = (S, \langle o, t \rangle)$ as above, a binary relation C on $S_1 \times S$ is called a control relation if for any $\langle s, t \rangle \in C$ the following items hold:

- (i) $\forall a \in A : s \xrightarrow{a}_1 s_a \Rightarrow t \xrightarrow{a} t_a$ and $\langle s_a, t_a \rangle \in C$
- (ii) $\forall u \in A_{uc} : t \xrightarrow{u} t_u \Rightarrow s \xrightarrow{u}_1 s_u$ and $\langle s_u, t_u \rangle \in C$

It has been shown in Rutten (1999) that

THEOREM 6.8 *A (partial) language K is controllable with respect to L and A_{uc} iff there exists a control relation $R \subseteq S_1 \times S$ such that $\langle s_0, s_0 \rangle \in R$.*

In the above definition, S_1 corresponds to the closed-loop system consisting of the plant and the supervisor and S corresponds to the open-loop plant. From Theorem 5.6 and Theorem 6.7 after comparing the definitions of observability and normal relations it follows immediately the well known fact that normality implies observability. More precisely:

COROLLARY 6.9 *K is (L, P) —normal iff K is observable with respect to L and P and controllable with respect to L and A_{uo} . In particular, we obtain the following well known implication. If K is observable with respect to L and A_{uc} , controllable with respect to L and A_{uc} , and $A_c \subseteq A_o$ (i.e., $A_{uo} \subseteq A_{uc}$), then K is (L, P) —normal.*

Finally let us compare our result with that in Cho and Marcus (1989b). The test for normality in Cho and Marcus (1989b) is made for all classes introduced in that paper that are in fact different weak derivatives $(s_0)_{\hat{a}}$, $d \in P(K)$. It means that their test for observability and/or normality requires the construction of an observer. It is known (Tsitsiklis, 1989) that these tests can be done in polynomial time. We have shown that

these tests do not rely on the observer, but on $Aux(S_1)$ instead in exactly the same way as the test for observability.

7. Supremal normal and controllable sublanguages

It is well known that the supremal observable sublanguage of a given language does not always exist. On the other hand, supremal normal and therefore also supremal controllable and normal sublanguages of a given language do exist. Algorithms for their computation have been presented in Cho and Marcus (1989a,b) and Cieslak et al. (1988). The algorithm in Cho and Marcus (1989b) has been developed using the invariance properties of equivalence relations induced on a given language by a natural projection. The concept of active event set of a given state or a subset of states (corresponding to an equivalence class) is used. However, this concept can be captured in a more natural way using the coalgebraic structure of partial automata.

7.1. Construction of supremal normal and controllable sublanguages using normal relations

Given two (ordinary and not necessarily prefix closed) languages K and L such that $K \subseteq L$, let us consider partial automata $S' = (S', \langle o', t' \rangle)$ and $S = (S, \langle o, t \rangle)$ representing K and L in the sense made precise below and such that S' is a subautomaton of S with s_0 their common initial state. Let $l' : S' \rightarrow \mathcal{L}$ and $l : S \rightarrow \mathcal{L}$ be the associated behavior homomorphisms, where $K = (l'(s_0))^1$ and $L = (l(s_0))^1$. Recall that such a representation with subautomaton always exists (Cho and Marcus, 1989a). Let the transition function of S be denoted by \rightarrow , i.e., $s \xrightarrow{a} s_a$ means $sa = t(s)(a)$ and similarly the transition function t' of S' is denoted by \rightarrow' , i.e., $s \xrightarrow{a'} s_a$ means $sa = t'(s)(a)$. This notation is possible, because S' is a subautomaton of S . Moreover we can assume without loss of generality that S' is a trim automaton.

We consider normal relations on $S' \times S$. Theorem 6.7 suggests a test for normality. We start by including $\langle s_0, s_0 \rangle \in N(S', S)$ and we continue by adding new states using (i) of the definition of normal relation. Every time a new state is included we test conditions (ii) and (iii), either these conditions are satisfied and we continue the construction of $N(S', S)$ or one of them is not satisfied, in which case the procedure aborts and the conclusion is that K is not (L, P) —normal. It is obvious that if this procedure is never aborted, it leads to the diagonal relation on S' , denoted by $diag(S')$, because S' is a trim subautomaton of S . In this case $diag(S')$ is a normal relation on $S' \times S$ proving that K is (L, P) —normal. Thus diagonal normal relations are of special interest for testing the normality of a language and computing supremal normal sublanguages. Conditions (ii) and (iii) of normal relations can be reformulated:

- (ii) $\forall s \in S' \subseteq S$ and $\forall a \in A_o : \left(s \xrightarrow{a} s_a \text{ and } [s]_{Aux(S')} \xrightarrow{a'} \right) \Rightarrow s \xrightarrow{a'} s_a.$
- (iii) $\forall s \in S' \subseteq S$ and $\forall u \in A_{uo} : s \xrightarrow{u} s_u \Rightarrow s \xrightarrow{u'} s_u.$

The procedure for computation of the supremal normal sublanguage can now be easily devised. It will consist in removing some strings that cause the violation of conditions (ii) and (iii) above. This amounts to removing some states and edges from automaton S' , which is made by Algorithms 1 and 2.

Note that condition (iii) of the definition of normal relation is just the “controllability” condition with respect to A_{uo} instead of A_{uc} that appears in the definition of control relation (Rutten, 1999). Therefore, the first step (algorithm) of our computation will be similar to the case of complete observations, i.e., the removal of certain states that violate our “extended controllability” condition (iii).

Now let us devise an algorithm that ensures condition (iii) of normality, which is of the same kind as (ii) of control relations. Therefore for computation of supremal normal and controllable sublanguages we can take care of (ii) of controllability and (iii) of normal relations at the same time and use a natural extension of the algorithm presented in Rutten (1999).

Let $R = \{\langle (s_0)_w, (s_0)_w \rangle \mid w \in K \subseteq L\} \subseteq S' \times S$. Clearly R is a diagonal relation on S' , i.e., $R = \text{diag}(S')$ and it carries an automaton structure as given in Rutten (1999). Recall that for $\langle t, t \rangle \in R$ we put $\langle t, t \rangle \xrightarrow{a}_R \langle t, t \rangle_a$ iff $t \xrightarrow{a} t_a$ in S' , which implies $t \xrightarrow{a} t_a$ (because S' is a subautomaton of S), in which case $\langle t, t \rangle_a = \langle t_a, t_a \rangle$. The output function plays no role here, it can be arbitrary. Now we can repeat the algorithm from Rutten (1999) for R .

ALGORITHM 1 Define the following operator that maps any diagonal relation $H \subseteq S' \times S$ to

$$\Gamma(H) = \{\langle s, s \rangle \in H \mid \forall u \in A_{uc} \cup A_{uo} : s \xrightarrow{u} s' \Rightarrow \langle s, s' \rangle \in H\}.$$

We put $\hat{R} = \bigcap_{i \geq 0} \Gamma^i(R)$.

Then $\hat{R} = \text{diag}(S_1)$ for some $S_1 \subseteq S'$ and \hat{R} is the greatest fixed point of Γ that is contained in R :

LEMMA 7.1 Let Γ and \hat{R} be as defined above. Then 1. $\hat{R} = \Gamma(\hat{R})$ and 2. For any $R' \subseteq R$: If $R' \subseteq \Gamma(R')$ then $R' \subseteq \hat{R}$

Proof: The operator Γ is monotone (as a set operator with respect to inclusion), i.e., there exists a fixpoint according to Tarski (1955). A detailed proof can be found in Rutten (1999). ■

The construction in Algorithm 1 yields the supremal sublanguage of K that is controllable with respect to L and $A_{uc} \cup A_{uo}$. More precisely, we have the following theorem:

THEOREM 7.2 Let $K = l'(s_0)^1$, $L = l(s_0)^1$, and automata R and \hat{R} be as above. Then $\langle s_0, s_0 \rangle \in \hat{R}$ and $\hat{l}(\langle s_0, s_0 \rangle)^1 = E$, where $\hat{l} : \hat{R} \rightarrow \mathcal{L}$ is the unique homomorphism describing the behavior of states in \hat{R} and E is the supremal sublanguage of K that is controllable with respect to L and $A_{uc} \cup A_{uo}$.

Proof: The same as the proof of Theorem 9.2 (Rutten, 1999) with the only difference that A_{uc} is replaced now by $A_{uc} \cup A_{uo}$. ■

The effect of the procedure described in Algorithm 1 is to remove the states of automaton S' that violate condition (iii). It is clear that Algorithm 1 stops after a finite number of iterations for regular languages K and L represented by finite automata S' and S , respectively. The representation \hat{R} of E (supremal sublanguage of K that is controllable with respect to L and $A_{uc} \cup A_{uo}$) given by Algorithm 1 is often not suitable for the forthcoming algorithm. Notice that in Algorithm 1 it was not necessary that S' is state-partition automaton. However, Algorithm 1 must be followed by another algorithm in order to ensure that condition (ii) of normal relation holds and this property of representation will be required.

Let us suppose that S_1 is a representation of E such that S_1 is a subautomaton of S and S_1 is a state-partition automaton. This will be needed for the correctness of the algorithm below. Denote by $l_1 : S_1 \rightarrow \mathcal{L}$ the behavior homomorphism of S_1 , i.e., $l_1(s_0)^1 = E$.

Remark 7.3: Note that S can now be a different representation than the one resulting from Algorithm 1, because of the requirements that S_1 is a subautomaton of S and that S_1 is a state-partition automaton. Such representations S_1 as subautomaton of S can be constructed using the procedure from Cho and Marcus (1989b) that ensures the condition of S_1 being a state-partition automaton. It is not difficult to see that in automaton $S_1 = (S_1, \langle o_1, t_1 \rangle)$ we have

$$(C) \forall s \in S_1 : \forall u \in A_{uc} \cup A_{uo} : (s \xrightarrow{u} \Rightarrow s \xrightarrow{u}_1).$$

This means that the controllability condition does not depend on the particular representation. This is very important, because in Algorithm 1 smaller representations of K and L can be used, while the condition (iii) of normal relations remains valid for representations S_1 and S that we use in Algorithm 2.

Thus we consider separately (ii) of normal relations in Algorithm 2 below. We denote by $Acc(S)$ the accessible part of an automaton S and make the following construction.

ALGORITHM 2 Construct the automaton $(\tilde{S}, \langle \tilde{o}, \tilde{t} \rangle)$ in the following way.

- 1) We put $\tilde{S} = S_1$.
- 2) $\tilde{t} : \tilde{S} \rightarrow (1 + \tilde{S})^A$ with

for $a \in A_{uo} : q \in \tilde{S} : \tilde{t}(q)(a)$ is defined iff $t_1(q)(a)$ is defined, in which case $\tilde{t}(q)(a)$,
and:

$$\text{for } a \in A_o : \tilde{t}(q)(a) \text{ is defined iff } \forall s \in [q]_{Acc(S_1)} : (s \xrightarrow{a} \Rightarrow s \xrightarrow{a}_1),$$

in which case $\tilde{t}(q)(a) = t_1(q)(a)$.

- 3) *The output function is unchanged: $\tilde{o} = o_1$.*
- 4) *Put $(\tilde{S}, \langle \tilde{o}, \tilde{t} \rangle) := \text{Acc}(\tilde{S}, \langle \tilde{o}, \tilde{t} \rangle)$.*

Algorithm 2 just states that for any state $q \in \tilde{S}$ an outgoing edge labeled by $a \in A_o$ is to be removed (of course only if $q \xrightarrow{a}_1$), whenever there exists $s \in [q]_{\text{Aux}(S_1)} : s \xrightarrow{a}$ and $s \xrightarrow{s}_1$ in S_1 . Intuitively it means that we remove an observable a -transition from all $s \in [q]_{\text{Aux}(S_1)}$ at the same time, which makes the resulting language normal. However, there might be a conflict if there exists $q' \in \tilde{S}$ such that $\langle q, q' \rangle \in \text{Aux}(S_1)$ (i.e., $q \in [q']_{\text{Aux}(S_1)}$) and $[q]_{\text{Aux}(S_1)} \neq [q']_{\text{Aux}(S_1)}$. Then it might happen that for some $s \in [q]_{\text{Aux}(S_1)} \cap [q']_{\text{Aux}(S_1)}$ we should remove $a \in A_o$ from $t_1(q)$ (regarding from the class $[q]_{\text{Aux}(S_1)}$) and on the other hand we should keep it regarding from $[q]_{\text{Aux}(S_1)}$. From this characterization the importance of the assumption that $\text{Aux}(S_1)$ is an equivalence relation is easily seen. Indeed, if $\text{Aux}(S_1)$ is an equivalence relation then for any $q \in \tilde{S}$ such that $\langle q, q_1 \rangle \in \text{Aux}(S_1)$ we obtain according to Observation 5.4 that $[q]_{\text{Aux}(S_1)} = [q_1]_{\text{Aux}(S_1)}$, i.e., the above conflict situation cannot happen. Nevertheless, a stronger condition for S_1 being a state-partition automaton is required to guarantee the supremality of the normal sublanguage represented by the resulting automaton $\text{Acc}(\tilde{S}, \langle \tilde{o}, \tilde{t} \rangle)$. Note also that the procedure described in Algorithm 2 can lead to removal of certain states that become inaccessible from the initial state after removing some observable transitions.

Algorithm 1 followed by Algorithm 2 ensure the conditions of normal relations are fulfilled. Unlike Algorithm 1, which consists in iterative application of the operator Γ , Algorithm 2 is a ‘single-step’ one. However there is an intrinsic difficulty concerning the computation of supremal controllable and normal sublanguages. It is possible that the diagonal relation of the resulting automaton \tilde{S} might not be a control relation, i.e., condition (ii) of control relation might be violated due to the removal of some uncontrollable edges in Algorithm 2. If the supremal normal sublanguage is of interest (i.e., the condition in the definition of Γ is required to be valid only for $u \in A_{uo}$ instead of $u \in A_{uo} \cup A_{uc}$), Algorithm 2 does not affect what has been done in Algorithm 1 due to the fact that in our definition of normal relations we have separated the conditions for observable events (ii) and unobservable events (iii). Therefore in Algorithm 2 we remove only observable transitions from states in \tilde{S} and condition (iii) will hold for \tilde{S} , i.e., its diagonal relation will be a normal relation on $\tilde{S} \times S$. However, if the supremal normal and controllable sublanguage is of interest, then Algorithm 2 may affect the condition (iii) for some $u \in A_o \cap A_{uc}$. Only in the case $A_o \subseteq A_c$, the supremal normal and controllable sublanguage is obtained after Algorithm 1 and Algorithm 2 have been applied. This explains why in general an iterative scheme that consists in consecutive repeating of these algorithms in the same way as in Cho and Marcus (1989a) or Cassandras and Lafortune (1999) is used. In the next section we show that it is not necessary to consider such an iterative scheme if we implement Algorithm 1 as a ‘single-step’ algorithm instead of an iteration. In the proof of the following theorem we use the notation \rightarrow for transition function \tilde{t} of \tilde{S} . Recall that transition functions t and t_1 of S and S_1 are denoted through \rightarrow and \rightarrow_1 , respectively.

Now we are interested in the computation of supremal (L, P) —normal sublanguage of K . We need the following modification of Algorithm 1, where the condition in the definition of operator Γ is required only for all $u \in A_{uo}$ instead of for all $u \in A_{uc} \cup A_{uo}$.

ALGORITHM 1' Define the following operator that maps any diagonal relation $H \subseteq S' \times S$ to

$$\Gamma(H) = \{ \langle s, s \rangle \in H \mid \forall u \in A_{uo} : s \xrightarrow{u} \Rightarrow s \xrightarrow{u}, \text{ and } \langle s_u, s_u \rangle \in H \}.$$

We put $\hat{R} = \bigcap_{i \geq 0} \Gamma^i(R)$

THEOREM 7.4 Algorithm 1' followed by Algorithm 2 yields the supremal (L, P) —normal sublanguage of K in the following sense: $\tilde{l}(s_0)^1$, where $\tilde{l} : \tilde{S} \rightarrow \mathcal{L}$ is the unique behavior homomorphism, is the supremal (L, P) —normal sublanguage of K .

Proof: The coinductive proof principle is used. Note that step 4) of Algorithm 2 is not used, because in fact the behavior homomorphism \tilde{l} takes automatically care of the accessibility operation. First we show that $\tilde{l}(s_0)^1$ is a (L, P) —normal sublanguage of K . To prove the normality of $\tilde{l}(s_0)^1$ we show that the following relation is a normality relation on $\tilde{S} \times S$. Then $\tilde{l}(s_0)$ is (L, P) —normal sublanguage of K according to Theorem 6.7. Since normality is a property of the prefix closure, this means that $\tilde{l}(s_0)^1$ is (L, P) —normal sublanguage of K in the classical framework.

$$R = \{ \langle \langle (s_0)_u, (s_0)_u \rangle \mid u \in \tilde{l}(s_0)^2 \}.$$

Take a pair $\langle (s_0)_v, (s_0)_v \rangle \in R$ for some $v \in \tilde{l}(s_0)^2$.

- (i) If $(s_0)_v \xrightarrow{a}$, for $a \in A$, then clearly by construction of Algorithm 2 $(s_0)_v \xrightarrow{a}$. It is clear from the definition of R that $\langle (s_0)_{va}, (s_0)_{va} \rangle \in R$.
- (ii) Let $a \in A_o$ be such that $(s_0)_v \xrightarrow{a}$ and let there exist $s' \in \tilde{S} : s' \approx_{Aux(\tilde{S})} (s_0)_v$ with $s' \xrightarrow{a}$. By Lemma 5.1 there exist two strings $w, w' \in A^*$ such that $P(w) = P(w'), (s_0)_v = (s_0)_w$, and $s' = (s_0)_{w'} \xrightarrow{a}$. According to the construction of Algorithm 2 for any $s \approx_{Aux(S_1)} (s_0)_{w'}$ there must be $s \xrightarrow{a} \Rightarrow s \xrightarrow{a}_1$. In order to show that $(s_0)_v \xrightarrow{a}$, it must be that for any $q \approx_{Aux(S_1)} (s_0)_v$ there must be $q \xrightarrow{a} \Rightarrow q \xrightarrow{a}_1$. But using the fact that $Aux(S_1)$ is transitive (follows from 5.3) and the fact that $s' \approx_{Aux(\tilde{S})} (s_0)_v$ implies that $s' \approx_{Aux(S_1)} (s_0)_v$ we obtain that $\langle s', q \rangle \in Aux(S_1)$. But this just means that for any $q \approx_{Aux(S_1)} (s_0)_v$ we have $q \xrightarrow{a} \Rightarrow q \xrightarrow{a}_1$, i.e., $(s_0)_v \xrightarrow{a}$.
- (iii) Let $a \in A_o$ be such that $(s_0)_v \xrightarrow{a}$. Then according to Algorithm 1' we have $(s_0)_v \xrightarrow{a}$, because $a \in A_{uo}$ and $\tilde{S} \subseteq S_1$. This shows together with (i) and (ii) that R is a normality relation on $\tilde{S} \times S$.

We show finally that $\tilde{l}(s_0)^1$ is the supremal (L, P) —normal sublanguage of K . Let N be a (L, P) —normal sublanguage of K . We construct an auxiliary partial language (N, \bar{N}) that is for the sake of simplicity also denoted by N . Similarly partial languages corresponding to K and L , i.e., (K, \bar{K}) and (L, \bar{L}) , respectively, are denoted by K and L . This abuse of notation should not lead to any confusion. Then it is sufficient to show that

$$R = \{ \langle N_u, \tilde{l}(s_0)_u \rangle \mid u \in N^2 \}$$

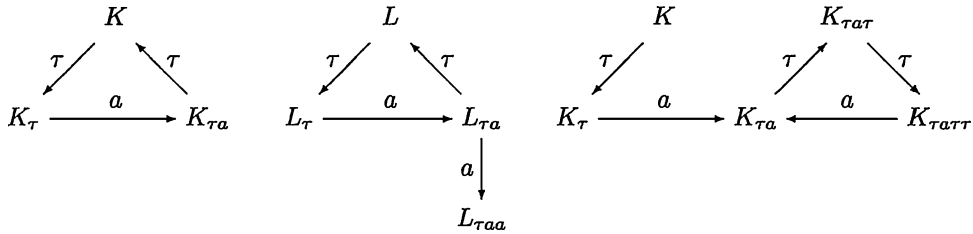
is a simulation relation. Then we will have $(N, \bar{N}) \subseteq \tilde{I}(s_0)$, i.e., in particular $N \subseteq \tilde{I}(s_0)^1$. Take an arbitrary pair $\langle N_w, \tilde{I}(s_0)_w \rangle \in R$ for some $w \in N^2$.

- (i) Let $N_w \downarrow$, i.e., $K_w \downarrow$, and therefore $\delta((s_0)_w) = o_1((s_0)_w) = 1$ according to point 3) of Algorithm 2. But this is equivalent to $\tilde{I}(s_0)_w \downarrow$ as a partial language.
- (ii) Let $N_w \xrightarrow{a}$ for $a \in A_o$. Then also $K_w \xrightarrow{a}$, because $N \subseteq K$ (the inclusion holds for both ordinary and induced partial languages). Thus, $L_w \xrightarrow{a}$ as well. This means that $(s_0)_w \xrightarrow{a}_1$ and $(s_0)_w \xrightarrow{a}$. In order to show that $\tilde{I}(s_0)_w \xrightarrow{a}$ for $a \in A_o$, i.e., $(s_0)_w \xrightarrow{a}$, we must prove that for any $q \approx_{Aux(S_1)} (s_0)_w : q \xrightarrow{a} \Rightarrow q \xrightarrow{a}_1$. There exist $v, v' : P(v) = P(v')$ such that $q = (s_0)_{v'}$ and $(s_0)_w = (s_0)_v$. Since S_1 is a state-partition automaton and $(s_0)_w$ is in two possibly different states of the observer automaton, we conclude by the property of state-partition automaton that these two states of the observer automaton coincide. But this means that there exists $w' \in A^*$ such that $P(w) = P(w')$ and $q = (s_0)_{w'}$. Now $q \xrightarrow{a}$ means that $w'a \in L^2$. Using normality of N it follows from $wa \in N^2$ and $w'a \in L^2$ that $w'a \in N^2$. Therefore $w'a \in K^2$ (because $N \subseteq K$), which means that $q \xrightarrow{a}_1$. The case $a \in A_{uo}$ is much easier. Again, $N_w \xrightarrow{a}$ implies that $K_w \xrightarrow{a}$. We show that $\tilde{I}(s_0)_w \xrightarrow{a}$, i.e., $(s_0)_w \xrightarrow{a}$. It follows from Algorithm 1' that $(s_0)_w \xrightarrow{a}$ implies that $(s_0)_w \xrightarrow{a}_1$, whence $(s_0)_w \xrightarrow{a}$, because Algorithm 2 does not affect transitions labeled by $a \in A_{uo}$.

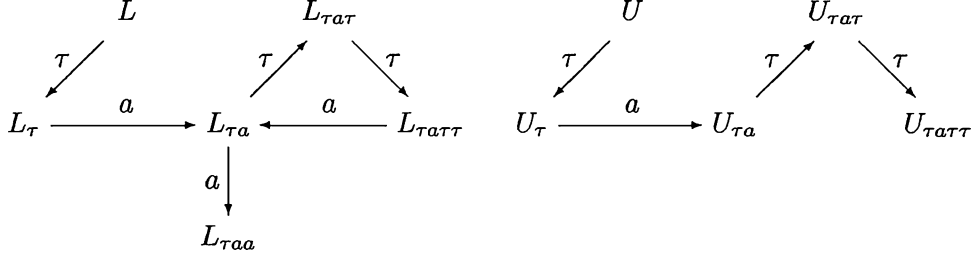
Note that since N was an arbitrary (L, P) —normal sublanguage of K and $\tilde{I}(s_0)^1$ was shown to be a normal sublanguage of K , $\tilde{I}(s_0)^1$ must be the supremal (L, P) —normal sublanguage of K . ■

Now we illustrate the computation of the supremal (L, P) —normal sublanguage of K by the following simple example.

Example 3: Let $A = \{a, \tau\}$ with $A_o = \{a\}$, K , and L are given by automata representations below. We assume that all states are marked, which does not play any role for normality.



The original representation of K is not a state-partition automaton. The corresponding state-partition automaton representing the same language K , i.e., the synchronous product $S_1 := \langle K \rangle_{A_o} \parallel_{Obs} \langle K \rangle$ is drawn on the right of K and L . The corresponding representation S of L such that S_1 is a subautomaton of S is given below together with the output of Algorithm 2, denoted by U .



Notice that K is controllable with respect to L and A_{uo} , i.e., the action of Algorithm 1' is empty in this case. The Algorithm 2 then removes the second transition labelled by a , yielding thus U , which is easily seen to be the supremal (L, P) —normal sublanguage of K .

To conclude, we present a coalgebraic interpretation of the algorithm given in Cho and Marcus (1989b). The algorithm given therein when interpreted in our framework yields a partial automaton $\tilde{S} = (\tilde{S}, \langle \tilde{o}, \tilde{i} \rangle)$, where $\tilde{S} = \{\cup_{d \in G} (s_0)_{\tilde{d}}\}$ with $G = \{d \in P(K) : \forall s \in (s_0)_{\tilde{d}} : \forall u \in A_{uo} : s \xrightarrow{u} \Rightarrow s \xrightarrow{u_1}\}$, which is the first part of the algorithm, and the definition of $\tilde{o} : \tilde{S} \rightarrow 2$ and $\tilde{i} : \tilde{S} \rightarrow (1 + \tilde{S})^A$ is the second part. The output function is unchanged, i.e., $o = o_1 \upharpoonright_{\tilde{S}}$. For any $q \in \tilde{S}$ there exists by definition of \tilde{S} a $d \in G : s \in (s_0)_{\tilde{d}}$. Using this,

for $q \in \tilde{S}$ and $a \in A_{uo}$ $\tilde{i}(q)(a)$ is defined iff $t_1(q)(a)$ is defined, in which case $\tilde{i}(q)(a) = t_1(q)(a)$, and

for $a \in A_o$ $\tilde{i}(q)(a)$ is defined iff $\forall s \in (s_0)_{\tilde{d}} : (s \xrightarrow{a} \Rightarrow s \xrightarrow{a_1})$, in which case $\tilde{i}(q)(a) = t_1(q)(a)$.

The construction of \tilde{i} is the second part of their algorithm, where the condition for \tilde{i} to be defined is similar to our condition 2 of Algorithm 2, but expressed using deterministic weak derivatives. An important feature is that S_1 is a state-partition automaton, thus $d \in P(K)^2$ such that $q \in (s_0)_{\tilde{d}}$ is for any state $q \in \tilde{S}$ uniquely determined. Remark that the last procedure which is taken from Cho and Marcus (1989b) is not correct if S_1 is not a state-partition automaton.

Our procedure for computation of the supremal (L, P) —normal sublanguage of K consists in Algorithm 1' followed by Algorithm 2. We show that it is different from that described in Cho and Marcus (1989b) in both steps. Implicitly, the two steps are also present in that paper, the first one is the construction of carrier set (removal of some states) and the second one consists in removing some observable transitions.

It has been shown that for regular languages K and L there exists always a finite automaton representation that satisfies the condition of state-partition automaton. Namely, the synchronized product $\langle K \rangle P := \langle K \rangle \parallel \text{Obs}(\langle K \rangle)$ is a state-partition automaton as follows from results in Cho and Marcus (1989a). In particular, $\text{Aux}(\langle K \rangle_P)$ is an equivalence relation according to Proposition 5.3.

Remark that this procedure is correct only if there is no conflict between different states of $\text{Obs}(S_1)$. This means that S_1 must be a state-partition automaton.

Finally, let us compare our algorithms with those from Cho and Marcus (1989b). Our algorithm for computation of supremal normal sublanguages (Algorithm 1' followed by Algorithm 2) differs from that presented in Cho and Marcus (1989b) in both steps. The first step (construction of \tilde{S}) differs from our Algorithm 1' in that we do not necessarily remove in Algorithm 1' all states in $(s_0)_d$ for $d \notin G$ as can be shown in a simple example. However these states are automatically removed by Algorithm 2, while producing the accessible part of \tilde{S} . This means that there is some saving on computational complexity using our algorithm comparing to that presented in Cho and Marcus (1989b), because Algorithm 1 can use minimal representations, the computation of the set G and the automaton \tilde{S} from Cho and Marcus (1989b) is not necessary. Nevertheless, both algorithms suffer from the exponential complexity in the worst case, because they rely on the subset constructions related to partial observations.

The concepts developed in this paper lead to new algorithms for supervisory control with partial observations presented without details in Komenda (2002a). In that paper the concept of normality of a language with respect to a plant is also captured by different relations. These are introduced on finite automata representations in order to make the computations feasible. Our approach is inspired by the work of Cho and Marcus (1989b), where algebraic characterizations using the concept of invariant relations have been presented. The main advantage of the coalgebraic approach is that the formulations using relations provide a canonical way how to check different properties of languages (like controllability, observability, and normality). Since all these relations are in fact different weaker forms of bisimulation, we can proceed in the same way as for checking the bisimilarity (Rutten, 1999). Coalgebraic methods yield new algorithms and more general results for the computation of the supremal normal and normal and controllable (see also next section) sublanguages based on the corresponding relations.

We remark finally that the algorithm we have presented is composed of two separate algorithms, which makes its use in some theoretical problems involving supremal normal sublanguages (e.g., conditions for its commutation with synchronous product) quite difficult. Therefore we will present in the next section 'single-step' algorithms for the computation of the supremal normal and normal and controllable sublanguages motivated by coinductive definitions.

7.2. *Note about maximal observable sublanguages*

A procedure for the computation of maximal observable sublanguages has been proposed by Cho and Marcus in Cho and Marcus (1989b). It turns out that there are many technical difficulties while computing such maximals. The main issue is to ensure the procedure to be nonretrospective, i.e., that the procedure does not affect what has been computed earlier in the algorithm.

It is to be expected that in our setting similar problems occur. An algorithm for maximal observable sublanguages can be designed using observability relations. However it is not easy to ensure the correctness of such a procedure because of the difficulties related to the fact that now some unobservable transitions are also to be removed. Moreover, it is not necessary to remove transitions simultaneously from all states that form a

state of the observer automaton and there is no unique way to do it, i.e., different orderings of the controllable event set must be considered. In the next section another approach to the synthesis of observable sublanguages is presented. It will be based on coinductive definitions.

8. Coinductive definition of supervised product and partial bisimulation under partial observations

This section and the two next ones make use of the application of a powerful technique called coinduction to discrete-event control. While coinductive proofs have already been used in the previous sections, coinductive definitions are used below to capture some important concepts like closed-loop language and optimal super/sublanguages. In this section we present the definition of a supervised product of languages that describes the behavior of a supervised DES under partial observations. Assume throughout this section that the specification K and the open-loop partial language L ($K \subseteq L$) are given.

In the last section we have been studying relations on automata representations and we have formulated the basic properties of observability and normality using these relations. Now we aim to use the coinductive definitions. For this reason we must work with the final automaton of partial languages, where the coinductive definitions can be used. Coinductive definitions are used for defining algebraic operations, e.g., binary operations, on elements of final coalgebras. They consist in defining the coalgebraic structure (given by the functor) on the result of operation. For partial languages this means that new operations can be introduced (or sometimes the known ones reintroduced) by defining the output and transition functions (i.e., input derivatives). Interestingly, differential equations from analysis may also be viewed as coinductive definitions of solutions they define if a suitable coalgebraic structure (stream automata for ODEs, weighted automata for PDEs) is used (Rutten, 2003).

Note that observability and normality relations can be defined in the final automaton \mathcal{L} . However, there is a difficulty with the fact that once we use the minimal representations $\langle K \rangle, \langle L \rangle \in \mathcal{L}$ as the subautomata of \mathcal{L} generated by K and L , respectively, it is not true in general that for $K \subseteq L$, $\langle K \rangle$ is a subautomaton of $\langle L \rangle$. Therefore some additional technicalities are involved. In particular, $Aux(S_1)$ is replaced by $Aux(K, L)$ to stress the fact that both $\langle K \rangle$ and $\langle L \rangle$ are involved. Its definition has been first presented in Komenda (2002b).

In order to characterize the observability property we first need to introduce the following auxiliary relation defined on $DK \times DL$. Note that any relation $R \subseteq (DK \times DL)^2$ can be endowed with the following transition structure: for $(M, N) \xrightarrow{P(a)} (M', N')$ iff $M \xrightarrow{a} M_a$ and $N \xrightarrow{a} N_a$ with $M' = M_a$ and $N' = N_a$. We write $(M, N) \Rightarrow (M', N')$ iff $\exists s \in M^2 \cap N^2 : P(s) = a, M' = M_s,$ and $N' = N_s$.

DEFINITION 8.1 *A binary relation $Aux(K, L) \subseteq (DK \times DL)^2$, called observational indistinguishability relation, is the smallest relation satisfying:*

- (i) $\langle (K, L), (K, L) \rangle \in Aux(K, L)$

- (ii) If $\langle (M, N), (Q, R) \rangle \in Aux(K, L)$ then $\forall a \in A : \text{if } (M, N) \xrightarrow{P(a)} (M', N') \text{ and } (Q, R) \xrightarrow{a} (Q', R') \Rightarrow \langle (M', N'), (Q', R') \rangle \in Aux(K, L)$

For $(M, N), (Q, R) \in DK \times DL$ we write $(M, N) \approx_{Aux}^{K, L} (Q, R)$ whenever $\langle (M, N), (Q, R) \rangle \in Aux(K, L)$.

LEMMA 8.1 For given partial languages $K, L : \langle (M, N), (Q, R) \rangle \in Aux(K, L)$ iff there exist two strings s, s' such that $P(s) = P(s')$ and $M = K_s, N = L_s, Q = K_{s'},$ and $R = L_{s'}$.

Proof: (\Leftarrow) Let $(M, N) \in DK \times DL$ and $(Q, R) \in DK \times DL$ and there exist two strings $s, s' \in K^2$ such that $P(s) = P(s'), M = K_s, N = L_s, Q = K_{s'},$ and $R = L_{s'}$. Let $s = s_1 \dots s_n$ and $s' = t_1 \dots t_m$. Let $P(s) = P(s') = a_1 \dots a_k$. Then $n \geq k, m \geq k$, and there exist two increasing sequences of integers (indices) $u_i \geq i, i = 1, \dots, k$ and $v_i \geq i, i = 1, \dots, k$ such that $a_i = s_{u_i} = t_{v_i}$. Since $s, s' \in K^2$, and all a_i are observable events we can write $(K, L) \xrightarrow{P(a_1) \dots P(a_n)} (M, N)$ and also $(K, L) \xrightarrow{P(a_1) \dots P(a_k)} (Q, R)$, whence by (ii) inductively applied $(M, N) \approx_{Aux}^{K, L} (Q, R)$.

(\Rightarrow) Let $(M, N) \approx_{Aux}^{K, L} (Q, R)$. By the construction of $Aux(K, L)$ there exist $a_1, \dots, a_k \in A$ such that $(K, L) \xrightarrow{P(a_1) \dots P(a_k)} (M, N)$ and $(K, L) \xrightarrow{P(a_1) \dots P(a_k)} (Q, R)$. Therefore there exist two strings s, s' with the same projection with $M = K_s, N = L_s, Q = K_{s'},$ and $R = L_{s'}$. ■

Now we repeat the definition of the observability relation used in Komenda (2002b).

DEFINITION 8.2 (Observability relation). Given two (partial) languages K and L , a binary relation $O(K, L) \subseteq DK \times DL$ is called an observability relation if for any $\langle M, N \rangle \in O(K, L)$ the following items hold:

- (i) $\forall a \in A : M \xrightarrow{a} N$ and $\langle M_a, N_a \rangle \in O(K, L)$.
- (ii) $\forall a \in A_c : N \xrightarrow{a}$ and $(\exists M' \in DK, N' \in DL : (M', N') \approx_{Aux}^{K, L} (M, N) \text{ and } M' \xrightarrow{a} N') \Rightarrow M \xrightarrow{a}$ and $\langle M_a, N_a \rangle \in O(K, L)$.

For $M \in DK$ and $N \in DL$ we write $M \approx_{O(K, L)} N$ whenever there exists an observability relation $O(K, L)$ on $DK \times DL$ such that $\langle M, N \rangle \in O(K, L)$. In order to check whether for a given pair of (partial) languages (K and L), K is observable with respect to L , it is sufficient to establish an observability relation $O(K, L)$ on $DK \times DL$ such that $\langle K, L \rangle \in O(K, L)$. Indeed, we have

THEOREM 8.2 A (partial) language K is observable with respect to L (with $K \subseteq L$) and P iff $K \approx_{O(K, L)}$.

Proof: (\Rightarrow) Let K be observable with respect to L . Denote

$$O_1(K, L) = \{ \langle K_u, L_u \rangle \in DK \times DL \mid u \in K^2 \}.$$

Let us show that $O_1(K, L)$ is an observability relation.

Let $\langle M, N \rangle \in O_1(K, L)$. We can assume that $M = K_s$ and $N = L_s$ for $s \in K^2$. We must show that conditions (i) and (ii) of the Definition 8.2 are satisfied.

- (i) Let $M \xrightarrow{a}$ for $a \in A$. Notice that $K \subseteq L$ implies that for any $u \in K^2$, $K_u \subseteq L_u$. In particular $N \xrightarrow{a}$, because $M = K_s \subseteq L_s = N$ and it follows from the definition of $O_1(K, L)$ that $\langle M_a, N_a \rangle \in O_1(K, L)$.
- (ii) Let $N \xrightarrow{a}$ for $a \in A_c$ and $\exists(M', N') \approx_{Aux}^{K, L}(M, N) : M' \xrightarrow{a}$. Then by Lemma 8.1 there exist two strings $s', s'' \in K^2$ such that $P(s') = P(s'')$ and $M' = K_{s'}$, $N' = L_{s'}$, $M = K_{s''}$ ($=K_s$), and $N = L_{s''}$ ($=L_s$). Now $M' \xrightarrow{a}$ implies that $s'a \in K^2$. From $N \xrightarrow{a}$ and $N = L_{s''}$ follows $s''a \in L^2$. Now by application of the observability of K with respect to L and P we deduce $s''a \in K^2$, i.e., $a \in K_{s''}^2 = M^2$. This means that $M \xrightarrow{a}$, which was to be proved. The rest follows from (i).

(\Leftarrow) Let $K \approx_{O(K, L)}$. Let us show that K is observable with respect to L and P . For this purpose, let $s \in K^2$ and $a \in A_c$ such that $s'a \in K^2$ and $sa \in L^2$ and $P(s) = P(s')$. Then $s \in K^2 \cap L^2$, i.e., $L \xrightarrow{s}$ and $K \xrightarrow{s}$, whence from (i) of Definition 8.2 inductively applied $K \approx_{O(K, L)} L_s$. Since $K \subseteq L$ and $s'a \in K^2$, we have $s' \in L^2$, because K^2 is prefix-closed. According to Lemma 8.1 we have $(K_s, L_s) \approx_{Aux}^{K, L}(K_{s'}, L_{s'})$. Notice that $sa \in L^2$ means $L_s \xrightarrow{a}$, and similarly $s'a \in K^2$ means $K_{s'} \xrightarrow{a}$. By (ii) of the definition of observability relation we obtain that $K_s \xrightarrow{a}$, i.e., $s'a \in K^2$. ■

Remark 8.3: In the sequel we need also another type of auxiliary relations $Aux(S)$ for the special case $S = \langle K \rangle$. We will write $Aux(K)$ instead of $Aux(\langle K \rangle)$. Notice that it is possible to extend the definition of $Aux(S)$ to $Aux(Pwr(S))$ with the only difference, that the propagation of this relation is realized by unions of nondeterministic transitions, in particular by deterministic weak transitions. In the case of the final automaton of partial languages similar construction of observational indistinguishability relation is to be realized on $Pwr(suffix(K))$. Now we prepare the coinductive definition of the supervised product. This definition will consider arguments from $Pwr(suffix(K))$ and $Pwr(suffix(L))$ rather than from DK and DL . In fact we will work with unions of the form $\bigcup_{i=1}^k K_{s_i} \in Pwr(suffix(K))$, where $P(s_1) = \dots = P(s_k)$. In order to keep the notation simple, we will use an extension of $Aux(K)$ to such unions of derivatives. In the definition of supervised product this will be needed.

Now we give a formal definition of $Aux(K)$ extended to $Pwr(suffix(K))$.

DEFINITION 8.3 (*Extension of $Aux(K)$ from DK to $Pwr(suffix(K))$* .) A binary relation $Aux(K) \subseteq (Pwr(suffix(K)))^2$, called observational indistinguishability relation is the smallest relation satisfying:

- (i) $\langle (K, K) \in Aux(K) \rangle$
- (ii) If $\langle M, N \rangle \in Aux(K)$ then $\forall a \in A : M \xrightarrow{a} M_a$ and $N \xrightarrow{a} N_a \Rightarrow \langle M_a, N_a \rangle \in Aux(K)$
- (iii) If $\langle M, N \rangle \in Aux(K)$ then $\forall m, n \in \mathbb{Z}_+ : \text{if } M \xrightarrow{\varepsilon} M_1, M \xrightarrow{\varepsilon} M_2, \dots, M \xrightarrow{\varepsilon} M_n, \text{ and } N \xrightarrow{\varepsilon} N_1, \dots, N \xrightarrow{\varepsilon} N_m, \text{ then } \langle \bigcup_{i=1}^n M_i, \bigcup_{j=1}^m N_j \rangle \in Aux(K)$.

Clearly, a natural extension of Lemma 5.1 holds. Namely, $\langle \cup_{i=1}^k K_{s_i}, \cup_{j=1}^l L_{t_j} \rangle \in \text{Aux}(K)$, where $P(s_1) = \dots = P(s_k)$ and $P(t_1) = \dots = P(t_l)$ iff $P(s_1) = P(t_1)$, which implies naturally $P(s_i) = P(t_j) \forall i, j$. The notation $\cup_{i=1}^k K_{s_i} \approx_{\text{Aux}}^K \cup_{j=1}^l L_{t_j}$ is also used.

DEFINITION 8.4 (*Supervised product under partial observations.*) Define the following binary operation on (partial) languages called supervised product under partial observations for all $M \in \text{Pwr}(\text{suffix}(K))$ and $N \in \text{Pwr}(\text{suffix}(L))$:

$$\left(M /_{\cup}^O N \right)_a =$$

- (1) $M_a /_{\cup}^O N_a$ if $M \xrightarrow{a}$ and $N \xrightarrow{a}$;
- (2) $\left(\cup_{\{M' : \langle M', M \rangle \in \text{Aux}(K)\}} M'_a \right) /_{\cup}^O N_a$ if $Ma \xrightarrow{a}$ and $\exists M' \in DK : M' \approx_{\text{Aux}}^K M$ such that $M' \xrightarrow{a}$ and $N \xrightarrow{a}$ and $a \in A_c \cup A_o$
- (3) $\emptyset /_{\cup}^O N_a$ if $Ma \xrightarrow{a}$ and $\forall M' \in DK : M' \approx_{\text{Aux}}^K M : M' \xrightarrow{a}$ and $N \xrightarrow{a}$ and $a \in A_{uc} \cap A_o$
- (4) $M /_{\cup}^O N_a$ if $M \xrightarrow{a}$ and $N \xrightarrow{a}$ and $a \in A_{uc} \cap A_{uo}$;
- (5) \emptyset otherwise

and $(M /_{\cup}^O N_a)$ iff N .

Remark 8.4:

1. According to Observation 3.4, $DL \subseteq \text{Pwr}(\text{suffix}(L))$ and since $K \subseteq L$ also $DK \subseteq \text{Pwr}(\text{suffix}(L))$.
2. It follows from the definition of supervised product that $K \subseteq (K /_{\cup}^O L) \subseteq L$. Both inclusions can be verified by construction of the corresponding simulation relations. Let us show that $K \subseteq (K /_{\cup}^O L)$. Consider the following relation:

$$R(K, L) = \left\{ \langle K_w, (K /_{\cup}^O L)_w \rangle \mid w \in K^2 \right\}.$$

It easy to see that $R(K, L)$ is a simulation relation proving the claimed inclusion. Take $w \in K^2$.

- (i) If $K_w \downarrow$, then $w \in K^1$, i.e., $w \in L^1$, which means $L_w \downarrow$. Furthermore, it follows from the Definition of 8.4 that $(K /_{\cup}^O L)_w = K_w /_{\cup}^O L_w$. Therefore, $(K /_{\cup}^O L)_w \downarrow$.
- (ii) if for $a \in A : K_w \xrightarrow{a}$, then $(K /_{\cup}^O L)_{wa} = (K_w /_{\cup}^O L_w)_a = K_{wa} /_{\cup}^O L_{wa}$, i.e., $(K /_{\cup}^O L)_w \xrightarrow{a}$ and $\langle K_w, (K /_{\cup}^O L)_w \rangle \in R(K, L)$.

As a consequence we conclude that the range of supervised product is again $\text{Pwr}(\text{suffix}(L))$, i.e., the supervised product can be also viewed as a (partial) binary operation on $\text{Pwr}(\text{suffix}(L))$.

The definition of supervised product under partial observations is quite complicated due to the interconnections between observability and controllability that must be taken into account. It deserves additional comments. Notice that several cases must be distinguished. First of all, by (1) the controller allows any event that does not exit from its (supervisor) language. A controllable event is enabled when the supervisor observes $s \in A^*$ iff there exists a string with the same projection as s that can be continued by this event within the supervisor's language, which is included in (2). The controller also enables all uncontrollable events that are possible in the plant, but the future actions depend on whether the occurred uncontrollable event is observable or not. If the uncontrollable event is unobservable then the first component of the supervised product need not move, but only the second component is updated as is seen from (4) above. In the case that the uncontrollable event a is observable, there must be further specified whether there exists a derivative indistinguishable from a derivative currently considered that can make an a -transition (i.e., there exists a string that has the same projection as s that can be continued by a within the supervisor's language), in which case the action is the same as for controllable events (i.e., this case is included in (2) above), or whether there is no such derivative, which means that only uncontrollable events that are possible in the plant are allowed in the future. The latter case corresponds to the term containing the zero partial language and is labeled by (3) above. In any other case (5) the controllable events are disabled by the supervisor. We have thus the coinductive definition of the closed-loop language that gives a clear picture of what is the mechanism of discrete-event control under partial observations.

Now we proceed in the same way as in the case of full observations. Let us define the following relation called partial bisimulation under partial observations.

DEFINITION 8.5 (*Partial bisimulation.*) *A binary relation $R(K, L) \subseteq DK \times DL$ is called a partial bisimulation under partial observations if for all $\langle M, N \rangle \in R(K, L)$:*

- (i) $o(M) = o(N)(M \downarrow \text{iff } N \downarrow)$
- (ii) $\forall a \in A : M \xrightarrow{a} \Rightarrow N \xrightarrow{a}$ and $\langle M_a, N_a \rangle \in R(K, L)$
- (iii) $\forall u \in A_{uc} : N \xrightarrow{u} \Rightarrow M \xrightarrow{u}$ and $\langle M_u, N_u \rangle \in R(K, L)$
- (iv) $\forall a \in A_c : N \xrightarrow{a}$ and $(\exists (M', N') \approx_{Aux}^{K, L} (M, N) : M' \xrightarrow{a}) \Rightarrow M \xrightarrow{a}$.

For $M \in DK$ and $N \in DL$ we write $M \approx_U^{O(K, L)} N$ whenever there exists a partial bisimulation under partial observations $R(K, L)$ such that $\langle M, N \rangle \in R(K, L)$. This relation is called partial bisimilarity under partial observations.

Remark 8.5: Notice that (i) relates the marking components of the languages involved and (ii) corresponds to the language simulation (inclusion), while (iii) to the controllability and (iv) to the observability condition. Observe also that the second statement on the right hand side of (iii) follows from the corresponding first statement and (ii).

Now we are ready to formulate the main theorem, which gives a coalgebraic formulation of the controllability and observability theorem (Cassandras and Lafontaine, 1999) in supervisory control of DES with partial observations.

THEOREM 8.6 *Let $K \subseteq L$ be given partial languages. Then $K \approx_U^{O(K,L)} L$ iff $K = K/\cup L$. The supervised product under partial observations of the languages K and L equals K iff K and L are partially bisimilar in the sense of Definition 8.5.*

Proof: (\Rightarrow) Let $K \approx_U^{O(K,L)} L$. Define

$$R(K, L) = \left\{ \langle M, (M/\cup N) \rangle \mid M \in DK, N \in DL \text{ and } M \approx_U^{O(K,L)} N \right\}.$$

According to the coinduction proof principle it is sufficient to prove that $R(K, L)$ is a bisimulation, because then $K \approx_U^{O(K,L)} L$, i.e., $\langle K, (K/\cup L) \rangle \in R(K, L)$ implies that $K = (K/\cup L)$. Let $\langle M, (M/\cup N) \rangle \in R(K, L)$.

- (i) $M \downarrow$ iff $N \downarrow$ (because $M \approx_U^{O(K,L)} N$) iff $(M/\cup N) \downarrow$.
- (ii) If $M \xrightarrow{a}$ for $a \in A$ then by (ii) of Definition 8.5 $N \xrightarrow{a}$ and $M_a \approx_U^{O(K,L)} N_a$. Thus, $(M/\cup N) \xrightarrow{a} (M/\cup N)_a = (M_a/\cup N_a)$, and $\langle M_a, (M/\cup N)_a \rangle \in R(K, L)$.
- (iii) If $(M/\cup N) \xrightarrow{a}$, then according to the (coinductive) definition of the supervised product we have four possibilities: either $M \xrightarrow{a}$ and $N \xrightarrow{a}$, or $M \xrightarrow{a}$ and $\exists M' \approx_{Aux}^K M : M' \xrightarrow{a}$ and $N \xrightarrow{a}$ and $a \in A_c \cup A_o$, or $M \xrightarrow{a}$ and $\forall M' \in DK : M' > \approx_{Aux}^K M : M' \xrightarrow{a}$ and $a \in A_{uc} \cap A_o$; or, finally, $M \xrightarrow{a}$ and $N \xrightarrow{a}$ and $a \in A_{uc} \cap A_o$. Notice however that the second case is contradicted by (iv) of Definition 8.5: it is sufficient to see that if $\exists M' \approx_{Aux}^K M : M' \xrightarrow{a}$ and $N \xrightarrow{a}$, then $\exists (M', N') \approx_{Aux}^{K,L} (M, N) : M' \xrightarrow{a}$ and $N \xrightarrow{a}$. Indeed, by Lemma 5.1 applied for $S = DK$ (recall that $M \in DK$) $M = K_s$ and $M' = K_{s'}$ for some $s, s' : P(s') = P(s)$, then it is sufficient to put $N' = L_{s'}$, which clearly exists, because $K \subseteq L$. The third and the fourth cases (with $a \in A_{uc}$) are both impossible due to (iii) of the same definition. Hence only the first possibility can occur, which brings us back to the previous case (ii).

(\Leftarrow) Let us show that the following relation is a partial bisimulation under partial observations. Define

$$T(K, L) = \left\{ \langle M, N \rangle \mid M \in DK, N \in DL \text{ and } M = (M/\cup N) \right\}.$$

Let $\langle M, N \rangle \in T(K, L)$.

- (i) $M \downarrow$ iff $(M/\cup N) \downarrow$ (from the definition of $T(K, L)$) iff $N \downarrow$ (from Definition 8.4).
- (ii) If $M \xrightarrow{a}$ for $a \in A$ then $(M/\cup N) \xrightarrow{a}$ and clearly (from the coinductive definition of supervised product) $N \xrightarrow{a}$. Also $M_a = (M/\cup N)_a = (M/\cup N_a)$, whence $\langle M_a, N_a \rangle \in T(K, L)$.

- (iii) If $N \xrightarrow{u}$ for $u \in A_{uc}$ then $(M/\overset{O}{U}N) \xrightarrow{u}$ according to the definition of supervised product. Thus $M \xrightarrow{u}$ as well. Furthermore, $M_u = (M/\overset{O}{U}N)_u = (M_u/\overset{O}{U}N_u)$ which means $\langle M_u, N_u \rangle \in T(K, L)$.
- (iv) If $N \xrightarrow{a}$ for $a \in A_c$ and $(\exists(M', N') \approx_{Aux}^{K, L}(M, N) : M' \xrightarrow{a})$ then from the definition of supervised product (the second case occurs: note that in particular $M' \approx_{Aux}^K M$, $(M/\overset{O}{U}N) \xrightarrow{a}$, i.e., $M \xrightarrow{a}$, which was to be shown.

Finally, similarly as in the case of full observations, there is the following characterization of partial bisimilarity.

COROLLARY 8.7 $K \approx_U^{O(K, L)} L$ iff $(K^2 A_{uc} \cap L^2 \subseteq K^2, K \approx_{O(K, L)} L, \text{ and } K^1 = K^2 \cap L^1)$.

Proof: It is quite analogous to the full observations case. In particular, notice that partial bisimulation under partial observations implies partial bisimulation as it has been first introduced in Rutten (1999). Thus, it is sufficient to consider only the additional property of observability, which appears on both sides of the claimed equivalence. ■

9. Infimal closed observable superlanguages and maximal observable sublanguages

This section contains only new results. In the last section we have introduced an operation on partial languages called supervised product under partial observations. This operation corresponds to the behavior of the supervised discrete-event system modeled by a partial automaton using the centralized version of *C&P* control architecture in the terminology of Yoo and Lafortune (2002). We call this control architecture in the centralized case simply permissive. Let $K = (K^1, K^2)$ be the desired behavior (partial language) and V be the supervisory controller. Then $\forall s \in A_o^*$ the associated control law (events enabled after V observes s) is:

$$\gamma_P(V, s) = A_{uc} \cup \{a \in A_c : \exists s' \in K^2 \text{ with } P(s') = P(s) \text{ and } s'a \in K^2\}.$$

The centralized counterpart of the *D&A* control architecture we call antipermissive and it is given by the following control law: $\forall s \in A_o^*$

$$\begin{aligned} \gamma_A(V, s) &= A_{uc} \cup \{a \in A_c : \forall s' \in K^2 \text{ with } P(s') \\ &= P(s) \text{ we have } s'a \in L^2 \Rightarrow s'a \in K^2\}. \end{aligned}$$

There is also an antipermissive control architecture counterpart of the supervised product, but its definition is postponed towards the end of this section. Let us call it antipermissive supervised product. We will show that it cannot be defined by coinduction, however in a similar way using suitable automata representations. Note that the permissiveness or antipermissiveness is related to the observability (controllable events). Recall that the control policy must be, by definition, permissive with respect to uncontrollable events in the sense that these are always enabled.

Remark 9.1: We consider from now on an order relation on partial languages induced by their second component only, i.e., we write $K \subseteq L$ iff $K^2 \subseteq L^2$. The same applies for infimum (supremum), and maximum operations. Note that only the second condition of simulation relations must be checked to prove such defined inclusion of partial languages.

Let us recall the coinductive definition of the supervised product in the case of full observations from Rutten, 1999.

DEFINITION 9.1 Define the following binary operation on (partial) languages for all $K, L \in \mathcal{L}$ and $\forall a \in A$:

$$(K/\cup L)_a = \begin{cases} K_a/\cup L_a & \text{if } K \xrightarrow{a} \text{ and } L \xrightarrow{a} \\ 0/\cup L_a & \text{if } K \not\xrightarrow{a} \text{ and } L \xrightarrow{a} \text{ and } a \in A_{uc} \\ \emptyset & \text{otherwise} \end{cases}$$

and $(K/\cup L) \downarrow$ iff $L \downarrow$.

THEOREM 9.2 $(K/\cup L) = \inf(\bar{C}(K, L)) = \inf\{M \supseteq K : M \text{ is controllable with respect to } L \text{ and } A_{uc}\}$, i.e., $K/\cup L$ equals the infimal controllable superlanguage of K .

Proof: Let us show that $K/\cup L$ is a superlanguage of K that is controllable with respect to L and A_{uc} . It is clear from the definition of supervised product that $K \subseteq (K/\cup L)$ in the sense of Remark 9.1. Let us show that $K/\cup L$ is controllable with respect to L and A_{uc} . It is sufficient to prove that the following relation is a control relation.

$$C = \{\langle (K/\cup L), L \rangle \mid K, L \in \mathcal{L}\}.$$

- (i) Let $(K/\cup L) \xrightarrow{a}$ and $L \xrightarrow{a}$ for $a \in A$. Then by coinductive definition of $K/\cup L$ either $(K/\cup L)_a = (K_a/\cup L_a)$ or $(K/\cup L)_a = (0/\cup L_a)$. However, by definition of C in both cases we have $\langle (K/\cup L)_a, L_a \rangle \in C$.
- (ii) If $L \not\xrightarrow{u}$ for $u \in A_{uc}$, then either $K \not\xrightarrow{u}$ and hence $(K/\cup L) \not\xrightarrow{u}$ or $K \xrightarrow{u}$, but according to the definition of $K/\cup L$ we have still $(K/\cup L) \xrightarrow{u} (0/\cup L_u)$.

It remains to show the infimality. Let $M \supseteq K$ be controllable with respect to L and A_{uc} .

$$R = \{\langle (K/\cup L), M \rangle \mid K, L, M \in \mathcal{L} : K \subseteq M \subseteq L, \text{ and } M^2 A_{uc} \cap L^2 \subseteq M^2\}.$$

satisfies (ii) of the definition of simulation relations. Let $(K/\cup L) \xrightarrow{a}$ for $a \in A$. According to the definition of $K/\cup L$ we have two possibilities: either $K \xrightarrow{a}$ and $L \xrightarrow{a}$, in which case $(K/\cup L)_a = K_a/\cup L_a$ or $K \not\xrightarrow{a}$ and $L \xrightarrow{a}$ and $a \in A_{uc}$. In the first case we have $M \xrightarrow{a}$ simply because $K \xrightarrow{a}$ and $K \subseteq M$, while in the latter case we have $M \xrightarrow{a}$ because of the controllability of M with respect to L and A_{uc} (by Definition 6.3 of control relations for $a \in A_{uc} : L \xrightarrow{a} \Rightarrow M \xrightarrow{a}$). Moreover in both cases $\langle (K/\cup L)_a, L_a \rangle \in R$. ■

Although the infimal controllable superlanguages are important (Lafortune and Chen, 1990), supremal controllable sublanguages are even more interesting as least restrictive

solutions of full observation supervisory control problems (Wonham and Ramadge, 1987). In Rutten (1999) an algorithm for the computation of supremal controllable sublanguages, based on control relations, has been presented. It turns out that it is also possible to define the supremal controllable sublanguage by coinduction.

DEFINITION 9.2 Define the following binary operation on (partial) languages for all $K, L \in \mathcal{L}$ and $\forall a \in A$:

$$(K/\mathcal{C}L)_a = \begin{cases} K_a/\mathcal{C}L_a & \text{if } K \xrightarrow{a} \text{ and } L \xrightarrow{a} \\ & \text{and if } \forall u \in \forall_{uc}^* : L_a \xrightarrow{u} \Rightarrow K_a \xrightarrow{u} \\ \emptyset & \text{otherwise} \end{cases}$$

and $(K/\mathcal{C}L) \downarrow$ iff $L \downarrow$.

THEOREM 9.3 $(K/\mathcal{C}L) = \sup(\underline{\mathcal{C}}(K, L)) = \sup\{M \subseteq K : M \text{ is controllable with respect to } L \text{ and } A_{uc}\}$, i.e., $(K/\mathcal{C}L)$ equals the supremal controllable sublanguage of K .

Proof: First we show that $K/\mathcal{C}L$ is a sublanguage of K that is controllable with respect to L and A_{uc} . It is clear from the definition of $K/\mathcal{C}L$ that $(K/\mathcal{C}L) \subseteq K$ in the sense of Remark 9.1. Indeed, if we take $U = (K/\mathcal{C}L)_w = K_w/\mathcal{C}L_w$ and $V = K_w$ for some $w \in (K/\mathcal{C}L)^2$, then $U \xrightarrow{a} \Rightarrow V \xrightarrow{a}$. Let us show that $K/\mathcal{C}L$ is controllable with respect to L and A_{uc} . It is sufficient to prove that the following relation is a control relation (Definition 6.3).

$$C = \left\{ \left\langle (K/\mathcal{C}L)_w, L_w \right\rangle \mid w \in (K/\mathcal{C}L)^2 \right\}.$$

Take a pair $M = (K/\mathcal{C}L)_s$ and $N = L_s$ for some $s \in (K/\mathcal{C}L)^2$.

- (i) Let $(K/\mathcal{C}L)_s \xrightarrow{a}$ and $L_s \xrightarrow{a}$ for $a \in A$. Then by coinductive definition of $K/\mathcal{C}L$ we have $(K/\mathcal{C}L)_{sa} = (K_{sa}/\mathcal{C}L_{sa})$, which by definition of C means that $\langle (K/\mathcal{C}L)_{sa}, L_{sa} \rangle \in C$.
- (ii) Let $L_s \xrightarrow{u}$ for $u \in A_{uc}$. Since $(K/\mathcal{C}L) \xrightarrow{s}$, we have by definition 9.2 that $K \xrightarrow{s}$ and $L \xrightarrow{s}$ and $\forall u \in A_{uc}^* : L_s \xrightarrow{u} \Rightarrow K_s \xrightarrow{u}$. Therefore we deduce $K_s \xrightarrow{u}$. Furthermore, $\forall v \in A_{uc}^* : L_{su} \xrightarrow{v} \Rightarrow L_s \xrightarrow{uv} \Rightarrow K_s \xrightarrow{uv} \Rightarrow K_{su} \xrightarrow{v}$, because $uv \in A_{uc}^*$ and $(K/\mathcal{C}L) \xrightarrow{s}$. Hence $(K/\mathcal{C}L)_s \xrightarrow{u}$, which proves that C is a control relation, i.e., $K/\mathcal{C}L$ is controllable with respect to L and A_{uc} .

It remains to show the supremality. Let $M \subseteq K$ be controllable with respect to L and A_{uc} . In order to show that $M^2 \subseteq (K/\mathcal{C}L)^2$, we consider

$$R = \left\{ \left\langle M_w, (K/\mathcal{C}L)_w \right\rangle \mid w \in M^2 \right\}.$$

Take $\langle M_s, (K/\mathcal{C}L)_s \rangle \in R$ for some $s \in M^2$. Let $M_s \xrightarrow{a}$ for $a \in A$. Then $K_s \xrightarrow{a}$, and $L_s \xrightarrow{a}$, since $M \subseteq K \subseteq L$. In order to prove that $(K/\mathcal{C}L)_s \xrightarrow{a}$, it remains to show that $\forall u \in A_{uc}^* : L_{sa} \xrightarrow{u} \Rightarrow K_{sa} \xrightarrow{u}$. But this is straightforward: if $L_{sa} \xrightarrow{u}$, then by controllability of M we deduce $M_{sa} \xrightarrow{u}$, thus from $M \subseteq K$ it follows that $K_{sa} \xrightarrow{u}$. It follows that R satisfies (ii) of simulation relations, i.e., $M \subseteq K/\mathcal{C}L$.

Let us now suppose that controllability is not an issue. Recall that an algorithm for supremal controllable sublanguage has been given in Rutten (1999). We have also shown that the supervised product in the case of full observations defined therein provides the infimal controllable superlanguage. As a byproduct we have its coinductive definition. In the case of partial observations, we can now separate the issue of controllability from observability and introduce the following modification of supervised product. Note that a similar method (separating the issue of controllability from observability) has been used in Bergeron (1993) for automata (supervisor) approach. Unlike the methods known from the literature (Bergeron, 1993; Rudie and Wonham, 1990) for infimal closed and observable superlanguages our coalgebraic approach (the following coinductive definition) has a direct algorithmic character, because coinduction defines the resulting structure event by event).

DEFINITION 9.3 Define the following binary operation on (partial) languages for all $M \in \text{Pwr}(\text{suffix}(K))$ and $N \in \text{Pwr}(\text{suffix}(L))$ and $\forall a \in A$:

$$(M/^{\circ}N)_a = \begin{cases} M_a/^{\circ}N_a & \text{if } M \xrightarrow{a} \text{ and } N \xrightarrow{a} \text{ and} \\ & \exists s \in K^2 : M = K_s \text{ and } N = L_s \\ \cup_{\{M' : (M', M) \in \text{Aux}(K)\}} M'_a/^{\circ}N_a & \text{if } M \xrightarrow{a} \text{ and } \exists M' \in DK : \\ & M' \approx_{\text{Aux}}^K M \text{ such that } M' \xrightarrow{a} \\ & \text{and } N \xrightarrow{a} \text{ and } a \in A_c \\ \emptyset & \text{otherwise} \end{cases}$$

and $(M/^{\circ}N) \downarrow$ iff $N \downarrow$.

The new operation has the following pleasant property:

THEOREM 9.4 $(K/^{\circ}L) = \inf(\bar{O}(K, L, P)) = \inf\{M \supseteq K : M \text{ is observable with respect to } L \text{ and } P\}$. The infimal observable superlanguage of K equals $(K/^{\circ}L)$.

Proof: It can be proven by coinduction using the formula for $\inf(\bar{O}(K, L, P))$ given in Rudie and Wonham (1990). Another, more direct, way is to show that $K/^{\circ}L$ is an observable partial language containing K that is smaller than any other observable superlanguage of K .

Let us show that $K/^{\circ}L$ is a superlanguage of K that is observable with respect to L . It is clear from the definition 9.3 that $(K/^{\circ}L)^2$ is a superlanguage of K^2 . Formally it can be checked by constructing an obvious simulation relation. Let us show that $K/^{\circ}L$ is observable with respect to L . According to Theorem 7.2 we put

$$O = \left\{ \left\langle (K/^{\circ}L)_u, L_u \right\rangle \mid u \in (K/^{\circ}L)^2 \right\}$$

and show that O is an observability relation on $D(K/^{\circ}L) \times DL$. Take a pair $\langle U, V \rangle \in R$. We can assume that $U = (K/^{\circ}L)_s$ and $V = L_s$ for some $s \in (K/^{\circ}L)^2$.

- (i) Let $a \in A$ such that $(K/^{\circ}L)_s \xrightarrow{a}$. It follows from the Definition 9.3 that $L_s \xrightarrow{a}$ and from the definition of O that $\langle (K/^{\circ}L)_{sa}, L_{sa} \rangle \in O$.

- (ii) Let $a \in A_c$ such that $L_s \xrightarrow{a}$ and there exists $M \in D(K/O_L) : M \approx_{Aux}^{K/O_L, L} (K/O_L)$ with $M \xrightarrow{a}$. It means that there exist $s', s'' \in A^*$ such that $P(s'') = P(s')$, $(K/O_L)_s = (K/O_L)_{s''}$, $L_s = L_{s''}$, and $M = (K/O_L)_{s'} \xrightarrow{a}$. According to Definition 9.3 inductively applied there exist $s_i \in A^*$, $i \in I$ such that $(K/O_L)_{s'} = (\cup_{i \in I} K_{s_i}) / O_{L_{s'}}$, where $P(s_i) = P(s') \forall i \in I$. Notice, that it can be that $I = \{1\}$ and $s_1 = s'$. Since $M \xrightarrow{a}$, by Definition 9.3 either there exist $s_j, j \in J \subseteq I$ such that $K_{s_j} \xrightarrow{a}$ for $j \in J$ and $M_a = (\cup_{j \in J} K_{s_j a}) / O_{L_{s' a}}$, or there exist $w_k, k \in K$ such that $K \xrightarrow{w_k a}$, $P(w_k) = P(s')$ and $M_a = (\cup_{k \in K} K_{w_k a}) / O_{L_{s' a}}$. Since also $P(w_k) = P(s'')$ for all $k \in K$, we deduce finally that according to Definition 9.3 in both cases there must be $(K/O_L)_s = (K/O_L)_{s''} \xrightarrow{a}$, which proves that O is an observability relation.

The last step of the proof is to show that if $M \supseteq K$ is a language which is observable with respect to L and P , then $(K/O_L) \subseteq M$. It is sufficient to prove that

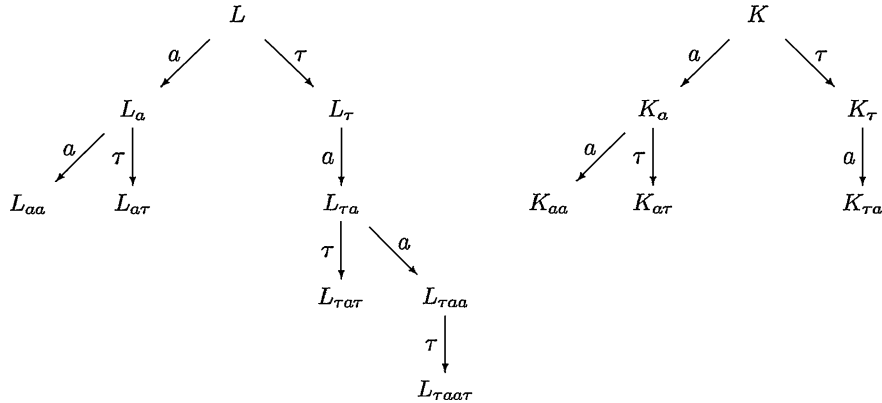
$$R = \left\{ \left\langle (K/O_L)_u, M_u \right\rangle \mid u \in (K/O_L)^2 \text{ and } K \subseteq M \approx_{O(M,L)} L \right\}$$

satisfies (ii) of simulation relation.

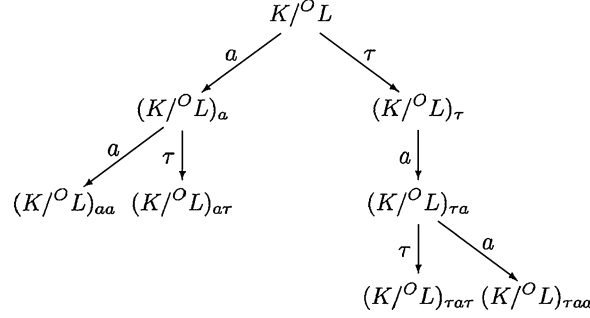
Take a pair $\langle U, V \rangle \in R$. We can assume that $U = (K/O_L)_w$ and $V = M_w$ for some $w \in (K/O_L)^2$. Let $U \xrightarrow{a}$. There exist $s_i \in K^2$ for i in some index set I such that $P(s_i) = P(w) \forall i \in I$ and $U = (\cup K_{s_i}) / O_{L_w}$. Now, $U \xrightarrow{a}$ implies that either $U = K_w / O_{L_w} \xrightarrow{a} K_{wa} / O_{L_{wa}}$ or there exists $J \subseteq I$ such that $K_{s_j} \xrightarrow{a}$ for $j \in J$ and $U_a = (\cup K_{s_j a}) / O_{L_{wa}}$ and $a \in A_c$ or finally there exist $w_k \in A^*$, $k \in K$ such that $P(w_k) = P(w)$, $a \in A_c$, and $U_a = (\cup K_{w_k a}) / O_{L_{wa}}$. In the first case we have directly $wa \in K^2$, i.e., $V = M_w \xrightarrow{a}$. In the second case $w \in M^2$ (because $V = M_w$), $s_j \in M^2$, because $s_j \in K^2 \subseteq M^2$, $s_j a \in M^2$, $wa \in L^2$, $a \in A_c$ (because we are in the second case of Definition 9.3), and $P(s_j) = P(w)$. Therefore $wa \in M^2$, because M is observable with respect to L and P . Finally, in the third case we have similarly $w \in M^2$, $w_k \in M^2$, $w_k a \in M^2$, $wa \in L^2$, $a \in A_c$, and $P(w_k) = P(w)$, which gives also $wa \in M^2$. Hence $V = M_w \xrightarrow{a}$, and trivially $\langle U_a, V_a \rangle \in R$, which was to be shown. ■

To illustrate the new operation, consider the following example.

Example 4: We consider prefix-closed languages K^2 and L^2 given by the following tree automata, different from $\langle K \rangle$, resp. $\langle L \rangle$ from \mathcal{L} ! The marked components are not considered, $A = \{a, \tau\}$, and $A_0 = \{a\}$.



Then



We have for instance $(K/O L)_{\tau a \tau} = (K_{\tau a}/O L_{\tau a})_{\tau} = K_{a\tau}/O L_{\tau a \tau}$ according to the Definition 9.3, because $K_{\tau a} \xrightarrow{\tau} K_{a\tau}$ but there exists $K_a \approx_{Aux}^K K_{\tau a}$ with $K_a \xrightarrow{\tau} K_{a\tau}$. Also, $K/O L$ is indeed the infimal observable superlanguage of K as stated in Theorem 9.4.

Recall that we use an order relation with respect to the second components of partial languages (Remark 1). As for the original definition of supervised product it can be shown in a similar way that

THEOREM 9.5. $(K/O L) = \inf(\bar{CO}(K, L, P)) = \inf\{M \supseteq K : M \text{ is controllable with respect to } L \text{ and } A_{uc} \text{ and observable with respect to } L \text{ and } P\}$. $(K/O L)$ equals the infimal controllable and observable superlanguage of K .

The proof of this theorem is similar to that of Theorems 9.2 and 9.4. As a direct consequence, the supervised product is monotone with respect to the specification:

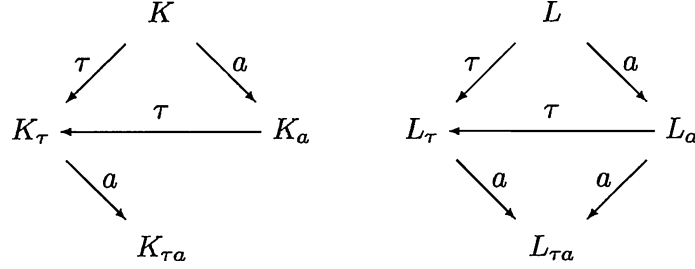
COROLLARY 9.6 For (partial) languages $K \subseteq K'$ we have $(K/O L) \subseteq (K'/O L)$.

Note that the infimality of the above defined operations is in both cases only with respect to the second (closed) components of the partial languages involved. The following example shows that the infimality with respect to the marking component can not hold.

Example 5: Take $K = (\{a\}, \{\varepsilon, a, \tau, \tau a, \tau ab\})$, $L = (\{a, ab\}, \{\varepsilon, a, ab, ab\tau, \tau, \tau a, \tau ab\})$, and $M = (\{a, \tau\}, \{\varepsilon, a, ab, \tau, \tau a, \tau ab\})$. Then $K/O L = (\{a, ab\}, \{\varepsilon, a, ab, \tau, \tau a, \tau ab\})$. Hence $K \subseteq M$, M is observable with respect to L and P , but $(K/O L)^1 \not\subseteq M^1$, because $ab \in (K/O L)^1 \setminus M^1$.

Similar examples can be constructed for $K/_O L$ or $K/_O L$. Before we study the antipermissive control law, we consider the case, where controllability is again not an issue. Unlike the permissive case, the fact that $Aux(K, L)$ is not an equivalence relation on $DK \times DL$ creates difficulties as is illustrated in the example below.

Example 6: Consider the following specification and plant languages:



We see that $K_\tau = K_{a\tau}$ and $L_\tau = L_{a\tau}$. This is a problem, because using the antipermissive control law one would like to allow $a \in A$ after observation of s if $M = K_s$ satisfies the condition

$$M \xrightarrow{a} \text{ and } \forall (M', N') \in DK \times DL : (M', N') \approx_{Aux}^{K,L} (M, N) : (N' \xrightarrow{a} \Rightarrow M' \xrightarrow{a}). \quad (1)$$

This condition seems to be a natural coalgebraic interpretation of the antipermissive control law $\gamma_A(V, s)$ introduced above. But the state K_τ can be reached by two strings, whose projections are ε and a , and this creates a difficulty. On one hand after $s = a$, event a should be disabled at $K_\tau = K_{a\tau}$, since $(K_\tau, L_\tau) \approx_{Aux}^{K,L} (K_a, L_a)$ and $L_a \xrightarrow{a}$, while $K_a \not\xrightarrow{a}$. On the other hand, after $s = \varepsilon$, event a can be enabled at K_τ , since the condition in the antipermissive control law $\gamma_A(V, s)$ is fulfilled. Using the minimal representation and Condition (1) we would define by coinduction a different language than the language of the closed-loop system. The problem is that the states of the minimal representations that lie in the intersection of two observer states might lead to the conflicts as is shown in this example. In order to avoid the above ambiguities and define the closed-loop system under the antipermissive control law, suitable (“unfolded”) automata representations, in general different from minimal ones, must be used.

In order to avoid the undesirable situation of the above example we use in the following definition underlying representations of languages K and L by automata S_1 and S , where S_1 is a subautomaton of S such that $Aux(S_1)$ is an equivalence relation. We have proven in Section 4 that the condition of S_1 being state-partition automaton (Yoo et al., 2001) is stronger, i.e., it guarantees that $Aux(S_1)$ is an equivalence relation. It is known how to construct such representations Cho and Marcus (1989b) or Yoo et al. (2001).

Let s_0 denote the common initial state of S_1 and S . The transition structure of S_1 and S is denoted by \rightarrow_1 and \rightarrow , respectively. In the following algorithm we compute a sublanguage of K that is observable with respect to L and P using the antipermissive control law.

ALGORITHM 3 Let automata S_1 and S represent K and L , respectively, be such that S_1 is a subautomaton of S and $Aux(S_1)$ is an equivalence relation. Let us construct the partial automaton $\tilde{S} = \langle \tilde{\delta}, \tilde{\tau} \rangle$ with the $\tilde{\tau}$ denoted by \rightarrow .

1. Put $\tilde{S} := \{s_0\}$.
2. For any $s \in \tilde{S}$ and $a \in A$ we put $s \xrightarrow{a} s_a$ if $\forall s' \in S_1 : s' \approx_{Aux(S_1)} s : (s' \xrightarrow{a} \Rightarrow s' \xrightarrow{a}_1)$ and we put in the case $s \xrightarrow{a}$, also $\tilde{S} := \tilde{S} \cup \{s_a\}$.
3. For any $s \in \tilde{S}$ we put $\tilde{\delta}(s) = o(s)$.

Let us denote by \tilde{l} the unique (behavior) homomorphism given by finality of \mathcal{L} .

THEOREM 9.7 $\tilde{l}(s_0)$ is an observable sublanguage with respect to L and P . Moreover, if S_1 is a state-partition automaton, then $\tilde{l}(s_0)$ contains the supremal (L,P) —normal sublanguage of K .

Proof: To prove the observability of $\tilde{l}(s_0)$ we show that the following relation is an observability relation on $\tilde{S} \times S$.

$$O = \{ \langle (s_0)_u, (s_0)_u \rangle \mid u \in \tilde{l}(s_0) \}.$$

Then $\tilde{l}(s_0)$ is observable respect to L and P according to Theorem 5.6. Take a pair $\langle (s_0)_v, (s_0)_v \rangle \in O$ for some $v \in \tilde{l}(s_0)$.

- (i) If $(s_0)_v \xrightarrow{a}$, for $a \in A$, then clearly by construction of Algorithm 3 $(s_0)_v \xrightarrow{a}$. It is clear from the definition of O that $\langle (S_0)_{va}, (s_0)_{va} \rangle \in O$.
- (ii) Let $a \in A_c$ be such that $(s_0)_v \xrightarrow{a}$ and let there exist $s' \in \tilde{S} : s' \approx_{Aux(\tilde{S})} (s_0)_v$ with $s' \xrightarrow{a}$. By Lemma 5.1 there exist two strings $w, w' \in A^*$ such that $P(w) = P(w')$, $(s_0)_v = (s_0)_w$, and $s' = (s_0)_{w'} \xrightarrow{a}$. According to the construction of Algorithm 3 for any $s \approx_{Aux(S_1)} (s_0)_{w'}$ there must be $s \xrightarrow{a} \Rightarrow s \xrightarrow{a}_1$. In order to show that $(s_0)_v \xrightarrow{a}$, it must be that for any $q \approx_{Aux(S_1)} (s_0)_v$, there must be $q \xrightarrow{a} \Rightarrow q \xrightarrow{a}_1$. But using the fact that $Aux(S_1)$ is transitive and the fact that $s' \approx_{Aux(\tilde{S})} (s_0)_v$ implies that $s' \approx_{Aux(S_1)} (s_0)_v$, we obtain that $\langle s', q \rangle \in Aux(S_1)$. But this just means that for any $q \approx_{Aux(S_1)} (s_0)_v$ we have $q \xrightarrow{a} \Rightarrow q \xrightarrow{a}_1$, i.e., $(s_0)_v \xrightarrow{a}$, and O is an observability relation.

We show finally that the supremal (L, P) —normal sublanguage of K is contained in $\tilde{l}(s_0)$. Let N be a (L,P) —normal sublanguage of K . Then it is sufficient to show that

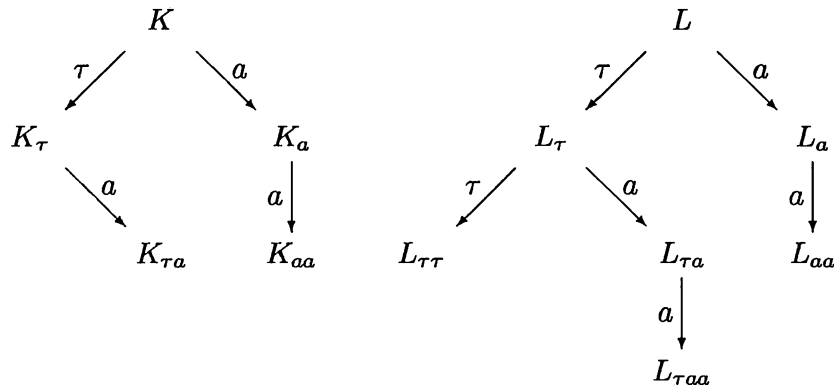
$$R = \{ \langle N_u, \tilde{l}(s_0)_u \rangle \mid u \in N^2 \}$$

satisfies (ii) of simulation relation in order to prove that $N^2 \subseteq \tilde{l}(s_0)^2$. Take an arbitrary pair $\langle N_w, \tilde{l}(s_0)_w \rangle \in R$ for some $w \in N^2$. Let $N_w \xrightarrow{a}$ for $a \in A$. Then also $K_w \xrightarrow{a}$, since $N \subseteq K$ and $L_w \xrightarrow{a}$ as well. This means that $(s_0)_{w \xrightarrow{a}_1}$ and $(s_0)_{w \xrightarrow{a}}$. In order to show that $\tilde{l}(s_0)_w \xrightarrow{a}$, i.e., $(s_0)_w \xrightarrow{a}$, we must prove that for any $q \approx_{Aux(S_1)} (s_0)_w : q \xrightarrow{a} \Rightarrow q \xrightarrow{a}_1$. There exist $v, v' : P(v) = P(v')$ such that $q = (s_0)_v$ and $(s_0)_w = (s_0)_{v'}$. Since S_1 is a state-partition automaton

and $(s_0)_w$ is in two possibly different states of the observer automaton, we conclude by the property of state-partition automaton that these two states of the observer automaton coincide. But this means that there exists $w' \in A^*$ such that $P(w) = P(w')$ and $q = (s_0)_{w'}$. Now $q \xrightarrow{a}$ means that $w'a \in L^2$. Using normality of N it follows from $wa \in N^2$ and $w'a \in L^2$ that $w'a \in N^2$. Therefore $w'a \in K^2$ (because $N \subseteq K$), which means that $q \xrightarrow{a} 1$. We conclude that $\tilde{I}(s_0)_{w'} \xrightarrow{a}$ and R satisfies (ii) of simulation relations, i.e., we have the inclusion $N^2 \subseteq \tilde{I}(s_0)^2$. Note that since N was an arbitrary (L, P) —normal sublanguage of K , the same inclusion must hold for the supremal (L, P) —normal sublanguage of K .

In the following example we show that $\tilde{I}(s_0)$ is not always a maximal observable sublanguage of K .

Example 7: We consider prefix-closed languages K^2 and L^2 given again by tree automata and we assume that all the states of both automata are marked. The alphabet is $A = \{a, b, \tau\}$, with $A_0 = \{a, b\}$.



Using Algorithm 3 we obtain the resulting automaton \tilde{S} , whose closed language is $\tilde{I}(s_0) = \{\varepsilon, a\}$. It is indeed an observable sublanguage of K^2 containing the supremal normal sublanguage $N = \{\varepsilon\}$. However, $\tilde{I}(s_0)$ is not a maximal observable sublanguage of K^2 , because $M = \{\varepsilon, a, aa\}$ is a larger observable sublanguage of K^2 . Notice also that $\tilde{I}(s_0)$ is not (L, P) —normal.

Note that because of the afore mentioned difficulties we do not present a coinductive definition of the antipermissive counterpart of supervised product that takes into account the issue of controllability, which would correspond to the definition of the antipermissive control policy. However it is possible to design an algorithm that describes the behavior of the closed-loop system under the antipermissive control policy similar to Algorithm 3. We remark that there is an asymmetry in the antipermissive control policy: it is imposed to be permissive with respect to the uncontrollable events, while it is antipermissive with respect to the controllable events. As a consequence specification K and the closed-loop language for the antipermissive control policy are not in general comparable.

Notice an important difference between the permissive and antipermissive control policy. Using the permissive control policy after having left from the specification language K by an uncontrollable event, there might still be some controllable events enabled in the future, while using the antipermissive control policy only uncontrollable events are enabled in such a situation.

To conclude, we have found an observable sublanguage that contains the supremal normal sublanguage. This is very useful, because supremal normal sublanguages are often too small (restrictive) in many concrete problems.

Our technique can be modified for constructing an observable and controllable sublanguage, because the idea in the coinductive definition of the supremal controllable sublanguage can be incorporated within Algorithm 3. In this way a ‘single-step’ algorithm for the computation of supremal normal and controllable sublanguages is developed in the next section.

10. ‘Single-step’ algorithms for supremal normal and controllable sublanguages

Now we show a ‘single-step’ algorithm for the computation of supremal normal sublanguages along the lines of Algorithm 3. The main idea is that the iterative procedure of Algorithm 1 is incorporated into Algorithm 2 using unobservable strings instead of events. Since we work with finite representations, our algorithm is still effective. The interest of this algorithm is not its computational complexity, but its formal simplicity. It is of high theoretical interest in the study of modularly distributed DES with partial observations of local modules. It may enable us in the future (Komenda and van Schuppen, 2004) to find the conditions under which the global supremal normal and/or supremal normal and controllable sublanguages can be synthesized locally.

ALGORITHM 4 *Let automata S_1 and S representing K and L , respectively be such that S_1 is a subautomaton of S and S_1 is a state-partition automaton. Let us construct partial automaton $\tilde{S} = (\tilde{S}, \langle \tilde{o}, \tilde{t} \rangle)$ with \tilde{t} denoted by \rightarrow .*

Define the auxiliary condition () as follows:*

if $a \in A_{uo}$ then $\forall u \in A_{uo}^ : s_a \xrightarrow{u} \Rightarrow s_a \xrightarrow{u}_1$;*

if $a \in A_o$ then $\forall s' \approx_{Aux(S_1)} s : s' \xrightarrow{a} \Rightarrow s' \xrightarrow{a}_1$, in which case also $\forall u \in A_{uo}^ : s'_a \xrightarrow{u} \Rightarrow s'_a \xrightarrow{u}_1$.*

Below are the steps of the algorithm.

1. Put $\tilde{S} := \{s_0\}$.
2. For any $s \in \tilde{S}$ and $a \in A$ we put $s \xrightarrow{a}$, s_a if $s \xrightarrow{a}_1$ and condition (*) is satisfied and we put in the case $s \xrightarrow{a}$, also $\tilde{S} := \tilde{S} \cup \{s_a\}$.
3. For any $s \in \tilde{S}$ we put $\tilde{o}(s) = o(s)$.

Let us denote by \tilde{l} the unique (behavior) homomorphism given by finality of \mathcal{L} .

THEOREM 10.1 $\tilde{I}(s_0)$ is the supremal (L, P) —normal sublanguage of K .

Proof: To prove the normality of $\tilde{I}(s_0)$ we show that the following relation is a normal relation on $\tilde{S} \times S$.

$$N = \left\{ \langle (s_0)_u, (s_0)_u \rangle \mid u \in \tilde{I}(s_0)^2 \right\}.$$

Then $\tilde{I}(s_0)$ is (L, P) —normal according to Theorem 6.7. Take a pair $\langle (s_0)_v, (s_0)_v \rangle \in N$ for some $v \in \tilde{I}(s_0)^2$.

- (i) If $(s_0)_v \xrightarrow{a}$, for $a \in A$, then clearly by construction of Algorithm 4 $(s_0)_v \xrightarrow{a}$. It is clear from the definition of N that $\langle (s_0)_{va}, (s_0)_{va} \rangle \in N$.
- (ii) Let $a \in A_{uo}$ be such that $(s_0)_v \xrightarrow{a}$. We must show that $(s_0)_v \xrightarrow{a}$, i.e., $\forall u \in A_{uo}^* : (s_0)_{va} \xrightarrow{u} \Rightarrow (s_0)_{va} \xrightarrow{u}_1$. It follows from $(s_0)_v \xrightarrow{a}$ and Algorithm 4 that $\forall u \in A_{uo}^* : (s_0)_v \xrightarrow{u} \Rightarrow (s_0)_v \xrightarrow{u}_1$. Indeed, if we assume $v = v_1 \dots v_k$ for some $k \in \mathbb{Z}$, then either $v_k \in A_{uo}$, i.e., $(s_0)_{v_1 \dots v_{k-1}} \xrightarrow{v_k}$, means directly that $\forall u \in A_{uo}^* : (s_0)_v \xrightarrow{u} \Rightarrow (s_0)_v \xrightarrow{u}_1$ or $v_k \in A_o$, but then the condition (*) is even stronger: by putting $s' = s$ we obtain the same conclusion. Since in both cases $au \in A_{uo}^*$, the required implication holds as well for $(s_0)_{va}$ as required for $(s_0)_v \xrightarrow{a}$.
- (iii) Let $a \in A_o$ be such that $(s_0)_v \xrightarrow{a}$ and let there exist $s' \in \tilde{S} : s' \approx_{Aux(\tilde{S})} (s_0)_v$ with $s' \xrightarrow{a}$. By Lemma 5.1 there exist two strings $w, w' \in A^*$ such that $P(w) = P(w')$, $(s_0)_v = (s_0)_w$, and $s' = (s_0)_{w'} \xrightarrow{a}$. According to the construction of Algorithm 4 for any $s \approx_{Aux(S_1)} (s_0)_w$ there must be $s \xrightarrow{a} \Rightarrow s \xrightarrow{a}_1$, in which case also $\forall u \in A_{uo}^* : s \xrightarrow{u} \Rightarrow s \xrightarrow{u}_1$. In order to show that $(s_0)_v \xrightarrow{a}$, it must be that for any $q \approx_{Aux(S_1)} (s_0)_v$ we have $q \xrightarrow{a} \Rightarrow q \xrightarrow{a}_1$, in which case also $\forall u \in A_{uo}^* : q \xrightarrow{u} \Rightarrow q \xrightarrow{u}_1$. But using the fact that $Aux(S_1)$ is transitive, because S_1 is a state-partition automaton, a stronger condition, and the fact that $s' \approx_{Aux(\tilde{S})} (s_0)_v$ implies that $s' \approx_{Aux(S_1)} (s_0)_v$ we obtain that $\langle s', q \rangle \in Aux(S_1)$. But this just means that for any $q \approx_{Aux(S_1)} (s_0)_v$ we have $q \xrightarrow{a} \Rightarrow q \xrightarrow{a}_1$, in which case also $\forall u \in A_{uo}^* : q \xrightarrow{u} \Rightarrow q \xrightarrow{u}_1$, i.e., $(s_0)_v \xrightarrow{a}$. Therefore N is a normal relation.

We show finally that the supremal (L, P) —normal sublanguage of K is contained in $\tilde{I}(s_0)$. Let N be a (L, P) —normal sublanguage of K . Then it is sufficient to show that

$$R = \left\{ \langle N_u, \tilde{I}(s_0)_u \rangle \mid u \in N^2 \right\}$$

satisfies (ii) of simulation relation in order to prove that $N^2 \subseteq \tilde{I}(s_0)^2$. Take an arbitrary pair $\langle N_w, \tilde{I}(s_0)_w \rangle \in R$ for some $w \in N^2$. Let $N_w \xrightarrow{a}$ for $a \in A$. Then also $K_w \xrightarrow{a}$, since $N \subseteq K$ and $L_w \xrightarrow{a}$ as well. This means that $(s_0)_w \xrightarrow{a}_1$ and $(s_0)_w \xrightarrow{a}$. In order to show that $\tilde{I}(s_0)_w \xrightarrow{a}$, i.e., $(s_0)_w \xrightarrow{a}$, it must be shown that the condition (*) is satisfied.

For $a \in A_{uo}$ we need to show that $\forall u \in A_{uo}^* : (s_0)_{wa} \xrightarrow{u} \Rightarrow (s_0)_{wa} \xrightarrow{u} 1$. But this is easy: $(s_0)_{wa} \xrightarrow{u}$ means $wau \in L^2$. Since N is (L, P) —normal, $wa \in N^2$ and $P(wa) = P(wau)$, we deduce $wau \in N^2 \subseteq K^2$. But this just means that $(s_0)_{wa} \xrightarrow{u} 1$.

For $a \in A_0$ it must be checked that for any $q \approx_{Aux(S_1)} (s_0)_w : q \xrightarrow{a} \Rightarrow q \xrightarrow{a} 1$, in which case also $\forall u \in A_{uo}^* : q_a \xrightarrow{u} \Rightarrow q_a \xrightarrow{u} 1$. There exist $v, v' : P(v) = P(v')$ such that $q = (s_0)_v$ and $(s_0)_w = (s_0)_{v'}$. Since S_1 is a state-partition automaton and $(s_0)_w = (s_0)_v$ is in two potentially different states of the observer automaton, we conclude by the property of state-partition automaton that these two states of the observer automaton coincide. But this means that there exists $w' \in A^*$ such that $P(w) = P(w')$ and $q = (s_0)_{w'}$. Now $q \xrightarrow{a}$ means that $w'a \in L^2$. By normality of N it follows from $wa \in N^2$ and $w'a \in L^2$ that $w'a \in N^2$. Therefore $w'a \in K^2$ (because $N \subseteq K$), which means that $q \xrightarrow{a} 1$. The rest is similar to the case $a \in A_{uo}$: if for $u \in A_{uo}^* : q_a = (s_0)_{w'a} \xrightarrow{u}$, then $w'au \in L^2$, by normality of N and using $wa \in N^2$, where $P(w'au) = P(wa)$ we have $w'au \in N^2 \subseteq K^2$. But this just means that $(s_0)_{w'a} = q_a \xrightarrow{u} 1$.

We conclude that $\tilde{I}(s_0)_w \xrightarrow{a}$ and R satisfies (ii) of simulation relation, i.e., we have the inclusion $N^2 \subseteq \tilde{I}(s_0)^2$. Note that since N was arbitrary (L, P) —normal sublanguage of K , and $\tilde{I}(s_0)$ has been shown to be a (L, P) —normal sublanguage of K , it follows that $\tilde{I}(s_0)$ is the supremal (L, P) —normal sublanguage of K .

Following the same technique we can synthesize a ‘single-step’ algorithm for computation of supremal normal and controllable sublanguages.

ALGORITHM 5 *Let automata S_1 and S representing K and L , respectively are such that S_1 is a subautomaton of S and S_1 is a state-partition automaton. Let us construct partial automaton $\tilde{S} = (\tilde{S}, \langle \tilde{o}, \tilde{r} \rangle)$ with \tilde{r} denoted by \rightarrow .*

*Define the auxiliary condition (***) as follows:*

if $a \in A_u \cup A_{uo}$ then $\forall u \in (A_u \cup A_{uo})^ : s_a \xrightarrow{u} \Rightarrow s_a \xrightarrow{u} 1$;*

if $a \in A_c \cap A_0$ then $\forall s' \approx_{Aux(S_1)} s : s' \xrightarrow{a} \Rightarrow s' \xrightarrow{a} 1$, in which case also $\forall u \in (A_u \cup A_{uo})^ : s'_a \xrightarrow{u} \Rightarrow s'_a \xrightarrow{u} 1$.*

Below are the steps of the algorithm.

1. Put $\tilde{S} := \{s_0\}$.
2. For any $s \in \tilde{S}$ and $a \in A$ we put $s \xrightarrow{a} s_a$ if $s \xrightarrow{a} 1$ and condition (***) is satisfied and we put in the case $s \xrightarrow{a}$, also $\tilde{S} := \tilde{S} \cup \{s_a\}$.
3. For any $s \in \tilde{S}$ we put $\tilde{o}(s) = o(s)$.

As usual, we denote by \tilde{I} the unique (behavior) homomorphism given by finality of \mathcal{L} . Similarly as for Algorithm 4, one can verify by coinduction that

THEOREM 10.2 $\tilde{I}(s_0)$ is the supremal controllable (with respect to L and A_u) and (L, P) —normal sublanguage of K .

Proof: The structure of the proof follows very much that of Theorem 1. First we prove that $\tilde{I}(s_0)$ is controllable with respect to L and A_u . According to Theorem 6.8 it is sufficient to show that the following relation is a control relation on $\tilde{S} \times S$.

$$C = \left\{ \langle (s_0)_u, (s_0)_u \rangle \mid u \in \tilde{I}(s_0)^2 \right\}.$$

Take a pair $\langle (s_0)_w, (s_0)_w \rangle \in N$ for some $w \in \tilde{I}(s_0)^2$.

- (i) If $(s_0)_w \xrightarrow{a}$, for $a \in A$, then clearly by construction of Algorithm 5 $(s_0)_w \xrightarrow{a}$, because $\tilde{I}(s_0) \subseteq K \subseteq L$. It is clear from the definition of C that $\langle (s_0)_{wa}, (s_0)_{wa} \rangle \in C$.
- (ii) Let $a \in A_u$ be such that $(s_0)_w \xrightarrow{a}$. We must show that $(s_0)_w \xrightarrow{a}$. According to Algorithm 5 condition (***) must be checked. Since $A_u \subseteq A_u \cup A_{uo}$ it amounts to show that $\forall u \in (A_u \cup A_{uo})^* : (s_0)_{wa} \xrightarrow{u} \Rightarrow (s_0)_{wa} \xrightarrow{u} 1$. We have $u = u_1 \dots u_l$ for some $l \in \mathbb{Z}$ with $\forall i \in \{1, \dots, l\} : u_i \in A_u \cup A_{uo}$. Notice that we have also $w = w_1 \dots w_k$ for some $k \in \mathbb{Z}$. There are 2 possibilities for w_k : $w_k \in A_u \cup A_{uo}$ or $w_k \in A_c \cap A_o$. According to condition (***) of Algorithm 5 for $(s_0)_w \xrightarrow{a}$, i.e., $(s_0)_{w_1 \dots w_{k-1}} \xrightarrow{w_k}$ in both cases means that in particular $\forall u \in (A_u \cup A_{uo})^* : s_w \xrightarrow{u} \Rightarrow s_w \xrightarrow{u} 1$. Indeed, condition (***) for $w_k \in A_c \cap A_o$ is stronger than for $w_k \in A_u \cup A_{uo}$ as is easily seen by taking $s' = s$. Since $au \in (A_u \cup A_{uo})^*$, the condition (***) for $(s_0)_w \xrightarrow{a}$ holds true, which proves the controllability of $\tilde{I}(s_0)$.

To prove the normality of $\tilde{I}(s_0)$ we show that the following relation is a normal relation on $\tilde{S} \times S$.

$$N = \left\{ \langle (s_0)_u, (s_0)_u \rangle \mid u \in \tilde{I}(s_0)^2 \right\}.$$

Then $\tilde{I}(s_0)$ is normal with respect to L and P according to Theorem 6.7. Take a pair $\langle (s_0)_v, (s_0)_v \rangle \in N$ for some $v \in \tilde{I}(s_0)^2$.

- (i) If $(s_0)_v \xrightarrow{a}$, for $a \in A$, then clearly by construction of Algorithm 5 $(s_0)_v \xrightarrow{a}$. It is clear from the definition of N that $\langle (s_0)_{va}, (s_0)_{va} \rangle \in N$.
- (ii) Let $a \in A_{uo}$ be such that $(s_0)_v \xrightarrow{a}$. We must show that $(s_0)_v \xrightarrow{a}$, i.e., $\forall u \in (A_u \cup A_{uo})^* : (s_0)_{va} \xrightarrow{u} \Rightarrow (s_0)_{va} \xrightarrow{u} 1$. It follows from $(s_0)_v \xrightarrow{a}$, $A_{uo} \subseteq A_u \cup A_{uo}$, and Algorithm 5 that $\forall u \in (A_u \cup A_{uo})^* : (s_0)_v \xrightarrow{u} \Rightarrow (s_0)_v \xrightarrow{u} 1$, the argument being the same as above in the proof of controllability. Since $au \in (A_u \cup A_{uo})^*$, the required implication holds as well.
- (iii) Let $a \in A_o$ be such that $(s_0)_v \xrightarrow{a}$ and let there exists $s' \in \tilde{S} : s' \approx_{\text{Aux}(\tilde{S})} (s_0)_v$ with $s' \xrightarrow{a}$. By Lemma 5.1 there exist two strings $w, w' \in A^*$ such that $P(w) = P(w')$, $(s_0)_v = (s_0)_w$, and $s' = (s_0)_{w'} \xrightarrow{a}$. But using the fact that S_1 is a state-partition automaton there exists $v' : P(v') = P(v)$ such that $s' = (s_0)_{v'} \xrightarrow{a}$. Two cases must be distin-

guished. Assume first that $a \in A_o \cap A_c$. It follows from Algorithm 5 that $\forall s \approx_{Aux(S_1)} (s_0)_v$, there must be $s \xrightarrow{a} \Rightarrow s \xrightarrow{a}_1$, in which case also $\forall u \in (A_u \cup A_{uo})^* : s_a \xrightarrow{u} \Rightarrow s_a \xrightarrow{u}_1$. Then $s \xrightarrow{a}$, as well using the transitivity of $Aux(S_1)$ and the obvious fact that $Aux(\tilde{S}) \subseteq Aux(S_1)$, which means that $s' \approx_{Aux(\tilde{S})} (s_0)_v$ implies that $s' \approx_{Aux(S_1)} (s_0)_v$. Now for any $q \approx_{Aux(S_1)} (s_0)_v$ there must be $\langle s', q \rangle \in Aux(S_1)$. Therefore $q \xrightarrow{a} \Rightarrow q \xrightarrow{a}_1$, in which case also $\forall u \in (A_u \cup A_{uo})^* : q_a \xrightarrow{u} \Rightarrow q_a \xrightarrow{u}_1$. Thus $(s_0)_v \xrightarrow{a}$. Now we assume that $a \in A_o \cap A_u$. According to the construction of Algorithm 5, it is sufficient to show that $\forall u \in (A_u \cup A_{uo})^* : (s_0)_{va} \xrightarrow{u} \Rightarrow (s_0)_{va} \xrightarrow{u}_1$. We know that $(s_0)_v \xrightarrow{v}$. Using the same argument as in the proof of controllability or (ii) of normality it follows that $\forall u \in (A_u \cup A_{uo})^* : (s_0)_v \xrightarrow{u} \Rightarrow (s_0)_v \xrightarrow{u}_1$. It is sufficient to notice that $a \in A_o \cap A_u \subseteq A_u \cup A_{uo}$, i.e., also $au \in (A_u \cup A_{uo})^*$. Thus, $\forall u \in (A_u \cup A_{uo})^* : (s_0)_{va} \xrightarrow{u} \Rightarrow (s_0)_v \xrightarrow{au} \Rightarrow (s_0)_v \xrightarrow{u}_1$, which is equivalent to $(s_0)_{va} \xrightarrow{u}_1$. Since in both cases $(s_0)_v \xrightarrow{a}$, we conclude that N is a normal relation.

We show finally that the supremal controllable (with respect to L and A_u) and (L, P) —normal sublanguage of K is contained in $\tilde{l}(s_0)$. Let N be a controllable and (L, P) —normal sublanguage of K . Then it is sufficient to show that

$$R = \{ \langle N_u, \tilde{l}(s_0)_u \rangle \mid u \in N^2 \}$$

satisfies (ii) of simulation relation in order to prove that $N^2 \subseteq \tilde{l}(s_0)^2$. Take an arbitrary pair $\langle N_w, \tilde{l}(s_0)_w \rangle \in R$ for some $w \in N^2$. Let $N_w \xrightarrow{a}$ for $a \in A$. Then also $K_w \xrightarrow{a}$, since $N \subseteq K$ and $L_w \xrightarrow{a}$ as well. This means that $(s_0)_w \xrightarrow{a}_1$ and $(s_0)_w \xrightarrow{a}$. In order to show that $\tilde{l}(s_0)_w \xrightarrow{a}$, i.e., $(s_0)_w \xrightarrow{a}$, it must be shown that condition (***) of Algorithm 5 is satisfied. If $a \in A_u \cup A_{uo}$ then we show that $\forall u \in (A_u \cup A_{uo})^* : (s_0)_{wa} \xrightarrow{u} \Rightarrow (s_0)_{wa} \xrightarrow{u}_1$. Indeed, if $u \in (A_u \cup A_{uo})^*$ then $u = u_1 \dots u_k$ for some $k \in \mathbb{Z}$ with $u_i \in A_u \cup A_{uo} \forall i \in \{1, \dots, k\}$. Thus, $(s_0)_{wa} \xrightarrow{u}$, i.e., $wau = wau_1 \dots u_k \in L^2$ together with $wa \in N^2$ and normality and controllability of N inductively used implies $wau_1 \in N^2, \dots, wau = wau_1 \dots u_k \in N^2 \subseteq K^2$. This means that $(s_0)_{wa} \xrightarrow{u}_1$, which was to be shown. Let $a \in A_c \cap A_o$. We need to show that $\forall s' \approx_{Aux(S_1)} (s_0)_w : s' \xrightarrow{a} \Rightarrow s' \xrightarrow{a}_1$, in which case also $\forall u \in (A_u \cup A_{uo})^* : s'_a \xrightarrow{u} \Rightarrow s'_a \xrightarrow{u}_1$. Let $s' \approx_{Aux(S_1)} (s_0)_w : s' \xrightarrow{a}$. According to Lemma 5.1 and by taking into account that S_1 is a state-partition automaton, there exists $w' \in K^2$ such that $P(w') = P(w)$ and $s' = (s_0)_{w'}$. Hence $s' \xrightarrow{a}$ is equivalent to $w'a \in L^2$. By normality of N it follows from $wa \in N^2$ and $w'a \in L^2$ that $w'a \in N^2$. Thus $w'a \in K^2$, because $N^2 \subseteq K^2$, but this means that $s'_a \xrightarrow{u}_1$. The second part is similar as for $a \in A_{uo} \cup A_u$. Indeed, for $u \in (A_u \cup A_{uo})^*$ with $s'_a \xrightarrow{u}$ we obtain consequently: $w'au \in L^2$, $w'a \in N^2$, i.e., by inductive application of normality and controllability of N we have finally $w'au \in N^2 \subseteq K^2$, which gives $s'_a \xrightarrow{u}_1$. To conclude, in any case we have obtained $\tilde{l}(s_0)_w \xrightarrow{a}$, i.e., R satisfies (ii) of simulation relation, and the inclusion $N^2 \subseteq \tilde{l}(s_0)^2$ has been shown. Note that since N was arbitrary controllable and (L, P) —normal sublanguage of K , it follows that $\tilde{l}(s_0)$ is the supremal controllable and (L, P) —normal sublanguage of K . ■

Remark 10.3: An important feature of Algorithm 5 is its compactness, i.e., it is not an iteration of two separate algorithms as are the algorithms in Yoo et al. (2001) or

Komenda (2002b). Therefore it looks almost like a coinductive definition of the supremal normal and controllable sublanguage, which is not possible to do directly in \mathcal{L} . Thus Algorithm 5 is suitable for investigating problems like “when does the supremal normal and controllable sublanguage commute with the synchronous product of (partial) languages?”

11. Distributivity of the Supervised Product

The behavior of the supervised DES has been formalized by the (partial) language operation of supervised product. It is of interest to study algebraic properties of this operation, e.g., distributivity with respect to (partial) language operations. The problem of distributivity of the supervised product with respect to language unions is addressed in this section. The following theorem answers the main question. It turns out that

THEOREM 11.1 *If $A_c \subseteq A_o$, then for any K and K' (partial) sublanguages of L we have:*

$$(K \cup K') / \overset{o}{U} L = (K / \overset{o}{U} L) \cup (K' / \overset{o}{U} L).$$

Proof: Formally, it can be checked that

$$R = \left\{ \left\langle [(K \cup K') / \overset{o}{U} L]_u, [(K / \overset{o}{U} L) \cup (K' / \overset{o}{U} L)]_u \right\rangle \mid u \in [(K \cup K') / \overset{o}{U} L]^2 \right\}$$

is a bisimulation relation. Take a $w \in [(K \cup K') / \overset{o}{U} L]^2$.

- (i) is straightforward: $[(K \cup K') / \overset{o}{U} L]_w \downarrow$ iff $w \in L^1$ iff $[(K / \overset{o}{U} L) \cup (K' / \overset{o}{U} L)]_w \downarrow$.
- (ii) If $[(K \cup K') / \overset{o}{U} L]_w \xrightarrow{a}$ for $a \in A$, then according to the Definition 8.4 of supervised products several cases must be distinguished. We have the following possibilities:

$$[(K \cup K') / \overset{o}{U} L]_w = \begin{cases} (K \cup K')_w / \overset{o}{U} L_w & \text{if } (K \cup K') \xrightarrow{w} \text{ and } L \xrightarrow{w} \\ \cup_{i \in I} (K \cup K')_{w_i} / \overset{o}{U} L_w & \text{if } (K \cup K') \xrightarrow{w} \text{ and } L \xrightarrow{w} \text{ and } \exists I \neq \emptyset \\ & \text{and } w_i, i \in I : P(w_i) = P(w) \text{ and } \forall i \in I : \\ & (K \cup K') \xrightarrow{w_i} \\ 0 / \overset{o}{U} L_w & \text{if } K \xrightarrow{w} \text{ and } L \xrightarrow{w} \text{ and } w \text{ cancel } \notin A_c^* \end{cases}$$

By applying again the definition of the supervised product several cases must be distinguished:

$$[(K \cup K') /_{\mathcal{U}}^{\mathcal{O}} L]_{wa} = \begin{cases} (K \cup K')_{wa} /_{\mathcal{U}}^{\mathcal{O}} L_{wa} & \text{if } (K \cup K') \xrightarrow{wa} \text{ and } L \xrightarrow{wa} \\ \bigcup_{j \in J} (K \cup K')_{v_j} /_{\mathcal{U}}^{\mathcal{O}} L_{wa} & \text{if } (K \cup K') \xrightarrow{w} \text{ and } L \xrightarrow{wa} \text{ and } a \in A_c \cup A_o \\ & \text{and } \exists J \neq \emptyset \text{ and } v_j, j \in J : P(v_j) = P(w) \\ & \text{and } \forall j \in J : (K \cup K')_{v_j} \xrightarrow{a} \\ (K \cup K')_w /_{\mathcal{U}}^{\mathcal{O}} L_{wa} & \text{if } K \xrightarrow{wa} \text{ and } L \xrightarrow{wa} \text{ and } a \in A_{uc} \cap A_{uo} \\ 0 /_{\mathcal{U}}^{\mathcal{O}} L_{wa} & \text{if } K \xrightarrow{wa} \text{ and } L \xrightarrow{wa} \text{ and } \forall v : P(v) = P(w) : \\ & K_v \xrightarrow{wa} \text{ and } a \in A_{uc} \cap A_o \end{cases}$$

Now combinations of different cases of both preceding equations must be considered. Some of the combinations are only hypothetic, and in fact they are impossible. For instance, if the last case in the first equation occurs, then only the last case in the second equation can occur. Some cases are easy, others are problematic. For instance, the last case of the other equation in combination with any case of the first equation, as well as the combination of the first cases of both equations are not problematic and the conclusion is easily drawn. Now we consider the problematic case $(K \cup K') \xrightarrow{w}$, namely e.g., $K \xrightarrow{w}$ and $K' \xrightarrow{w}$, while there exists an index set J such that $\forall j \in J : (K \cup K')_{v_j} \xrightarrow{a}$ and $P(v_j) = P(w)$, namely e.g., $\forall j \in J : K_{v_j} \xrightarrow{a}$ and $K'_{v_j} \xrightarrow{a}$. This is a problem, because in order to draw the plausible conclusion $(K' /_{\mathcal{U}}^{\mathcal{O}} L)_w \xrightarrow{a}$, we need first be sure that $(K' /_{\mathcal{U}}^{\mathcal{O}} L) \xrightarrow{w}$, which is not obvious. However our assumption $A_c \subseteq A_o$ will be used. It is known from Corollary 22 that under the assumption $A_c \subseteq A_o$ observability together with controllability are equivalent to the normality. Since the supervised product is known to be controllable and observable, it follows that the supervised product is also (L, P) —normal. Therefore $(K' /_{\mathcal{U}}^{\mathcal{O}} L)$ is (L, P) —normal and from $K'_{v_j} \xrightarrow{a}$, i.e., $(K' /_{\mathcal{U}}^{\mathcal{O}} L)_{v_j} \xrightarrow{a}$ it follows that $(K' /_{\mathcal{U}}^{\mathcal{O}} L) \xrightarrow{w}$, and thus $(K \cup K' /_{\mathcal{U}}^{\mathcal{O}} L)_w \xrightarrow{a}$.

The same problem appears if the second case in the first equation occurs. The situation is similar to the one above with w replaced by w_i , but owing to the (L, P) —normality of the supervised product this is not substantial: again $(K' /_{\mathcal{U}}^{\mathcal{O}} L)_{v_j} \xrightarrow{a}$ implies that $(K' /_{\mathcal{U}}^{\mathcal{O}} L) \xrightarrow{w}$.

- (iii) This inclusion (simulation) is easy and holds always: it follows from the monotonicity of the supervised product with respect to the specification (see Corollary 7.13), therefore $K \subseteq K \cup K'$ implies that $K /_{\mathcal{U}}^{\mathcal{O}} L \subseteq (K \cup K') /_{\mathcal{U}}^{\mathcal{O}} L$. Similarly, $K' \subseteq K \cup K'$ implies that $K' /_{\mathcal{U}}^{\mathcal{O}} L \subseteq (K \cup K') /_{\mathcal{U}}^{\mathcal{O}} L$. Hence, $(K /_{\mathcal{U}}^{\mathcal{O}} L) \cup (K' /_{\mathcal{U}}^{\mathcal{O}} L) \subseteq (K \cup K') /_{\mathcal{U}}^{\mathcal{O}} L$. ■

Under the structural assumption on the event set $A_c \subseteq A_o$, the supervised product with partial observations distributes with language unions. This distributivity implies that important properties are preserved by unions: if $K /_{\mathcal{U}}^{\mathcal{O}} L = K$, i.e., K is controllable, $L_m(G)$ —closed and observable and $K' /_{\mathcal{U}}^{\mathcal{O}} L = K$, i.e., K' is controllable, $L_m(G)$ —closed and observable, then $(K \cup K') /_{\mathcal{U}}^{\mathcal{O}} L = (K /_{\mathcal{U}}^{\mathcal{O}} L) \cup (K' /_{\mathcal{U}}^{\mathcal{O}} L) = K \cup K'$, i.e., $K \cup K'$ is also controllable, $L_m(G)$ —closed and observable. Note finally that it is not difficult to extend

the above distributivity to an arbitrary number of specifications, even to an infinite number (which amounts to the lattice theoretical lower semicontinuity).

Similarly, if $A_c \subseteq A_o$ then our technique can be used to show that the supervised product is also distributive with respect to partial language intersections. The situation is symmetric in the sense that the opposite inclusion is trivial here (always holds). To conclude, we have shown that the concept of supervised product is useful for investigation of properties of closed-loop languages in discrete-event control with partial observations.

12. Conclusion

Supervisory control of DES with partial observations has been treated by coalgebraic techniques. The new concept of deterministic weak transitions gives rise to the definition of projected and observer automata. Observability and normality have been characterized by appropriate relations in this framework, which gives an insight into problems of partially observed DES. They have been used to design algorithms for supremal normal and/or normal and controllable sublanguages. These are discussed in detail and compared to those encountered in the literature.

Another approach, based on finality of the automaton of partial languages, consists in using coinductive or similar definitions for describing permissive or antipermissive control laws under partial observations. As a byproduct coinductive definitions of observable approximations of a given language have been obtained. These definitions give rise to new algorithms for the computation of infimal closed and observable superlanguages and observable sublanguages larger than the supremal normal sublanguage because of their coinductive nature. They rely only on observational indistinguishability relations, which can be constructed directly from the corresponding definitions that give at the same time algorithms for their construction. The lack of the existence of an optimal (maximally permissive) solution for the supervisory control with partial observations is related to the fact that the supervised product does not in general distribute with (partial) language unions when the controller has only partial information about the DES.

The naturally algorithmic character of the coalgebraic approach is one of its main advantages. While the algebraic approach works with strings (words), which is sometimes cumbersome, the coalgebraic approach relies on the relational framework (various weakening of bisimulation relations) and we proceed event by event. The use of coinductive definitions and proofs makes coalgebraic techniques relevant for control of DES. Coinductive definitions enable to characterize languages of the closed (controlled) DES and coinductive proofs are used to check different properties like controllability, observability, normality, or distributivity of operations on (partial) languages.

The results of this paper are being generalized to the decentralized and modular supervisory control. For instance, in modular control of DES, the system is composed of local subsystems that run concurrently, i.e., the global system is the parallel composition of local systems. To each local system a local supervisor is associated. Many interesting questions arise: can the control be exerted at the local level without violating our control objectives or without affecting the optimality of the solution? If the answers to these questions are positive, there is an exponential saving on the computational complexity. In

our coalgebraic framework these two problems can be paraphrased as follows: when does the supervised product commute with synchronous product and when does the supremal controllable sublanguage (as an operation defined by coinduction in Section 7) commute with synchronous product? [recall that the synchronous product of partial languages has been defined by coinduction in Rutten (1999)].

The contribution of the coalgebraic approach to the control and systems theory remains to be further evaluated. However, we believe that application of the coinductive techniques is not limited to discrete-event systems, but it can be useful for other type of systems. It seems possible to study with coalgebraic techniques some problems of hybrid systems, especially if the control objectives are only at the discrete-event level (safety or minimal required behavior). In some areas of control and systems theory with a high level of abstraction there might be interest to apply the coalgebraic techniques. Various coalgebras (systems) can be obtained by varying the functor on the category of sets. Moreover, the use of this method is not limited to systems defined by functors in the category of sets, but functors on some “structurally richer” categories (like the categories of topological or metric spaces and continuous functions between them as morphisms). An interesting application of coalgebra to symbolic dynamics in one dimensional discrete-time dynamical systems defined by a continuous function on a complete metric space can be found in Rutten (2000). Different types of coalgebras have their own notions of homomorphism and bisimulation, as well as cofreeness and finality, i.e., coinduction, yet there is a unifying theory of universal coalgebra.

Future research tasks might include a study of how to improve the computational complexity of our algorithms, a study of decentralized, hierarchical, and modular control of DES using coalgebra as well as an application of the coalgebraic techniques to timed DES. Optimal supervisory control can be investigated by coalgebraic techniques using coalgebras of weighted automata (Rutten, 2003) (weights here correspond to the costs) with formal power series as their final coalgebra.

Acknowledgments

The investigation was primarily carried out during the first author’s stay at CWI Amsterdam, where he has worked on the NWO project COCON together with his research advisor Jan H. van Schuppen. This research was partially supported by the Academy of Sciences of the Czech Republic, Institutional Research Plan No. AV0Z10190503. The discussion with Stéphane Lafortune and his Ph.D. students is gratefully acknowledged and has motivated a part of the results of Section 7.

References

- Aczel, P. Non-Well-Founded Sets. CSLI Lecture Notes, Number 14, Stanford University, 1988.
- Aczel, P., and Mendler, N. 1989. A final coalgebra theorem. In D. H. Pitt, D. E. Ryehard, P. Dybjer, A. M. Pitts, and A. Poigne (eds.), *Proc. Category Theory and Computer Science, Lecture Notes in Computer Science*, Volume 389, pp. 357–365.

- Barrett, G., and Lafortune, S. 1998. Bisimulation, the supervisory control problem and strong model matching for finite state machines. *J. Discret. Event Dyn. Syst.: Theory Appl.* 8(4): 377–429.
- Bergeron, A. 1993. A unified approach to control problems in discrete event processes. *Inform. Théor. Appl.* 27(6) : 555–573.
- Brandt, R. D., Garg, V., Kumar, R., Lin, F., Marcus, S. I., and Wonham, W. M. 1990. Formulas for calculating supremal controllable and normal sublanguages. *Syst. Control Lett.* 15: 111–117.
- Cassandras, S. G., and Lafortune, S. 1999. *Introduction to Discrete Event Systems*. Dordrecht: Kluwer Academic Publishers.
- Cieslak, R., Desclaux, C., Fawaz, A., and Varayia, P. 1988. Supervisory control of a class of discrete event processes. *IEEE Trans. Automat. Contr.* 33: 249–260.
- Cho, H., and Marcus, S. I. 1989a. On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observations. *Math. Control Signal Syst.* 2: 47–69.
- Cho, H., and Marcus, S. I. 1989b. Supremal and maximal sublanguages arising in supervisor synthesis problems with partial observations. *Math. Syst. Theory* 22: 171–211.
- Eilenberg, S. 1974. *Automata, Languages, and Machines*. New York: Academic Press.
- Eilenberg, S., and Moore, J. C. 1965. Adjoint functors and triples. *Ill. J. Math.* 9: 381–398.
- Grossman, R., and Larson, R. G. 1992. The realization map of input–output maps using bialgebras. *Forum Math.* 4: 109–121.
- Gumm, H. P. 2003. State based systems are coalgebras. *Cubo—Matemática Educacional* 5(2): 239–262.
- Hopcroft, J. E., and Ullman, J. D. 1979. *Introduction to Automata Theory, Languages and Computation*. Reading, MA: Addison-Wesley.
- Jacobson, N. 1980. *Basic Algebra*, Volume 2. New York: W. H. Freeman and Company.
- Komenda, J. 2002a. Computation of supremal sublanguages of supervisory control using coalgebra. In *Proceedings WODES'02, Workshop on Discrete-Event Systems*, Zaragoza, October 2–4, pp. 26–33.
- Komenda, J. 2002b. Coalgebra and supervisory control of discrete-event systems with partial observations. In *Proceedings of MTNS 2002*, Notre Dame (IN), August.
- Komenda, J., and van Schuppen, J. H. 2003. Decentralized supervisory control with coalgebra. In *Proceedings European Control Conference, ECC'03*, Cambridge, September 1–4, 2003, only CD-ROM. Also appeared as *Research Report CWI*, MAS-E0310, ISSN 1386-3703, Amsterdam.
- Komenda, J., and van Schuppen, J. H. 2004. Supremal normal sublanguages of large distributed discrete-event systems. In *Proceedings WODES'04, Workshop on Discrete-Event Systems*, Reims, September 22–24.
- Lafortune, S., and Chen, E. 1990. The infimal closed and controllable superlanguage and its applications in supervisory control. *IEEE Trans. Automat. Contr.* 35(4): 398–405.
- Lin, F., and Wonham, W. M. 1988. On observability of discrete-event systems. *Inf. Sci.* 44: 173–198.
- Lin, F., and Wonham, W. M. 1990. Decentralized control and coordination of discrete-event systems with partial observations. *IEEE Trans. Automat. Contr.* 35: 1330–1337.
- Milner, R. 1989a. A complete axiomatisation for observational congruence of finite-state behaviors. *Inf. Comput.* 81: 227–247.
- Milner, R. 1989b. *Communication and Concurrency*. Prentice Hall International Series in Computer Science. New York: Prentice Hall International.
- Overkamp, A., and van Schuppen, J. H. 2000. Maximal solutions in decentralized supervisory control. *SIAM J. Control Optim.* 39(2): 492–511.
- Park, D.M.R. *Concurrency and Automata on Infinite Sequences*, volume 104 of LNCS. Springer, 1980.
- Ramadge, P. J., and Wonham, W. M. 1989. The control of discrete-event systems. *Proc. IEEE* 77: 81–98.
- Rudie, K., and Wonham, W. M. 1990. The infimal prefix-closed and observable superlanguage of a given language. *Syst. Control Lett.* 15: 361–371.
- Rudie, K., and Wonham, W. M. 1992. Think globally, act locally: Decentralized supervisory control. *IEEE Trans. Automat. Contr.* 37(11): 1692–1708.
- Rutten, J. J. M. M. 1998. Automata and Coinduction (An Exercise in Coalgebra). *Research Report CWI*, SEN-R9803, Amsterdam, May. Available also at <http://www.cwi.nl/~janr>.
- Rutten, J. J. M. M. 1999. Coalgebra, Concurrency, and Control. *Research Report CWI*, SEN-R9921, Amsterdam, November. Available also at <http://www.cwi.nl/~janr>.
- Rutten, J. J. M. M. 2000. Universal coalgebra: A theory of systems. *Theor. Comp. Sci.* 249(1): 3–80.

- Rutten, J. J. M. M. 2003. Fundamental study. Behavioural differential equations: A coinductive calculus of streams, automata, and power series. *Theor. Comp. Sci.* 308(1): 1–53.
- Sipser, M. 1997. *Introduction to the Theory of Computation*. Boston: PWS Publishing Company.
- Takai, S., and Ushio, T. 2002. Effective computation of an $L_m(G)$ -closed, controllable, and observable sublanguage arising in supervisory control. In *Proceedings WODES'02, Workshop on Discrete-Event Systems*, Zaragoza, October 2–4, pp. 34–39.
- Tarski, A. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pac. J. Math.* 5: 285–309.
- Thistle, J. G. 1996. Supervisory control of discrete event systems. *Math. Comput. Model.* 11/12: 25–53.
- Thistle, J. G., and Wonham, W. M. 1994. Supervision of infinite behavior of discrete-event systems. *SIAM J. Control Optim.* 32(4): 1098–1113.
- Tsitsiklis, J. N. 1989. On the control of discrete-event dynamical systems. *Math. Control Signals Syst.* 95–107.
- Wonham, W. M. 1976. Towards an abstract internal model principle. *IEEE Trans. Syst. Man Cybern.* 6(11): 735–740.
- Wonham, W. M., and Ramadge, P. J. 1987. On the supremal controllable sublanguage of a given language. *SIAM J. Control Optim.* 25: 637–659.
- Yoo, T. S., and Lafortune, S. 2002. General architecture for decentralized supervisory control of discrete-event systems. *Discret. Event Dyn. Syst.: Theory Appl.* 12: 335–377.
- Yoo, T. S., Lafortune, S., and Lin, F. 2001. A uniform approach for computing supremal sublanguages arising in supervisory control theory. *Preprint, Dept. of Electrical Engineering and Computer Science, University of Michigan*, Ann Arbor.