2014 11th IEEE International
Conference on Control & Automation (ICCA)
June 18-20, 2014. Taichung, Taiwan

Supervisory Control with Progressive Events

Simon Ware Robi Malik

Abstract— **This paper investigates some limitations of the *nonblocking* property when used for supervisor *synthesis* in discrete event systems. It is shown that there are cases where synthesis with the nonblocking property gives undesired results. To address such cases, the paper introduces *progressive events* as a means to specify more precisely how a synthesised supervisor should complete its tasks. The nonblocking property is modified to take progressive events into account, and appropriate methods for verification and synthesis are proposed.**

I. Introduction

In *supervisory control theory* [1], it is common to use the *nonblocking* property to ensure liveness when automatically synthesising supervisors. A discrete event system is nonblocking if, from every reachable state, all involved components can cooperatively complete their common tasks. It is not required that task completion is achieved on every possible execution path, only that there exists an execution path to a terminal state. For finite-state systems, the nonblocking property is equivalent to termination under the *fairness assumption* that events that are enabled infinitely often will be taken eventually [2]. This weak liveness condition ensures the existence of least restrictive synthesis results and has been used successfully in many applications [1], [3].

On the other hand, the nonblocking property is weaker than a guarantee of termination, and it is not always expressive enough to give the intended results. Several alternatives and extensions to the standard nonblocking property have been proposed. *Multi-tasking supervisory control* [4] allows the specification of multiple nonblocking requirements that must be satisfied simultaneously. The *generalised nonblocking* property [5] restricts the situations in which nonblocking is required, which is useful in hierarchical interface-based supervisory control [6]. *Nonblocking under control* [7] changes the fairness assumption of standard nonblocking by making the assumption that controllable events can preempt uncontrollable events when completing tasks, facilitating reasoning about supervisor implementations. Finally, the authors of [8] replace the nonblocking property by the requirement of true termination and perform synthesis using ω-languages.

This paper proposes a different generalisation of the nonblocking property. It introduces *progressive events*, which are the only events that can be used in traces towards task completion when checking the nonblocking property. Sect. II introduces the definitions for discrete event systems and supervisory control theory. Sect. III shows two examples of discrete event systems, for which the standard nonblocking property fails to give a useful synthesis result. Sect. IV introduces progressive events to model these examples more appropriately, and shows how nonblocking verification and synthesis are adapted for progressive events. Afterwards, Sect. V compares nonblocking with progressive events to the other nonblocking properties mentioned above, and Sect. VI adds some concluding remarks.

II. Preliminaries

A. Events and Languages

The behaviour of discrete event systems is modelled using events and languages [1]. *Events* represent incidents that cause transitions from one state to another and are taken from a finite alphabet Σ. For the purpose of supervisory control, this alphabet is partitioned into the set $\Sigma_{\rm c}$ of *controllable* events and the set $\Sigma_{\rm uc}$ of *uncontrollable* events. Controllable events can be disabled by a supervising agent, while uncontrollable events cannot be disabled. Independently of this distinction, the alphabet Σ is also partitioned into the set $\Sigma_{\rm o}$ of *observable* events and the set $\Sigma_{\rm uo}$ of *unobservable* events. Observable events are visible to the supervising agent, while unobservable events are not. In this paper, it is assumed that all unobservable events are also uncontrollable.

Given an event alphabet Σ, the term Σ^* denotes the set of all finite *traces* of the form $\sigma_1\sigma_2\cdots\sigma_n$ of events from Σ, including the *empty trace* ε. The *concatenation* of two traces $s,t \in \Sigma^*$ is written as st. A subset $\mathcal{L} \subseteq \Sigma^*$ is called a *language*. A trace $s \in \Sigma^*$ is a *prefix* of $t \in \Sigma^*$, written $s \sqsubseteq t$, if $t = su$ for some $u \in \Sigma^*$. The *prefix-closure* of a language $L \subseteq \Sigma^*$ is $\overline{L} = \{\, s \in \Sigma^* \mid s \sqsubseteq t \text{ for some } t \in L \,\}$, and L is *prefix-closed* if $\overline{L} = L$.

For $\Omega \subseteq \Sigma$, the *natural projection* $P_{\Sigma\to\Omega}\colon \Sigma^* \to \Omega^*$ is the operation that removes from traces $s \in \Sigma^*$ all events not in Ω. Its inverse image $P^{-1}_{\Sigma\leftarrow\Omega}\colon \Omega^* \to 2^{\Sigma^*}$ is defined by $P^{-1}_{\Sigma\leftarrow\Omega}(t) = \{\, s \in \Sigma^* \mid P_{\Sigma\to\Omega}(s) = t \,\}$. If the source alphabet is clear from the context, these functions are also written as $P_\Omega = P_{\Sigma\to\Omega}$ and $P^{-1}_\Sigma = P^{-1}_{\Sigma\leftarrow\Omega}$.

The *synchronous composition* of two languages $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ is $L_1 \parallel L_2 = P^{-1}_{\Sigma_1\cup\Sigma_2}(L_1) \cap P^{-1}_{\Sigma_1\cup\Sigma_2}(L_2)$.

B. Discrete Event Systems

In this paper, discrete event systems are modelled as pairs of languages or as finite-state automata.

Definition 1: Let Σ be a finite set of events. A *discrete event system* over Σ (*Σ-DES*) is a pair $\mathbf{L} = (L, L^\omega)$ where $L \subseteq \Sigma^*$ is a prefix-closed language, and $L^\omega \subseteq L$. These languages are also denoted by $\mathcal{L}(\mathbf{L}) = L$ and $\mathcal{L}^\omega(\mathbf{L}) = L^\omega$.

The prefix-closed behaviour $\mathcal{L}(\mathbf{L})$ contains possibly incomplete system executions. The (not necessarily prefix-

The authors are with the Department of Computer Science, University of Waikato, Hamilton, New Zealand, {siw4,robi}@waikato.ac.nz

978-1-4799-2836-1/14/$31.00 ©2014 IEEE

closed) sublanguage $\mathcal{L}^\omega(\mathbf{L}) \subseteq \mathcal{L}(\mathbf{L})$ is the so-called *marked* behaviour and contains traces representing completed tasks.

Language operations are applied to discrete events systems by applying them to both components. For example, if $\mathbf{L}_i = (L_i, L_i^\omega)$ for $i = 1, 2$, then $\mathbf{L}_1 \parallel \mathbf{L}_2 = (L_1 \parallel L_2, L_1^\omega \parallel L_2^\omega)$, and the same notation is used for $\cup$. Discrete event systems form a lattice with inclusion, $\mathbf{L}_1 \subseteq \mathbf{L}_2$, defined to hold if and only if $L_1 \subseteq L_2$ and $L_1^\omega \subseteq L_2^\omega$.

Alternatively, it is common to model discrete event systems as finite-state machines or automata.

Definition 2: A (nondeterministic) *automaton* is a tuple $\mathbf{G} = \langle \Sigma, Q, \rightarrow, Q^\circ, Q^\omega \rangle$ where Σ is a finite set of *events*, Q is a set of *states*, $\rightarrow \subseteq Q \times \Sigma \times Q$ is the *state transition relation*, $Q^\circ \subseteq Q$ is the set of *initial states*, and $Q^\omega \subseteq Q$ is the set of *marked states*.

$\mathbf{G}$ is *finite-state* if the state set Q is finite, and $\mathbf{G}$ is *deterministic* if $|Q^\circ| \leq 1$ and $x \xrightarrow{\sigma} y_1$ and $x \xrightarrow{\sigma} y_2$ always implies $y_1 = y_2$. Here, the transition relation is written in infix notation, $x \xrightarrow{\sigma} y$, and extended to traces in Σ^* in the standard way. Also, $\mathbf{G} \xrightarrow{s} x$ means $x^\circ \xrightarrow{s} x$ for some $x^\circ \in Q^\circ$. The prefix-closed and marked languages of an automaton $\mathbf{G}$ are

$$\mathcal{L}(\mathbf{G}) = \{\, s \in \Sigma^* \mid \mathbf{G} \xrightarrow{s} y \text{ for some } y \in Q \,\} \; ; \qquad (1)$$

$$\mathcal{L}^\omega(\mathbf{G}) = \{\, s \in \Sigma^* \mid \mathbf{G} \xrightarrow{s} y^\omega \text{ for some } y^\omega \in Q^\omega \,\} \; . \qquad (2)$$

Using these definitions, an automaton $\mathbf{G}$ is also considered as the Σ-DES $\mathbf{G} = (\mathcal{L}(\mathbf{G}), \mathcal{L}^\omega(\mathbf{G}))$.

C. Supervisory Control

Given a *plant* $\mathbf{L}$ and a *specification* $\mathbf{K}$, *supervisory control theory* [1] is concerned about the question whether and how the plant can be controlled in such a way that the specification is satisfied. This is dependent on the conditions of *controllability*, *normality*, and *nonblocking*.

Definition 3: Let $\mathbf{K}$ be a $\Sigma_\mathbf{K}$-DES, $\mathbf{L}$ a $\Sigma_\mathbf{L}$-DES, and let $\Sigma = \Sigma_\mathbf{K} \cup \Sigma_\mathbf{L}$. Then $\mathbf{K}$ is *controllable* with respect to $\mathbf{L}$ if

$$P_\Sigma^{-1}(\mathcal{L}(\mathbf{K}))\Sigma_{\mathrm{uc}} \cap P_\Sigma^{-1}(\mathcal{L}(\mathbf{L})) \subseteq P_\Sigma^{-1}(\mathcal{L}(\mathbf{K})) \; . \qquad (3)$$

Definition 4: Let $\mathbf{K}$ be a $\Sigma_\mathbf{K}$-DES, $\mathbf{L}$ a $\Sigma_\mathbf{L}$-DES, and let $\Sigma = \Sigma_\mathbf{K} \cup \Sigma_\mathbf{L}$. Then $\mathbf{K}$ is *normal* with respect to $\mathbf{L}$ if

$$P_\Sigma^{-1}(P_{\Sigma_\mathrm{o} \cap \Sigma_\mathbf{K}}(\mathcal{L}(\mathbf{K}))) \cap P_\Sigma^{-1}(\mathcal{L}(\mathbf{L})) \subseteq P_\Sigma^{-1}(\mathcal{L}(\mathbf{K})) \; . \qquad (4)$$

Controllability expresses that a supervisor cannot disable uncontrollable events, and normality expresses that a supervisor cannot detect the occurrence of unobservable events. Every controllable and normal behaviour can be implemented by a supervisor that only uses observable events as input and only disables controllable events.

In addition to the safety properties of controllability and normality, it is common to require the *nonblocking* property to ensure some form liveness.

Definition 5: A Σ-DES $\mathbf{L}$ is called *standard nonblocking* (or simply *nonblocking*) if, for every trace $s \in \mathcal{L}(\mathbf{L})$, there exists a trace $t \in \Sigma^*$ such that $st \in \mathcal{L}^\omega(\mathbf{L})$.

If a given system behaviour $\mathbf{K}$ is not controllable, normal, or nonblocking, then this behaviour cannot be implemented through control or is undesirable due to *livelock* or *deadlock*. The question then arises whether $\mathbf{K}$ can somehow be modified to satisfy the requirements. A key result from supervisory control theory states that every DES $\mathbf{K}$ has a largest possible sub-behaviour $\mathbf{K}' \subseteq \mathbf{K}$ that exhibits the desired properties of controllability, normality, and nonblocking.

Theorem 1: [1] Let $\mathbf{K}$ and $\mathbf{L}$ be two DES. There exists a unique supremal sub-behaviour $\sup\mathcal{CN}(\mathbf{K}) \subseteq \mathbf{K}$ that is controllable, normal, and nonblocking:

$$\begin{aligned} \sup\mathcal{CN}_\mathbf{L}(\mathbf{K}) = \bigcup \{\, & \mathbf{K}' \subseteq \mathbf{K} \mid \mathbf{K}' \text{ is controllable} \\ & \text{and normal with respect to } \mathbf{L}\text{, and} \\ & \mathbf{K}' \parallel \mathbf{L} \text{ is nonblocking} \,\} \; . \end{aligned} \qquad (5)$$

Furthermore, if $\mathbf{K}$ and $\mathbf{L}$ are represented by finite-state automata, a finite-state representation of the supremal controllable, normal, and nonblocking sub-behaviour $\sup\mathcal{CN}_\mathbf{L}(\mathbf{K})$ can be computed using a fixpoint iteration. This computation is called supervisor *synthesis*, and its result can be used to implement an appropriate supervisor [1].

III. Applications

This paper is concerned about the nonblocking property and its use in synthesis. In the following, two examples are discussed where the synthesis of a least restrictive supervisor using the standard nonblocking property from Def. 5 gives unexpected and probably undesirable results.

A. Computer-Controlled Board Game

A board game is to be controlled, where a computer player and an opponent are taking moves in turn [5]. The control objective it to prevent the computer player from losing, while it is always possible for the game to end, either by the computer player winning or by a draw being declared. This is achieved by marking all states where the computer player has won, or the game is over without a winner. A least restrictive nonblocking supervisor can be synthesised to ensure that the game can always end in the desired way.

To complicate the example slightly, a reset feature is added: an additional event reset is introduced, which can always be executed by the environment and resets the game to its initial state. With this addition, the standard nonblocking property is much less expressive. Now, a least restrictive supervisor may allow the game to enter states where defeat for the computer player is inevitable, however due the omnipresent possibility of reset, the system is still nonblocking as long as there is some way of ending the game from its initial state. A synthesised supervisor may exploit this and make bad moves, knowing it is always possible to restart. In this modified model, it is much more interesting to synthesise a supervisor to ensure that *"the game can always end, even if* reset *is not used."*

B. Manufacturing Cell

Fig. 1 shows a modified version of a manufacturing cell proposed in [9], which consists of a robot, a machine, two conveyors, two buffers, and a switch. The machine (plant $\mathbf{machine}$) can manufacture two types of products. Event $\mathsf{start}[k]$ initiates the manufacturing of a type k product ($k = 1$ or $k = 2$) from a workpiece in input buffer $\mathbf{inbuf}$, which upon completion is placed in output buffer $\mathbf{outbuf}$,

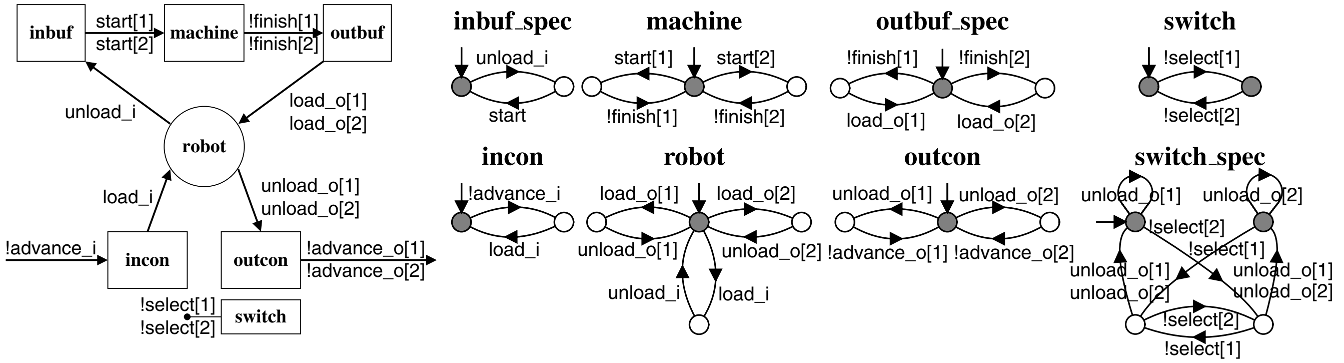

Fig. 1. Manufacturing cell example. Uncontrollable events are prefixed with !, and all events are observable.

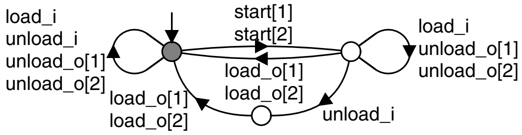

Fig. 2. Synthesised manufacturing cell supervisor.

indicated by the uncontrollable event !finish$[k]$. The robot (plant $\mathbf{robot}$) takes workpieces from the input conveyor (plant $\mathbf{incon}$) on event load_i and puts them in $\mathbf{inbuf}$ on event unload_i, and it takes type k products from $\mathbf{outbuf}$ on event load_o$[k]$ and puts them on the output conveyor (plant $\mathbf{outcon}$) on event unload_o$[k]$. The conveyors can be advanced to bring in new workpieces (!advance_i), or to remove completed products (!advance_o$[k]$). Specifications $\mathbf{inbuf\_spec}$ and $\mathbf{outbuf\_spec}$ request a supervisor that prevents overflow and underflow of two one-place buffers.

In addition, there is a switch (plant $\mathbf{switch}$) that allows the user to choose the type of products to be delivered. Specification $\mathbf{switch\_spec}$ requires that, when the user changes the desired output type to k (!select$[k]$), at most one product of the other type may be released from the cell; after that only type k products may be released (unload_o$[k]$) until the switch is operated again.

The model in Fig. 1 is not controllable and blocking. Standard synthesis [1] with supervisor reduction [10] gives the least restrictive supervisor in Fig. 2. This supervisor correctly prevents buffer overflow by not allowing the machine to start before the output buffer is empty, and prevents deadlock by restricting the number of workpieces in the cell to two.

The supervisor does not distinguish between start$[1]$ and start$[2]$, always allowing both types of products to be manufactured. This works because specification $\mathbf{switch\_spec}$ can be satisfied by disabling the controllable event unload_o$[k]$ when the robot holds a workpiece of an undesired type k, delaying delivery until the user changes the switch with another !select$[k]$ event. While this is the least restrictive controllable and nonblocking behaviour, it seems unreasonable to delay delivery and override the user's choice in this way. A more reasonable supervisor would respect the user's choice when starting the machine, instead of relying on the user to request delivery of what has already been produced.

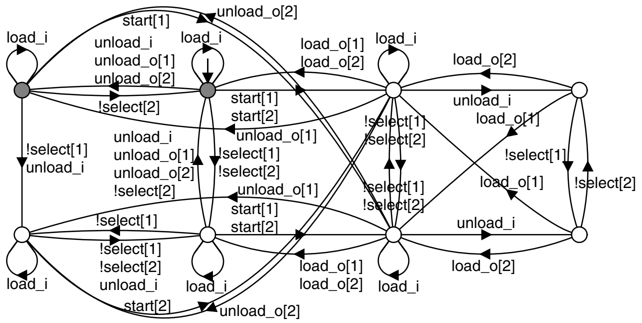

Fig. 3. Synthesised manufacturing cell supervisor with progressive events.

IV. Nonblocking with Progressive Events

A. Progressive Events

To provide a better way of modelling examples such as those in Sect. III, this section proposes to distinguish events that can be used to establish the nonblocking property from other events. Independently of controllability and observability, the event set Σ is partitioned into the sets $\Sigma_{\rm p}$ of *progressive* events and $\Sigma_{\rm np}$ of *non-progressive* events.

Definition 6: Let $\mathbf{L}$ be a Σ-DES, and let $\Sigma_{\rm p} \subseteq \Sigma$. Then $\mathbf{L}$ is $\Sigma_{\rm p}$-*nonblocking* if, for every trace $s \in \mathcal{L}(\mathbf{L})$, there exists a trace $t \in \Sigma_{\rm p}^*$ such that $st \in \mathcal{L}^\omega(\mathbf{L})$.

Nonblocking with progressive events requires that, from all reachable states, it is possible to reach a marked state using only progressive events. Non-progressive events are assumed to occur only occasionally or as external input, and a supervisor should not rely on them for task completion.

Definition 7: Let $\mathbf{K}$ and $\mathbf{L}$ be two DES, and let $\Sigma_{\rm p}$ be a set of progressive events. The least restrictive controllable, normal, and $\Sigma_{\rm p}$-nonblocking sub-behaviour of $\mathbf{K}$ with respect to $\mathbf{L}$ is

$$\sup\mathcal{CN}_{\mathbf{L},\Sigma_{\rm p}}(\mathbf{K}) = \bigcup \{\, \mathbf{K}' \subseteq \mathbf{K} \mid \mathbf{K}' \text{ is controllable and normal with respect to } \mathbf{L}, \text{ and } \mathbf{K}' \parallel \mathbf{L} \text{ is } \Sigma_{\rm p}\text{-nonblocking} \,\} \;.$$

Def. 7 redefines the objective of synthesis to use the modified nonblocking property. It follows from Prop. 2 below that the definition is sound in that it indeed defines a controllable, normal, and $\Sigma_{\rm p}$-nonblocking behaviour.

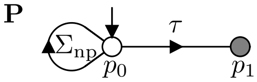

Fig. 4. The DES $\mathbf{P}$ to express Σ_{p}-nonblocking as standard nonblocking. The selfloop marked Σ_{np} stands for transitions with all events in Σ_{np}, and $\tau \notin \Sigma$ is a new event that does not appear elsewhere in the system.

In Sect. III, events reset and !select$[k]$ would be non-progressive. Then a Σ_{p}-nonblocking supervisor ensures task completion even if the game is not reset, or the manufacturing cell user never changes the requested workpiece type. Fig. 3 shows a least restrictive reduced supervisor for the manufacturing cell subject to the !select$[k]$ events being non-progressive. In addition to preventing buffer overflow and deadlock, this supervisor prevents the machine from producing a second workpiece while another is being delivered.

B. Relationship to Standard Nonblocking

Clearly, standard nonblocking is a special case of nonblocking with progressive events as Defs. 5 and 6 coincide when $\Sigma_{\mathrm{p}} = \Sigma$. If there are non-progressive events, then nonblocking with progressive events is a stronger condition.

Yet, nonblocking with progressive events can be expressed using standard nonblocking by means of an additional DES $\mathbf{P}$ as shown in Fig. 4, which uses a new event τ that disables all non-progressive events. Initially, non-progressive events are possible, but τ may be executed at any time, taking $\mathbf{P}$ to state p_1 where only progressive events can occur. When $\mathbf{P}$ is composed with a system to be analysed, all states remain reachable, yet standard nonblocking can only hold if marked states can be reached using progressive events only.

Proposition 2: Let $\mathbf{L}$ be a Σ-DES with $\Sigma = \Sigma_{\mathrm{p}} \dot\cup \Sigma_{\mathrm{np}}$. Then $\mathbf{L}$ is Σ_{p}-nonblocking if and only if $\mathbf{L} \parallel \mathbf{P}$ is nonblocking, where $\mathbf{P} = (\overline{\Sigma_{\mathrm{np}}^*\tau}, \Sigma_{\mathrm{np}}^*\tau)$ is the $(\Sigma_{\mathrm{np}} \cup \{\tau\})$-DES in Fig. 4.

Proof: First assume that $\mathbf{L}$ is Σ_{p}-nonblocking, and let $s \in \mathcal{L}(\mathbf{L} \parallel \mathbf{P})$. Then $P_\Sigma(s) \in \mathcal{L}(\mathbf{L})$, and as $\mathbf{L}$ is Σ_{p}-nonblocking, there exists $t \in \Sigma_{\mathrm{p}}^*$ such that $P_\Sigma(s)t \in \mathcal{L}^\omega(\mathbf{L})$, This implies $st\tau \in P_{\Sigma\cup\{\tau\}}^{-1}(\mathcal{L}^\omega(\mathbf{L}))$. Since $t \in \Sigma_{\mathrm{p}}^*$, it holds that $P_{\Sigma_{\mathrm{np}}\cup\{\tau\}}(st\tau) = P_{\Sigma_{\mathrm{np}}\cup\{\tau\}}(s)\tau \in \mathcal{L}^\omega(\mathbf{P})$, and thus $st\tau \in P_{\Sigma\cup\{\tau\}}^{-1}(\mathcal{L}^\omega(\mathbf{P}))$. Therefore, $st\tau \in \mathcal{L}^\omega(\mathbf{L} \parallel \mathbf{P})$, i.e., $\mathbf{L} \parallel \mathbf{P}$ is nonblocking.

Conversely assume $\mathbf{L}\|\mathbf{P}$ is nonblocking, and let $s \in \mathcal{L}(\mathbf{L})$. Then $s\tau \in \mathcal{L}(\mathbf{L} \parallel \mathbf{P})$. As $\mathbf{L} \parallel \mathbf{P}$ is nonblocking, there exists $t \in \Sigma^*$ such that $s\tau t \in \mathcal{L}^\omega(\mathbf{L} \parallel \mathbf{P})$. Then $P_{\Sigma_{\mathrm{np}}\cup\{\tau\}}(s\tau t) \in \mathcal{L}^\omega(\mathbf{P})$, which by construction of $\mathbf{P}$ implies $P_{\Sigma_{\mathrm{np}}\cup\{\tau\}}(t) = \varepsilon$, i.e., $t \in \Sigma_{\mathrm{p}}^*$. Since furthermore $st = P_\Sigma(s\tau t) \in \mathcal{L}^\omega(\mathbf{L})$, it follows that $\mathbf{L}$ is Σ_{p}-nonblocking. □

Prop. 2 shows that any nonblocking verification task with progressive events can be reduced to a standard nonblocking verification task. However, composition with the additional automaton $\mathbf{P}$ doubles the state space and verification time.

The extra effort is not necessary. Standard nonblocking can be checked by searching backwards from marked states to see whether all states are reached. By changing the backward search to use progressive events only, nonblocking with progressive events can be checked on the original system state space, exploring less transitions than a standard nonblocking check.

Prop. 2 is of theoretical interest, because it shows that progressive events do not add to the expressive power of standard nonblocking, and it can be of practical use, because it shows that a wide variety of nonblocking verification algorithms, particularly *compositional* verification [11], can also be used with progressive events.

It is not immediately clear whether the additional DES $\mathbf{P}$ can also be used to express synthesis with progressive events as standard synthesis. Indeed, if there are uncontrollable non-progressive events, then $\mathbf{P}$ used as an additional plant will disable some uncontrollable events, and a supervisor could wait for the auxiliary event τ to occur in order to avoid controllability problems.

This is avoided if τ is *unobservable*. Then the supervisor cannot distinguish the states of $\mathbf{P}$, so it has to enable uncontrollable events enabled in p_0 and at the same time ensure task completion from p_1. Lemma 3 shows for unobservable τ that controllability and normality are preserved by the addition of $\mathbf{P}$, which together with Prop. 2 implies the preservation of synthesis results as shown in Prop. 4.

Lemma 3: Let $\mathbf{K}$ and $\mathbf{L}$ be Σ-DES with $\Sigma = \Sigma_{\mathrm{p}} \dot\cup \Sigma_{\mathrm{np}}$, and let $\mathbf{P} = (\overline{\Sigma_{\mathrm{np}}^*\tau}, \Sigma_{\mathrm{np}}^*\tau)$ be the $(\Sigma_{\mathrm{np}} \cup \{\tau\})$-DES in Fig. 4 where $\tau \notin \Sigma$ is an uncontrollable and unobservable event.

(i) $\mathbf{K}$ is controllable with respect to $\mathbf{L}$ if and only if $\mathbf{K}$ is controllable with respect to $\mathbf{L} \parallel \mathbf{P}$.

(ii) $\mathbf{K}$ is normal with respect to $\mathbf{L}$ if and only if $\mathbf{K}$ is normal with respect to $\mathbf{L} \parallel \mathbf{P}$.

Proof: (i) First assume that $\mathbf{K}$ is controllable with respect to $\mathbf{L} \parallel \mathbf{P}$, and let $sv \in \mathcal{L}(\mathbf{K})\Sigma_{\mathrm{uc}} \cap \mathcal{L}(\mathbf{L})$. Then $sv \in \Sigma^*$, and $P_{\Sigma_{\mathrm{np}}\cup\{\tau\}}(sv) \in \mathcal{L}(\mathbf{P})$ by construction of $\mathbf{P}$, and thus $sv \in P_{\Sigma\cup\{\tau\}}^{-1}(\mathcal{L}(\mathbf{K}))\Sigma_{\mathrm{uc}} \cap \mathcal{L}(\mathbf{L} \parallel \mathbf{P}) \subseteq P_{\Sigma\cup\{\tau\}}^{-1}(\mathcal{L}(\mathbf{K}))$ as $\mathbf{K}$ is controllable with respect to $\mathbf{L} \parallel \mathbf{P}$. It follows that $\mathbf{K}$ is controllable with respect to $\mathbf{L}$. The converse inclusion holds by Prop. 3 in [12].

(ii) First assume that $\mathbf{K}$ is normal with respect to $\mathbf{L}$, and let $s \in P_{\Sigma\cup\{\tau\}}^{-1}(P_{\Sigma_{\mathrm{o}}}(\mathcal{L}(\mathbf{K}))) \cap \mathcal{L}(\mathbf{L} \parallel \mathbf{P})$. Then clearly $P_\Sigma(s) \in P_\Sigma^{-1}(P_{\Sigma_{\mathrm{o}}}(\mathcal{L}(\mathbf{K}))) \cap \mathcal{L}(\mathbf{L}) \subseteq \mathcal{L}(\mathbf{K})$ as $\mathbf{K}$ is normal with respect to $\mathbf{L}$. Thus, $\mathbf{K}$ is normal with respect to $\mathbf{L} \parallel \mathbf{P}$.

Conversely assume $\mathbf{K}$ is normal with respect to $\mathbf{L} \parallel \mathbf{P}$, and let $s \in P_\Sigma^{-1}(P_{\Sigma_{\mathrm{o}}}(\mathcal{L}(\mathbf{K}))) \cap \mathcal{L}(\mathbf{L})$. Then $s \in \Sigma^*$ and $P_{\Sigma_{\mathrm{np}}\cup\{\tau\}}(s) \in \Sigma_{\mathrm{np}}^* \subseteq \mathcal{L}(\mathbf{P})$ and $s \in P_{\Sigma\cup\{\tau\}}^{-1}(P_{\Sigma_{\mathrm{o}}}(\mathcal{L}(\mathbf{K}))) \cap \mathcal{L}(\mathbf{L} \parallel \mathbf{P}) \subseteq \mathcal{L}(\mathbf{K})$, since $\mathbf{K}$ is normal with respect to $\mathbf{L} \parallel \mathbf{P}$. This shows that $\mathbf{K}$ is normal with respect to $\mathbf{L}$. □

Proposition 4: Let $\mathbf{K}$ and $\mathbf{L}$ be Σ-DES with $\Sigma = \Sigma_{\mathrm{p}} \dot\cup \Sigma_{\mathrm{np}}$, and let $\mathbf{P} = (\overline{\Sigma_{\mathrm{np}}^*\tau}, \Sigma_{\mathrm{np}}^*\tau)$ be the $(\Sigma_{\mathrm{np}} \cup \{\tau\})$-DES in Fig. 4 where $\tau \notin \Sigma$ is an uncontrollable and unobservable event. Then $\sup\mathcal{CN}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}) = P_\Sigma(\sup\mathcal{CN}_{\mathbf{L}\|\mathbf{P}}(\mathbf{K}))$.

Proof: Consider an arbitrary sub-behaviour $\mathbf{K}' \subseteq \mathbf{K}$. In Lemma 3 it has been shown that $\mathbf{K}'$ is controllable and normal with respect to $\mathbf{L}$ if and only if $\mathbf{K}'$ is controllable and normal with respect to $\mathbf{L} \parallel \mathbf{P}$, and in Prop. 2 it has been shown that $\mathbf{K}' \parallel \mathbf{L}$ is Σ_{p}-nonblocking if and only if $\mathbf{K}' \parallel \mathbf{L} \parallel \mathbf{P}$ is nonblocking. As this holds for all sub-behaviours $\mathbf{K}'$ of $\mathbf{K}$, the least restrictive sub-behaviours must also be equal. □

Thus, synthesis with progressive events can be achieved using standard synthesis methods. However, the introduced automaton $\mathbf{P}$ includes the unobservable event τ, making it necessary to use the more complex synthesis algorithm with unobservable events [1], even if the original model only has observable events.

C. Direct Synthesis Algorithm

This section proposes a direct synthesis algorithm with progressive events, which avoids the unobservable event τ in the case of *total observation*, i.e., when all events are observable. The following synthesis objective is considered.

Definition 8: Let $\mathbf{K}$ and $\mathbf{L}$ be Σ-DES, and let $\Sigma_{\mathrm{p}} \subseteq \Sigma$. The least restrictive controllable and Σ_{p}-nonblocking sub-behaviour of $\mathbf{K}$ with respect to $\mathbf{L}$ is

$$\sup\mathcal{C}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}) = \bigcup \{\, \mathbf{K}' \subseteq \mathbf{K} \mid \mathbf{K}' \text{ is controllable with respect to } \mathbf{L}, \text{ and } \mathbf{K}' \parallel \mathbf{L} \text{ is } \Sigma_{\mathrm{p}}\text{-nonblocking} \,\} . \tag{6}$$

Def. 9 defines an operator on the sub-behaviours of $\mathbf{L}$, which afterwards is shown to have the above $\sup\mathcal{C}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K})$ as its *greatest fixpoint* [13].

Definition 9: Let $\mathbf{L}$ be a Σ-DES, and let $\Sigma_{\mathrm{p}} \subseteq \Sigma$. The operator $\Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}$ on the lattice of Σ-DES is defined by

$$\Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}) = (\theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}), \theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}) \cap \mathcal{L}^\omega(\mathbf{K})) \; ; \tag{7}$$

$$\theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}) = \theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}) \cap \theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}) \; ; \tag{8}$$

$$\theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}) = \{\, s \in \mathcal{L}(\mathbf{K}) \mid \text{for all } r \sqsubseteq s \text{ and } \upsilon \in \Sigma_{\mathrm{uc}} \text{ such that } r\upsilon \in \mathcal{L}(\mathbf{L}), \text{ it holds that } r\upsilon \in \mathcal{L}(\mathbf{K}) \,\} \; ;$$

$$\theta^{\mathrm{nonb}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}) = \{\, s \in \mathcal{L}(\mathbf{K}) \mid \text{for all } r \sqsubseteq s \text{ there exists } t \in \Sigma_{\mathrm{p}}^* \text{ such that } rt \in \mathcal{L}^\omega(\mathbf{L} \parallel \mathbf{K}) \,\} \; .$$

Proposition 5: Let $\mathbf{L}$ and $\mathbf{K}$ be a Σ-DES such that $\mathbf{K} \subseteq \mathbf{L}$, and let $\Sigma_{\mathrm{p}} \subseteq \Sigma$. Then $\mathbf{K} \subseteq \Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K})$, if and only if $\mathbf{K}$ is controllable with respect to $\mathbf{L}$ and $\mathbf{L} \parallel \mathbf{K}$ is Σ_{p}-nonblocking.

Proof: First assume $\mathbf{K} \subseteq \Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K})$. To see that $\mathbf{K}$ is controllable with respect to $\mathbf{L}$, let $s \in \mathcal{L}(\mathbf{K})$ and $\upsilon \in \Sigma_{\mathrm{uc}}$ such that $s\upsilon \in \mathcal{L}(\mathbf{L})$. As $s \in \mathcal{L}(\mathbf{K})$ and $\mathbf{K} \subseteq \Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K})$, it holds that $s \in \theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K})$, which implies $s\upsilon \in \mathcal{L}(\mathbf{K})$. It follows by Def. 3 that $\mathbf{K}$ is controllable with respect to $\mathbf{L}$. To see that $\mathbf{K} \parallel \mathbf{L}$ is Σ_{p}-nonblocking, let $s \in \mathcal{L}(\mathbf{K} \parallel \mathbf{L})$. Then $s \in \mathcal{L}(\mathbf{K}) \subseteq \theta^{\mathrm{nonb}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K})$, i.e., there exists $t \in \Sigma_{\mathrm{p}}^*$ such that $st \in \mathcal{L}^\omega(\mathbf{L} \parallel \mathbf{K})$. Thus, $\mathbf{K} \parallel \mathbf{L}$ is Σ_{p}-nonblocking.

Conversely, assume that $\mathbf{K}$ is controllable with respect to $\mathbf{L}$ and $\mathbf{L} \| \mathbf{K}$ is Σ_{p}-nonblocking, and let $s \in \mathcal{L}(\mathbf{K})$. Let $r \sqsubseteq s$ and $\upsilon \in \Sigma_{\mathrm{uc}}$ such that $r\upsilon \in \mathcal{L}(\mathbf{L})$. As $\mathbf{K}$ is controllable with respect to $\mathbf{L}$, it follows that $r\upsilon \in \mathcal{L}(\mathbf{K})$ and thus $s \in \theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K})$. Further, as $\mathbf{L} \parallel \mathbf{K}$ is Σ_{p}-nonblocking, for $r \in \mathcal{L}(\mathbf{K}) \subseteq \mathcal{L}(\mathbf{L})$, there exists $t \in \Sigma_{\mathrm{p}}^*$ such that $rt \in \mathcal{L}^\omega(\mathbf{L} \parallel \mathbf{K})$, i.e., $s \in \theta^{\mathrm{nonb}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K})$. Thus $s \in \theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K})$, and it follows from (7) that $\mathbf{K} \subseteq \Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K})$. □

Proposition 6: Let $\mathbf{L}$, $\mathbf{K}_1$, and $\mathbf{K}_2$ be Σ-DES and $\Sigma_{\mathrm{p}} \subseteq \Sigma$. If $\mathbf{K}_1 \subseteq \mathbf{K}_2$ then $\Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}_1) \subseteq \Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}_2)$.

Proof: Assume $\mathbf{K}_1 \subseteq \mathbf{K}_2$. Considering Def. 9, it is enough to show $\theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}_1) \subseteq \theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}_2)$ and $\theta^{\mathrm{nonb}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}_1) \subseteq \theta^{\mathrm{nonb}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}_2)$. Firstly, for $s \in \theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}_1)$, it holds that $s \in \mathcal{L}(\mathbf{K}_1) \subseteq \mathcal{L}(\mathbf{K}_2)$ and for all $r \sqsubseteq s$ and all $\upsilon \in \Sigma_{\mathrm{uc}}$ such that $r\upsilon \in \mathcal{L}(\mathbf{L})$ it holds that $r\upsilon \in \mathcal{L}(\mathbf{K}_1) \subseteq \mathcal{L}(\mathbf{K}_2)$, and thus $s \in \theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}_2)$. Secondly, for $s \in \theta^{\mathrm{nonb}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}_1)$ it holds that $s \in \mathcal{L}(\mathbf{K}_1) \subseteq \mathcal{L}(\mathbf{K}_2)$ and for all $r \sqsubseteq s$ there exists $t \in \Sigma_{\mathrm{p}}^*$ such that $rt \in \mathcal{L}^\omega(\mathbf{L} \parallel \mathbf{K}_1) \subseteq \mathcal{L}^\omega(\mathbf{L} \parallel \mathbf{K}_2)$, and thus $s \in \theta^{\mathrm{nonb}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{K}_2)$. □

Prop. 6 shows that $\Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}$ is a monotonic operator on the lattice of Σ-DES, so it follows by the Knaster-Tarski theorem [13] that $\Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}$ has a greatest fixpoint, which by Prop. 5 is the least restrictive controllable and Σ_{p}-nonblocking sub-behaviour of $\mathbf{L}$. To compute the fixpoint in a finite number of steps, Def. 11 introduces an iteration on the state set of $\mathbf{L} \parallel \mathbf{K}$, which in Prop. 7 is shown to be equivalent to the above $\Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}$.

Definition 10: The *restriction* of $\mathbf{G} = \langle \Sigma, Q, \rightarrow, Q^\circ, Q^\omega \rangle$ to $X \subseteq Q$ is $\mathbf{G}_{|X} = \langle \Sigma, X, \rightarrow_{|X}, Q^\circ \cap X, Q^\omega \cap X \rangle$ where $\rightarrow_{|X} = \{\, (x, \sigma, y) \in \rightarrow \mid x, y \in X \,\}$.

Definition 11: Let $\mathbf{L} = \langle \Sigma, Q_L, \rightarrow_L, Q_L^\circ, Q_L^\omega \rangle$ and $\mathbf{K} = \langle \Sigma, Q_K, \rightarrow_K, Q_K^\circ, Q_K^\omega \rangle$ be two deterministic finite-state automata, and let $\Sigma_{\mathrm{p}} \subseteq \Sigma$. The *synthesis step operator* $\bar{\Theta}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}} \colon 2^{Q_L \times Q_K} \rightarrow 2^{Q_L \times Q_K}$ for $\mathbf{L}$ and $\mathbf{K}$ is defined by

$$\bar{\Theta}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X) = \bar{\theta}^{\mathrm{cont}}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X) \cap \bar{\theta}^{\mathrm{nonb}}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X) \; ; \tag{9}$$

$$\bar{\theta}^{\mathrm{cont}}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X) = \{\, (x_L, x_K) \in Q_L \times Q_K \mid \text{for all } \upsilon \in \Sigma_{\mathrm{uc}} \text{ such that } x_L \xrightarrow{\upsilon}_L y_L \text{ there exists } y_K \in Q_K \text{ such that } (x_L, x_K) \xrightarrow{\upsilon} (y_L, y_K) \in X \,\} \; ;$$

$$\bar{\theta}^{\mathrm{nonb}}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X) = \{\, (x_L, x_K) \in Q_L \times Q_K \mid (x_L, x_K) \xrightarrow{t}_{|X} (y_L, y_K) \text{ for some } t \in \Sigma_{\mathrm{p}}^*,\ y_L \in Q_L^\omega, \text{ and } y_K \in Q_K^\omega \,\} \; .$$

Proposition 7: Let $\mathbf{L} = \langle \Sigma, Q_L, \rightarrow_L, Q_L^\circ, Q_L^\omega \rangle$ and $\mathbf{K} = \langle \Sigma, Q_K, \rightarrow_K, Q_K^\circ, Q_K^\omega \rangle$ be two deterministic finite-state automata, let $\mathbf{S} = \mathbf{L} \parallel \mathbf{K}$, and let $\Sigma_{\mathrm{p}} \subseteq \Sigma$. For every state set $X \subseteq Q_L \times Q_K$, it holds that $\Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{S}_{|X}) = \mathbf{S}_{|\bar{\Theta}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X)}$.

Proof: Based on Def. 1, Def. 10, and (2), it is enough to show $\mathcal{L}(\Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{S}_{|X})) = \mathcal{L}(\mathbf{S}_{|\bar{\Theta}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X)})$.

First let $s \in \mathcal{L}(\Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{S}_{|X})) = \theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{S}_{|X}) \cap \theta^{\mathrm{nonb}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{S}_{|X})$. Then $s \in \mathcal{L}(\mathbf{S}_{|X})$, so there exists a path $\mathbf{S}_{|X} \xrightarrow{s} (x_L, x_K) \in X$. It will be shown that $(x_L, x_K) \in \bar{\Theta}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X)$. First, for $\upsilon \in \Sigma_{\mathrm{uc}}$ such that $x_L \xrightarrow{\upsilon}_L y_L$, it holds that $s\upsilon \in \mathcal{L}(\mathbf{L})$, and since $s \in \theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{S}_{|X})$ it follows that $s\upsilon \in \mathcal{L}(\mathbf{S}_{|X})$. As $\mathbf{L}$ and $\mathbf{K}$ are deterministic, this implies $(x_L, x_K) \in \bar{\theta}^{\mathrm{cont}}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X)$. Second, as $s \in \theta^{\mathrm{nonb}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{S}_{|X})$, there exists $t \in \Sigma_{\mathrm{p}}^*$ such that $st \in \mathcal{L}^\omega(\mathbf{S}_{|X})$, which by determinism of $\mathbf{L}$ and $\mathbf{K}$ implies $(x_L, x_K) \xrightarrow{t}_{|X} (y_L, y_K) \in Q_L^\omega \times Q_K^\omega$. This shows $(x_L, x_K) \in \bar{\theta}^{\mathrm{cont}}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X) \cap \bar{\theta}^{\mathrm{nonb}}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X) = \bar{\Theta}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X)$. As the same can be shown for all prefixes $r \sqsubseteq s$, it follows that $s \in \mathcal{L}(\mathbf{S}_{|\bar{\Theta}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X)})$.

Conversely, let $s \in \mathcal{L}(\mathbf{S}_{|\bar{\Theta}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X)})$, and let $r \sqsubseteq s$. Then $\mathbf{S}_{|\bar{\Theta}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X)} \xrightarrow{r} (x_L, x_K) \in \bar{\Theta}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X)$. If $r\upsilon \in \mathcal{L}(\mathbf{L})$ for $\upsilon \in \Sigma_{\mathrm{uc}}$, then as $\mathbf{L}$ is deterministic also $x_L \xrightarrow{\upsilon}_L y_L$ for some $y_L \in Q_L$, which given $(x_L, x_K) \in \bar{\theta}^{\mathrm{cont}}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X)$ implies $r\upsilon \in \mathcal{L}(\mathbf{S}_{|X})$. Thus $s \in \theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{S}_{|X})$. Further, as $(x_L, x_K) \in \bar{\theta}^{\mathrm{nonb}}_{\mathbf{L},\mathbf{K},\Sigma_{\mathrm{p}}}(X)$ there exists $t \in \Sigma_{\mathrm{p}}^*$ such that $(x_L, x_K) \xrightarrow{t}_{|X} (y_L, y_K) \in Q_L^\omega \times Q_K^\omega$, and thus $s \in \theta^{\mathrm{nonb}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{S}_{|X})$. Therefore, $s \in \theta^{\mathrm{cont}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{S}_{|X}) \cap \theta^{\mathrm{nonb}}_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{S}_{|X}) = \mathcal{L}(\Theta_{\mathbf{L},\Sigma_{\mathrm{p}}}(\mathbf{S}_{|X}))$. □

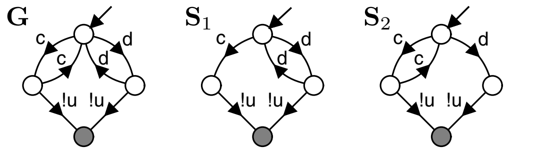

Fig. 5. A DES $\mathbf{G}$ that has no least restrictive supervisor that is nonblocking under control. Events c and d are controllable, while !u is uncontrollable.

By Prop. 7, a language-based step of $\Theta_{\mathbf{L},\Sigma_\mathrm{p}}$ gives the same result as a state-based step of $\bar{\Theta}_{\mathbf{L},\mathbf{K},\Sigma_\mathrm{p}}$. To synthesise the least restrictive controllable and Σ_p-nonblocking sub-behaviour of specification $\mathbf{K}$ with respect to plant $\mathbf{L}$, one first constructs the composition $\mathbf{S} = \mathbf{L} \| \mathbf{K}$. Then the iteration

$$X^0 = Q_L \times Q_K \qquad X^{i+1} = \bar{\Theta}_{\mathbf{L},\mathbf{K},\Sigma_\mathrm{p}}(X^i) \qquad (10)$$

converges against a greatest fixpoint X^n in a finite number of n steps, which by Prop. 7 satisfies $\mathbf{S}_{|X^n} = \sup\mathcal{C}_{\mathbf{L},\Sigma_\mathrm{p}}(\mathbf{K})$.

V. Related Work

This section relates nonblocking with progressive events to other nonblocking conditions studied in the literature.

Multi-tasking supervisory control [4] requires a synthesised supervisor to be nonblocking with respect to several sets of marked states at the same time. *Generalised nonblocking* [5] uses a second set of marked states to specify a subset of the states, from which marked states must be reachable. Both conditions are amenable to synthesis and can be combined with progressive events to further increase modelling capabilities.

The condition of *nonblocking under control* [7] is more similar to that of nonblocking with progressive events. When modelling a supervisor implementation, it is assumed that an implemented supervisor or *controller* sends controllable events as commands to the plant. Typically, the controller can generate several controllable events in quick sequence, and it is considered unlikely that uncontrollable events occur during such a sequence. Then it makes sense to require the system to complete its tasks using Σ_c-*complete* traces.

Definition 12: [7] Let $\mathbf{G} = \langle \Sigma, Q, \rightarrow, Q^\circ, Q^\omega \rangle$ and $\Sigma_\mathrm{c} \subseteq \Sigma$. The path $x_0 \xrightarrow{\sigma_1} x_1 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} x_n$ is Σ_c-*complete*, if for each $i = 1, \ldots, n$ it holds that either $\sigma_i \in \Sigma_\mathrm{c}$ or there do not exist $\sigma \in \Sigma_\mathrm{c}$ and $y \in Q$ such that $x_{i-1} \xrightarrow{\sigma} y$.

Definition 13: [7] Let $\mathbf{G} = \langle \Sigma, Q, \rightarrow, Q^\circ, Q^\omega \rangle$ and $\Sigma_\mathrm{c} \subseteq \Sigma$. Then $\mathbf{G}$ is *nonblocking under* Σ_c-*control* if for all paths $\mathbf{G} \xrightarrow{s} x$, there exists a Σ_c-complete path $x \xrightarrow{t} y^\omega$ such that $y^\omega \in Q^\omega$.

Nonblocking under control is similar to nonblocking with progressive events, in that it considers uncontrollable events as non-progressive in states where a controllable event is enabled. However, it depends on the state whether an event is progressive or not, and this dependency means that in general there do not exist least restrictive supervisors that are nonblocking under control.

For example, Fig. 5 shows a DES $\mathbf{G}$ which is not nonblocking under control. As both !u-transitions are only enabled in states where controllable events are also enabled, these transitions are considered as non-progressive and cannot be used to prove that the marked state is reachable. The two sub-behaviours $\mathbf{S}_1$ and $\mathbf{S}_2$ are nonblocking under control, however neither of them is least restrictive, and their least upper bound, $\mathbf{G}$, is not nonblocking under control.

It is shown in [14] how the property of nonblocking under control can be verified. Synthesis for this and similar properties can be achieved using ω-languages [8], however these methods do not in general produce a state-based supervisor that can be readily implemented.

VI. Conclusions

The condition of nonblocking with progressive events is introduced as an extension of standard nonblocking. It is shown that there are situations where synthesis using the standard nonblocking property results in an unexpected result, because the synthesised supervisor can complete its tasks only if certain rare or undesirable events occur. Using progressive events, it can be specified more precisely how a synthesised supervisor must complete its tasks. While progressive events increase the modelling capabilities, verification and synthesis can still be achieved without increase in complexity over the standard nonblocking property.

References

[1] P. J. G. Ramadge and W. M. Wonham, “The control of discrete event systems,” *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.

[2] A. Arnold, *Finite Transition Systems: Semantics of Communicating Systems*. Prentice-Hall, 1994.

[3] Y.-L. Chen, S. Lafortune, and F. Lin, “Design of nonblocking modular supervisors using event priority functions,” *IEEE Trans. Autom. Control*, vol. 45, no. 3, pp. 432–452, Mar. 2000.

[4] M. H. de Queiroz, J. E. R. Cury, and W. M. Wonham, “Multi-tasking supervisory control of discrete-event systems,” in *Proc. 7th Int. Workshop on Discrete Event Systems, WODES ’04*, Reims, France, Sept. 2004, pp. 175–180.

[5] R. Malik and R. Leduc, “Generalised nonblocking,” in *Proc. 9th Int. Workshop on Discrete Event Systems, WODES ’08*, Göteborg, Sweden, May 2008, pp. 340–345.

[6] R. J. Leduc, B. A. Brandin, M. Lawford, and W. M. Wonham, “Hierarchical interface-based supervisory control—part I: Serial case,” *IEEE Trans. Autom. Control*, vol. 50, no. 9, pp. 1322–1335, Sept. 2005.

[7] P. Dietrich, R. Malik, W. M. Wonham, and B. A. Brandin, “Implementation considerations in supervisory control,” in *Synthesis and Control of Discrete Event Systems*, B. Caillaud, P. Darondeau, L. Lavagno, and X. Xie, Eds. Kluwer, 2002, pp. 185–201.

[8] C. Baier and T. Moor, “A hierarchical control architecture for sequential behaviours,” in *Proc. 11th Int. Workshop on Discrete Event Systems, WODES ’12*, Guadalajara, Mexico, Oct. 2012, pp. 259–264.

[9] W. M. Wonham, “Supervisory control of discrete-event systems,” Systems Control Group, Dept. of Electrical Engineering, University of Toronto, ON, Canada; at http://www.control.utoronto.edu/DES/, 2009.

[10] R. Su and W. M. Wonham, “Supervisor reduction for discrete-event systems,” *Discrete Event Dyn. Syst.*, vol. 14, no. 1, pp. 31–53, Jan. 2004.

[11] R. Malik and R. Leduc, “Compositional nonblocking verification using generalised nonblocking abstractions,” *IEEE Trans. Autom. Control*, vol. 58, no. 8, pp. 1–13, Aug. 2013.

[12] B. A. Brandin, R. Malik, and P. Malik, “Incremental verification and synthesis of discrete-event systems guided by counter-examples,” *IEEE Trans. Control Syst. Technol.*, vol. 12, no. 3, pp. 387–401, May 2004.

[13] A. Tarski, “A lattice-theoretical fixpoint theorem and its applications,” *Pacific J. Math.*, vol. 5, no. 2, pp. 285–309, 1955.

[14] P. Malik, “From supervisory control to nonblocking controllers for discrete event systems,” Ph.D. dissertation, University of Kaiserslautern, Kaiserslautern, Germany, 2003.