Technical Report
Lehrstruhl für Regelungstechnik
Friedrich-Alexander Universität Erlangen-Nürnberg
Cauerstraße 7, D-91058 Erlangen, Germany

Supervisory Control of Non-Terminating Processes: an Interpretation of Liveness Properties

Thomas Moor, 2017-11-25 (2nd public revison[1])

Abstract: This report discusses the control of non-terminating processes that are representable as ω-languages. More specifically, I am interested in how technical liveness properties identified for the model are interpreted w.r.t. the actual phenomenon in closed-loop configuration with a physical implementation of a supervisor. *This report evolved through a number of stages, starting in late 2015. Its initial purpose was to support a discussion of the relationship between supervisory control and reactive synthesis. This topic was followed up in a cooperative research project with Anne-Kathrin Schmuck and Rupak Majumda, MPI for Software Systems, Kaiserslautern, Germany; see Schmuck et al. (2017). It was then decided to file the present report independently for its potential value as a "gentle introduction" to the supervision of ω-languages. The first public revision was substantially improved by Anne's feedback.*

Keywords: discrete-event systems, supervisory control, ω-languages, sequential behaviours, liveness properties

Introduction

For the common synthesis problem in supervisory control, we are given a plant and an upper bound specification on the closed-loop behaviour. Then, the task is to synthesise a controller that restricts the plant behaviour accordingly. We

[1] Fixed two annoying typos in technical formulae on 2018-10-22

consider the situation of non-terminating processes and represent the plant and the specification as ω-languages $\mathcal{L} \subseteq \Sigma^\omega$ and $\mathcal{E} \subseteq \Sigma^\omega$, respectively, where Σ^ω denotes the set of infinite length strings of symbols from some alphabet Σ. An ω-language is topologically closed if it can be characterised in terms of its prefix. The absence of this property is the main concern of this report: an ω-language that fails to be topologically closed expresses liveness properties other than not to deadlock with eventuality as the prototypical example. Regarding the synthesis problem, we distinguish four specific cases depending on which of the parameters $\mathcal{L}$ and $\mathcal{E}$ are assumed to be topologically closed.

(1) When the only liveness property to address is not to deadlock, the synthesis problem can be solved by a variation of methods used for supervisory control of $*$-languages. Technically, this corresponds to both $\mathcal{L}$ and $\mathcal{E}$ being closed. For this "basic case" we refer to Ramadge and Wonham (1987) with the further development to prevent deadlocks by Ramadge (1989) and by Kumar et al. (1992).

(2) When a language inclusion specification imposes additional liveness requirements, the synthesis problem can be identified as a special case of the study by Thistle and Wonham (1994a,b). Here, we assume $\mathcal{L}$ to be closed, with no further restrictions regarding the closedness of $\mathcal{E}$. The algorithmic solution proposed in the given references substantially differs from the methods commonly employed for $*$-languages. With $\mathcal{L} = \Sigma^\omega$, reactive synthesis in its basic form can be interpreted in this context.

(3) When the plant exhibits additional liveness properties which the supervisor shall respect, the synthesis problem is interpreted in the context of results reported by Ramadge (1989), with relevant aspects of the synthesis problem addressed by Kumar et al. (1992) and by Moor et al. (2012). Technically, this case is characterised by $\mathcal{E}$ to be relatively closed w.r.t. $\mathcal{L}$, with no further restrictions regarding the closedness of $\mathcal{L}$. As with case (1), this case can be addressed by a variation of methods used for $*$-languages.

(4) When the plant exhibits liveness properties that a supervisor may utilise in order to enforce further liveness properties imposed by a language inclusion specification, we again refer to Thistle and Wonham (1992, 1994a). Here, neither $\mathcal{L}$ nor $\mathcal{E}$ are assumed to be closed or relatively closed. Technically, this "most general case" subsumes the three previous cases.

Most technical results in this report are taken from the literature as cited above — there is nothing really new. The purpose of the discussion is to review the implicit requirements imposed on the phenomenon under control. The report is organised in Sections 1 to 4, one for each of the above cases, and an additional Section 5 to review liveness in the context of abstraction-based supervisory control. The latter appears to be a natural setting for relevant applications in which the plant fails to be topologically closed. The additional conditions imposed for abstraction-based synthesis are closely related to those conditions, that I believe are also sensible when encoding liveness properties of the plant as assumptions in an assume-guarantee

reasoning. A discussion of this conjecture was the initial motivation of this report and it is followed up in all detail by Schmuck et al. (2017).

Preliminaries

We outline notation, operators and elementary properties regarding $*$-languages and ω-languages as they are relevant to this report.

1: Finite strings and $*$-languages

Let Σ be a *finite alphabet*. The *Kleene-closure* Σ^* is the set of finite strings $s = \sigma_1\sigma_2\cdots\sigma_n$, $n \geq 1$, $\sigma_i \in \Sigma$, and the *empty string* $\epsilon \in \Sigma^*$, $\epsilon \notin \Sigma$. If for two strings $s, r \in \Sigma^*$ there exists $t \in \Sigma^*$, such that $s = rt$, we say r is a *prefix* of s, and write $r \leq s$; r is a *strict prefix* of s, and we write $r < s$, if $r \leq s$ and $r \neq s$. A *$*$-language* over Σ is a subset $L \subseteq \Sigma^*$.

2: Prefix and prefix-closure

The *prefix* of a $*$-language L is defined by $\operatorname{pre} L := \{r | \exists\, s \in L : r \leq s\} \subseteq \Sigma^*$. The prefix operator distributes over arbitrary unions of $*$-languages. However, for the intersection of two $*$-languages L and H over Σ, we have $\operatorname{pre}(L \cap H) \subseteq (\operatorname{pre} L) \cap (\operatorname{pre} H)$. If equality holds, L and H are said to be *non-conflicting*. This is trivially the case for $H \subseteq L$. The prefix of a $*$-language is also referred as its *prefix-closure* (or short *closure*). A language $L \subseteq \Sigma^*$ is *prefix-closed* or (short *closed*) if $L = \operatorname{pre} L$. Prefix-closedness is retained under arbitrary union and arbitrary intersection, i.e., for a family of closed $*$-languages, the intersection and the union are closed, too. A language K is *relatively prefix-closed w.r.t. L* if $K = (\operatorname{pre} K) \cap L$. Relative prefix-closedness w.r.t. a language L is retained under arbitrary union and arbitrary intersection. The intersection $(\operatorname{pre} K) \cap L$ is always relatively closed w.r.t. L. If a language K is relatively closed w.r.t. a closed language, then K itself is closed. Given three languages K, M, $L \subseteq \Sigma^*$, such that K is relatively closed w.r.t. M and M is relatively closed w.r.t. L, then K is relatively closed w.r.t. L.

3: Completeness and controllability

A $*$-language L is *complete* if for all $s \in L$ there exists $\sigma \in \Sigma$ such that $s\sigma \in \operatorname{pre} L$; see e.g. Kumar et al. (1992). Technically, $L = \emptyset$ is complete. Completeness is retained under arbitrary union. Completeness of a $*$-language is also referred to *liveness* and must not be confused with *behavioural completeness* as defined by Willems (1991). In the context of control and unless otherwise noted, the alphabets Σ, Σ_c and Σ_{uc} refer to the *common partitioning* $\Sigma = \Sigma_c \dot\cup \Sigma_{uc}$ in *controllable events*

Σ_c and *uncontrollable events* Σ_{uc}, respectively. Given two $*$-*languages* L and K over Σ, we say K is *controllable* w.r.t. L if $(\mathrm{pre}\, K)\Sigma_{uc} \cap (\mathrm{pre}\, L) \subseteq \mathrm{pre}\, K$. By Ramadge and Wonham (1987, 1989), controllability is retained under arbitrary union.

4: Infinite strings and ω-languages

The set of ω-*strings* (countably infinite length strings) over Σ is denoted $\Sigma^\omega := \{ w \,|\, w = \sigma_1\sigma_2\sigma_3\cdots, \text{ with } \sigma_i \in \Sigma \text{ for all } i \in \mathbb{N}\}$. An ω-*language* over Σ is a subset $\mathcal{L} \subseteq \Sigma^\omega$. If for two strings $w \in \Sigma^\omega$, $r \in \Sigma^*$, there exists $v \in \Sigma^\omega$ such that $w = rv$, we say r is a *strict prefix* of w and write $r < w$. The *prefix* of an ω-language $\mathcal{L} \subseteq \Sigma^\omega$ is defined $\mathrm{pre}\, \mathcal{L} = \{ s \,|\, \exists\, w \in \mathcal{L} : s < w\} \subseteq \Sigma^*$. Note that the prefix of any ω-language is complete. The prefix operator distributes over arbitrary unions of ω-languages. However, for the intersection of two ω-languages $\mathcal{L}$ and $\mathcal{H}$ over Σ, we have $\mathrm{pre}\,(\mathcal{L} \cap \mathcal{H}) \subseteq (\mathrm{pre}\, \mathcal{L}) \cap (\mathrm{pre}\, \mathcal{H})$.

5: Limit and topological closure

For a language $L \subseteq \Sigma^*$, the *limit* is defined $\lim L = \{ w \in \Sigma^\omega \,|\, w$ has infinitely many prefixes in $L\}$. If and only if a $*$-language $L \subseteq \Sigma^*$ is complete and prefix-closed, we have $\mathrm{pre} \lim L = L$; see Kumar et al. (1992). Thus, we have $\mathrm{pre}\, \mathcal{L} = \mathrm{pre} \lim \mathrm{pre}\, \mathcal{L}$ for arbitrary languages $\mathcal{L} \subseteq \Sigma^\omega$. The *closure* of an ω-language $\mathcal{L}$ is defined by $\mathrm{clo}\, \mathcal{L} := \lim \mathrm{pre}\, \mathcal{L}$. This notion of a closure can indeed be induced by a metric and therefore is also referred to as the *topological closure*. An ω-language $\mathcal{L}$ is *topologically closed* (or short *closed*) if $\mathcal{L} = \mathrm{clo}\, \mathcal{L}$. In the context of behavioural systems theory, topological closedness is referred to as *behavioural completeness*; see, e.g., Willems (1991), Definition II.4. The limit of a prefix-closed $*$-language is closed. The topological closure operator distributes over finite unions, however, for an arbitrary family of ω-languages $\mathcal{L}_a$, $a \in A$, we have $\cup_{a\in A} \mathrm{clo}\, \mathcal{L}_a \subseteq \mathrm{clo}\, (\cup_{a\in A} \mathcal{L}_a)$; see also (Ramadge, 1989). Topological closedness is retained under arbitrary intersection, but, and in contrast to prefix-closedness, it is not retained under arbitrary union. Given two ω-languages $\mathcal{K}$ and $\mathcal{L}$, we say $\mathcal{K}$ is *relatively closed w.r.t.* $\mathcal{L}$ if $\mathcal{K} = (\mathrm{clo}\, \mathcal{K}) \cap \mathcal{L}$. The intersection $(\mathrm{clo}\, \mathcal{K}) \cap \mathcal{L}$ is always relatively closed w.r.t. $\mathcal{L}$. If a language $\mathcal{K}$ is relatively closed w.r.t. a closed language, then $\mathcal{K}$ itself is closed. Given three languages $\mathcal{K}$, $\mathcal{M}$, $\mathcal{L} \subseteq \Sigma^*$, such that $\mathcal{K}$ is relatively closed w.r.t. $\mathcal{M}$ and $\mathcal{M}$ is relatively closed w.r.t. $\mathcal{L}$, then $\mathcal{K}$ is relatively closed w.r.t. $\mathcal{L}$.

6: Non-conflicting ω-languages

Two ω-languages $\mathcal{L}$ and $\mathcal{H}$ over Σ are *non-conflicting*, if $\mathrm{pre}\,(\mathcal{L} \cap \mathcal{H}) = (\mathrm{pre}\, \mathcal{L}) \cap (\mathrm{pre}\, \mathcal{H})$. The two languages $\mathcal{L}$ and $\mathcal{H}$ are *locally non-conflicting* if $(\mathrm{pre}\, \mathcal{L}) \cap (\mathrm{pre}\, \mathcal{H})$ is complete. Any two languages that are non-conflicting are also locally non-conflicting. Note that for $\mathcal{H} \subseteq \mathcal{L}$ both non-conflicting conditions are trivially satisfied. Provided that two ω-languages $\mathcal{L}$ and $\mathcal{H}$ are locally non-conflicting, $\mathrm{clo}\,(\mathcal{L} \cap \mathcal{H}) = (\mathrm{clo}\, \mathcal{L}) \cap (\mathrm{clo}\, \mathcal{H})$ is equivalent to the two languages to be non-conflicting. In particular, both non-conflicting conditions are equivalent for closed

languages.

7: Automata

An *automaton* is a tuple $G := (Q, \Sigma, \delta, Q_o, Q_m)$ with $Q_m \subseteq Q$, $Q_o \subseteq Q$ and the transition relation $\delta \subseteq Q \times \Sigma \times Q$. We write $\delta(q, \sigma)$ for the set of states that can be reached from $q \in Q$ by a transition labelled $\sigma \in \Sigma$, i.e., $\delta(q, \sigma) := \{ q' \in Q \,|\, (q, \sigma, q') \in \delta \}$, with the common inductive extension to a string-valued second argument $s \in \Sigma^*$ by $\delta(q, \epsilon) := \{q\}$ and $\delta(q, s\sigma) := \delta(\delta(q, s), \sigma)$. If $|Q_o| \leq 1$ and $|(\delta(q, s)| \leq 1$ for all $q \in Q$, $\sigma \in \Sigma$, then G is said to be *deterministic*. For deterministic automata, we occasionally write $\delta(q, s) = q'$, $q' \in Q$, and $\delta(q, s)!$ as short forms for $\delta(q, s) = \{q'\}$ and $\delta(q, s) \neq \emptyset$, respectively.

With G, we associate the *generated* $*$-language $L(G) := \{ s \in \Sigma^* \,|\, \delta(Q_o, s) \neq \emptyset \}$ and the *accepted* $*$-language $L_m(G) := \{ s \in \Sigma^* \,|\, \delta(Q_o, s) \cap Q_m \neq \emptyset \}$. Languages that are accepted by a finite automaton are referred to as *regular*. Given a $*$-language $L \subseteq \Sigma^*$, a minimal realisation can be constructed with the state set $Q = \Sigma^* / [\equiv_L]$, where $[\equiv_L] \subseteq \Sigma^* \times \Sigma^*$ denotes the Nerode equivalence w.r.t. L and is defined by $s' [\equiv_L] s''$ if and only if $(\forall\, t \in \Sigma^*)[\, s't \in L \Leftrightarrow s''t \in L]$. In particular, L is regular if and only if $[\equiv_L]$ consists of finitely many equivalence classes. Any finite automata can be converted to a deterministic finite automata without affecting the associated languages. Regarding ω-languages, we reinterpret G as *Büchi automaton* with associated *generated* ω-language $B(G) := \lim L(G)$ and *accepted* ω-language $B_m(G)$ consisting of all those infinite strings $w \in B(G)$ for which there exists a corresponding path through the graph δ which starts in Q_o and passes through Q_m infinitely often. For deterministic automata, this amounts to $B_m(G) = \lim L_m(G)$. The class of languages accepted by finite deterministic Büchi automata is a strict subset of the class of languages accepted by finite but not-necessarily deterministic Büchi automata. The latter are referred to as *ω-regular* languages.

1 The Basic Case

Following the introduction given in Ramadge and Wonham (1989), we consider phenomena that are characterised by distinguished internal configurations, possibly represented as a finite state set. Transitions from one to another configuration occur at discrete instances of physical time and are labelled with external *events* from some alphabet Σ. It is assumed that an observation of the phenomenon over a finite duration of physical time yields a finite string $s \in \Sigma^*$, i.e., the physical timing is regarded irrelevant and the string s is interpreted w.r.t. *logic time*. The set of all strings the phenomenon can generate during an arbitrary finite duration of physical time is referred to as the *local behaviour* $L_{loc} \subseteq \Sigma^*$. Note that, in order to generate a

string $s \in L_{\text{loc}}$, all prefixes pre s must have been generated before. Hence, the local behaviour is prefix closed. At this stage, we consider the *discrete-event system* L_{loc} as a formal *model of the phenomenon* and we summarise our considerations as an informal hypothesis that relates the phenomenon with model.

> *Hypothesis — the local behaviour as a model: during any finite duration of physical time, the phenomenon generates not more than a finite number of events; at any instance of physical time, the phenomenon may only generate an event that together with the event sequence generated so far forms a string from the local behaviour.*

Safety properties require "bad things not to happen". In the proposed modelling framework, safety properties can be conveniently expressed as an upper bound on the local behaviour: with $E \subseteq \Sigma^*$ the complement of the "bad things" we say that the *phenomenon exhibits the safety property* E if the model satisfies the inclusion $L_{\text{loc}} \subseteq E$. There are two paths to argue that E can be chosen prefix closed. First, the verb "to happen" is understood to refer to an instance of time and if "a bad thing ever happened" this will remain to be the case for all future. Thus, we may assume that the complement of E is suffix closed,

$$(\forall\ s,\ t \in \Sigma^*)[\ s \notin E \ \ \Rightarrow \ \ st \notin E\ ]\,, \tag{1}$$

which is equivalent to E being prefix closed. Second, since the models under consideration are prefix closed languages, the upper bound E in the inclusion $L_{\text{loc}} \subseteq E$ can be equivalently substituted by its supremal prefix closed subset.

Liveness properties require "good things to happen". In addressing non-terminating processes, the liveness property relevant for this section is *not to deadlock*. In terms of the model, we require that L_{loc} is complete, i.e.,

$$(\forall\ s \in L_{\text{loc}}\ \exists\, \sigma \in \Sigma)[\ s\sigma \in \text{pre}\, L_{\text{loc}}\ ]\,. \tag{2}$$

In contrast to the situation of safety properties, the technical requirement imposed on the model does not suffice to conclude a liveness property of the phenomenon. The following hypothesis is justified by the term "non-terminating process".

> *Hypothesis — non-terminating process: if, at any instance of physical time, the local behaviour indicates that the event sequence generated so far can be extended by one more event, then the phenomenon, for any configuration that is compatible with the event sequence generated so far, will eventually generate one more event.*
>
> *Consequence — provided that the local behaviour is complete, an infinite number of events will be generated during the elapse of an infinite physical duration.*

The notion of an infinite physical duration should be of no concern: while a physical phenomenon is not expected to be operational for an infinite duration, an infinite time axis is a common abstraction to reason about long term perspectives. Regarding the clause on the internal configuration, common supervisory control refers to a deterministic automaton realisation. Then, the above hypothesis requires that some enabled transition will be executed eventually, where the persistent existence of enabled transitions is implied by the technical liveness requirement imposed on the model. The latter implication does not hold for non-deterministic realisations. Here, the above hypothesis needs to be accompanied by the technical requirement that for every reachable state there exists at least one enabled transition.

Example: stack feeder, capacity 2, local behaviour. The deterministic automaton G in the below Figure 1 realises the local behaviour $L_{\text{loc}} = \mathrm{L}(G)$ of a prototypical stack feeder with capacity two. Its stack is initially loaded with two workpieces where the lower one can be fed to its environment, say, a transport system or a processing machine. Feeding is indicated by the event f. After feeding all workpieces, the stack feeder becomes empty, immediately indicated by event e. The event r indicates a reload to the full capacity from some unlimited supply of workpieces.

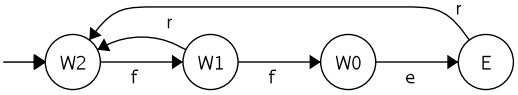

Figure 1: stack feeder, capacity 2, local behaviour

For supervisory control as proposed by Ramadge and Wonham (1987), the alphabet is composed as a disjoint union $\Sigma := \Sigma_{\text{c}} \dot{\cup} \Sigma_{\text{uc}}$ of *controllable events* Σ_{c} and *uncontrollable events* Σ_{uc}, respectively. The phenomenon under control, now also referred to as the *plant*, is equipped with an interface that allows an external agent to disable any controllable event at any physical time. Assuming that the decision of the agent is exclusively based on past events, the original literature models the agent as a *supervisor* $f\colon \Sigma^* \to \Gamma$ that maps past observations to *control patterns* $\Gamma := \{\, \gamma \,|\, \Sigma_{\text{uc}} \subseteq \gamma \subseteq \Sigma \,\}$, i.e., $f(s)$ denotes the set of events that the supervisor enables after the observation of $s \in L_{\text{loc}}$ and until the next event is generated. For the purpose of this report, the supervisor f can be equivalently represented by the language $H := \{\, s\sigma \in \Sigma^* \,|\, \sigma \in f(s) \,\} \cup \{\epsilon\} \subseteq \Sigma^*$ to obtain $f(s) = \{\, \sigma \in \Sigma \,|\, s\sigma \in H \,\}$. With this representation, a language $H \subseteq \Sigma^*$ corresponds to some supervisor f if and only if H is closed and satisfies the *universal controllability condition* $(\mathrm{pre}\, H)\Sigma_{\text{uc}} \subseteq \mathrm{pre}\, H$. Moreover, the local closed-loop behaviour amounts to the intersection $K_{\text{loc}} := L_{\text{loc}} \cap H$. Addressing non-terminating processes we follow Ramadge (1989) and require the closed loop K_{loc} not to deadlock, i.e., to be complete. The technical conditions imposed on the supervisor so far are summarised by the following notion of *local admissibility*.

Definition 1. Given an alphabet Σ with uncontrollable events Σ_{uc}, a language H is a *locally admissible controller* for the closed local behaviour $L_{\text{loc}} \subseteq \Sigma^*$ of a non-terminating process, if

[H0] H is closed,

[H1] $(\operatorname{pre} H)\Sigma_{\mathrm{uc}} \subseteq \operatorname{pre} H$, and

[H2] $L_{\mathrm{loc}} \cap (\operatorname{pre} H)$ is complete. □

To conclude from [H2] that the actual closed loop formed by an implementation of the supervisor and the phenomenon indeed exhibits the desired liveness property, we need to adapt our initial hypothesis regarding the phenomenon.

> *Hypothesis — supervised non-terminating process: whenever the supervisor agrees to at least one event the phenomenon can generate in any configuration compatible with the event sequence generated so far, then it will eventually generate one such event*
>
> *Consequence — under locally admissible control, an infinite number of events will be generated during the elapse of an infinite physical duration.*

The achievable closed loop behaviours are characterised as follows.

Theorem 2. Consider an alphabet Σ with uncontrollable events Σ_{uc}. For the closed local behaviour $L_{\mathrm{loc}} \subseteq \Sigma^*$ of a non-terminating process and a locally admissible controller $H \subseteq \Sigma^*$ let $K_{\mathrm{loc}} = L_{\mathrm{loc}} \cap H$. Then

[K0] K_{loc} is closed,

[K1] K_{loc} is controllable w.r.t. L_{loc},

[K2] K_{loc} is complete.

Moreover, if $L_{\mathrm{loc}} \neq \emptyset \neq H$, then $K_{\mathrm{loc}} \neq \emptyset$. Vice versa, if $K_{\mathrm{loc}} \subseteq L_{\mathrm{loc}}$ satisfies [K0]–[K2], then there exists a locally admissible controller H such that $K_{\mathrm{loc}} = L_{\mathrm{loc}} \cap H$.

Proof. The claim is essentially a special case of the common setup proposed by Ramadge and Wonham (1987), with the further development to avoid deadlocks by Ramadge (1989). For the purpose of this report, we refer to the facts that (i) closed $*$-languages do not conflict, and, (ii) a closed-loop behaviour is relatively closed w.r.t. a closed language if and only if it is itself closed. Then, the claim is identified a special case of Theorem 9 with a proof provided in Section 3. To this end, note that if K_{loc} satisfies [K0]–[K2], then a corresponding admissible controller is given by $H := K_{\mathrm{loc}}\Sigma_{\mathrm{uc}}^*$. □

For the common synthesis problem in this setting, we are given the local plant behaviour $L_{\mathrm{loc}} \subseteq \Sigma^*$ and a safety property $E \subseteq \Sigma^*$, to ask for an admissible controller $H \subseteq \Sigma^*$ that restricts the plant behaviour accordingly, i.e., we require

$L_{\text{loc}} \cap H \subseteq E$. The key observation here is that each of the properties [K0]–[K2] is retained under arbitrary union. Without loss of generality, we assume that $E \subseteq L_{\text{loc}}$ and obtain the supremal achievable closed-loop language

$$K_{\text{loc}}^{\uparrow} := \sup\{\, K_{\text{loc}} \subseteq E \mid \; K_{\text{loc}} \text{ satisfies [K0]–[K2]} \,\} \tag{3}$$

with associated minimal restrictive controller $H^{\uparrow} := K_{\text{loc}}^{\uparrow}\Sigma_{\text{uc}}^{*}$ that enforces the specification E. For regular parameters, a synthesis procedure is provided by Kumar et al. (1992). Effectively, the computations amount to an iteration of the basic synthesis algorithm outlined in Ramadge and Wonham (1987) with the successive removal of deadlocking states.

Example: stack feeder, capacity 2, safety specification. Continuing the stack feeder example, consider the situation where it is regarded unsafe to reload the stack unless it is empty. This could be a constraint imposed by the supply only being capable to deliver two work pieces at the time. The corresponding safety property is generated by the automaton in Figure 2 on the left. For the purpose of control, we regard r the only controllable event; i.e., the physical implementation allows the controller to prevent a reload but it can not stop the environment from triggering a feed. As it turns out, the formal safety specification E obtained by intersecting the local plant behaviour L_{loc} with the safety property is controllable and complete. It therefore matches the supremal achievable closed-loop behaviour, generated by the automaton in Figure 2 on the right.

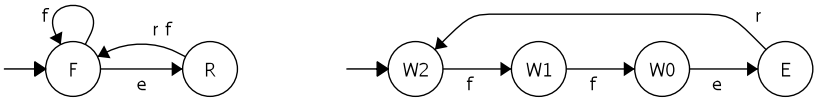

Figure 2: safety specification and local closed-loop behaviour of stack feeder cap. 2

Example: discussion continued regarding non-determinism: Lets assume that the stack feeder is initially loaded with one or two workpieces and that a reload r adds one or two workpieces to the stack. This leads to a non-deterministic representation of the phenomenon as given in Figure 3. By the so called subset-construction we can still represent the local behaviour by a deterministic automaton. However, our assumption on how the phenomenon interacts with the controller are violated: e.g., the string s = fer drives the representation either to state W1 or W2 and the local behaviour suggests that the two successor events feed f and reload r may occur next; if they are both regarded controllable, a controller may disable either single one of them, leading to a deadlock if the actual phenomenon indeed holds two workpieces W2 and only feed f can be executed. Determinisation by subset-construction alone will not resolve this issue, however, the controller "not to know the plant state" can be interpreted as *partial observation*: a formal determinisation by the insertion of additional unobservable events to precede non-deterministic transitions together with the technical requirement that control patterns must not depend on unobservable events, i.e., $H = \mathrm{p}_{\mathrm{o}}^{-1}\mathrm{p}_{\mathrm{o}}\, H$ with the natural projection p_{o} from Σ^{*} to Σ_{o}^{*}. Note that this is the "tame variant" of partial observation where all controllable events are observable, i.e., $\Sigma_{\mathrm{c}} \subseteq \Sigma_{\mathrm{o}}$, and the achievable closed-loop

behaviours are characterised by prefix-normality as an additional property next to [K0]–[K2]. Thus, for the purpose of controller synthesis, non-determinism and partial observation are closely related.

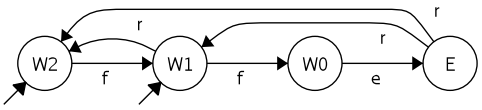

Figure 3: stack feeder of capacity 2 with non-deterministic reload

Preparing the discussion in the subsequent section, we reformulate the synthesis problem in terms of ω-languages. Since the relevant $*$-languages were assumed complete and closed, they can be equivalently characterised by their topologically closed limits: for the local plant behaviour, let $\mathcal{L}_{\mathrm{loc}} := \lim L_{\mathrm{loc}}$ to obtain $L_{\mathrm{loc}} = \mathrm{pre}\,\mathcal{L}_{\mathrm{loc}}$ and $\mathrm{clo}\,\mathcal{L}_{\mathrm{loc}} = \lim \mathrm{pre}\,\mathcal{L}_{\mathrm{loc}} = \mathcal{L}_{\mathrm{loc}}$. Here, we regard the *discrete-event system* $\mathcal{L}_{\mathrm{loc}}$ a formal model of the phenomenon. Likewise, we obtain the topologically closed specification $\mathcal{E} := \lim E$ and ask for a supervisor $\mathcal{H} \subseteq \Sigma^\omega$ with admissibility conditions adapted as follows. [2]

Definition 3. Given an alphabet Σ with uncontrollable events Σ_{uc}, a language $\mathcal{H} \subseteq \Sigma^\omega$ is a *locally admissible controller* for the closed local behaviour $\mathcal{L}_{\mathrm{loc}} \subseteq \Sigma^\omega$ of a non-terminating process, if

[H0] $\mathcal{H}$ is closed,

[H1] $(\mathrm{pre}\,\mathcal{H})\Sigma_{\mathrm{uc}} \subseteq \mathrm{pre}\,\mathcal{H}$, and

[H2] $\mathcal{L}_{\mathrm{loc}}$ and $\mathcal{H}$ are locally non-conflicting. □

If a controller $\mathcal{H} \subseteq \Sigma^\omega$ satisfies the above admissibility criteria, then $\mathrm{pre}\,\mathcal{H}$ is admissible to $\mathrm{pre}\,\mathcal{L}_{\mathrm{loc}}$, too. Vice versa, if $\mathrm{pre}\,\mathcal{H}$ satisfies the admissibility criteria stated for $*$-languages, then $\mathcal{H}$ satisfies [H1] and [H2] of the admissibility criteria stated for ω-languages. Thus, a topologically closed controller candidate $\mathcal{H} \subseteq \Sigma^\omega$ is admissible if and only if its prefix is admissible to $\mathrm{pre}\,\mathcal{L}_{\mathrm{loc}}$.

Theorem 4. Consider an alphabet Σ with uncontrollable events Σ_{uc}. For the closed local behaviour $\mathcal{L}_{\mathrm{loc}} \subseteq \Sigma^\omega$ of a non-terminating process and a locally admissible controller $\mathcal{H} \subseteq \Sigma^\omega$ let $\mathcal{K}_{\mathrm{loc}} = \mathcal{L}_{\mathrm{loc}} \cap \mathcal{H}$. Then

[K0] $\mathcal{K}_{\mathrm{loc}}$ is closed,

[K1] $\mathrm{pre}\,\mathcal{K}_{\mathrm{loc}}$ is controllable w.r.t. $\mathrm{pre}\,\mathcal{L}_{\mathrm{loc}}$,

Moreover, if $\mathcal{L}_{\mathrm{loc}} \neq \emptyset \neq \mathcal{H}$, then $\mathcal{K}_{\mathrm{loc}} \neq \emptyset$. Vice versa, if $\mathcal{K}_{\mathrm{loc}} \subseteq \mathcal{L}_{\mathrm{loc}}$ satisfies

[2] The assumption that L_{loc} and E are complete is not restrictive. If L_{loc} did deadlock, we could formally introduce a locking event to be generated after those strings $s \in L_{\mathrm{loc}}$ that deadlock; for a subsequent controller synthesis the locking event would be regarded uncontrollable and a safety specification would be put into place to forbid the locking event; then the supervisor will controll the plant such that the deadlock is avoided in the closed-loop configuration. If E did deadlock, we could substitute E with its supremal complete sublanguage.

[K0] and [K1], then there exists a locally admissible controller $\mathcal{H}$ such that $\mathcal{K}_{\text{loc}} = \mathcal{L}_{\text{loc}} \cap \mathcal{H}$.

Proof. The claim is essentially covered by the results provided by Ramadge (1989) and the further discussion by Kumar et al. (1992). For the purpose of this report, we refer to the facts that (i) for closed ω-languages not to conflict is equivalent with not to conflict locally, and, (ii) that a closed-loop behaviour is relatively closed w.r.t. a closed language if and only if it is itself closed. Then, the claim is identified a special case of Theorem 11 with a proof provided in Section 3. □

If we now look at the synthesis problem, we may consider the supremum of all sublanguages $\mathcal{K}_{\text{loc}} \subseteq \mathcal{E} \subseteq \mathcal{L}_{\text{loc}}$ that satisfy the above conditions [K0] and [K1]. While in general topological closedness [K0] is not retained under arbitrary union, it is the assumption of a topologically closed specification that allows to recover the unique supremal closed-loop behaviour experienced from the perspective of $*$-languages.

Lemma 5. Consider an alphabet Σ with uncontrollable events Σ_{uc}, a closed local behaviour $\mathcal{L}_{\text{loc}} \subseteq \Sigma^\omega$ of a non-terminating process and a closed safety specification $\mathcal{E} \subseteq \mathcal{L}_{\text{loc}}$. Then the supremum

$$\mathcal{K}_{\text{loc}}^{\uparrow} := \sup\{\, \mathcal{K}_{\text{loc}} \subseteq \mathcal{E} \mid \mathcal{K}_{\text{loc}} \text{ satisfies [K0] and [K1]} \,\} \tag{4}$$

itself satisfies [K0] and [K1], as identified in Theorem 4.

Proof. The claim is a special case of Proposition 3.2 reported by Ramadge (1989). For the purpose of this report, we refer to the proof of Lemma 12, Section 3. □

Given an ω-language $\mathcal{K}_{\text{loc}}$ that satisfies conditions [K0] and [K1], its prefix satisfies the respective conditions [K0]–[K2] for $*$-languages. Vice versa, if a $*$-language satisfies [K0]–[K2], then its limit satisfies the respective conditions [K0] and [K1] for ω-languages. In this sense, both synthesis problems are equivalent. This was expected, since the second problem was constructed as a re-interpretation of the first problem. Since the limit operator is monotone, we can compute the supremal closed-loop behaviour as an ω-language by applying the algorithms developed for $*$-languages on the prefix of the input data $\mathcal{L}_{\text{loc}} \subseteq \Sigma^\omega$ and $\mathcal{E} \subseteq \Sigma^\omega$, both assumed to be topologically closed and regular. Technically, this is done by interpreting deterministic Büchi automata with trivial acceptance condition as ordinary automata that realise the input data $\operatorname{pre}\mathcal{L}_{\text{loc}}$ and $\operatorname{pre}\mathcal{E}$.

2 Synthesis of Liveness Properties

When we drop the requirement of the specification $\mathcal{E}$ to be topologically closed it can be used to represent liveness properties. A prototypical example here is to require that a processing machine after some events that indicate activity eventually issues an event to acknowledge that a particular task has been successfully completed. In this section, we still consider the local behaviour $\mathcal{L}_{\text{loc}}$ as a model of the process, which, as the limit of a prefix closed $*$-language, is topologically closed. Thus, if the process can at all be controlled to eventually report success, this can be achieved within a bounded amount of logic time. Here, eventuality is seen as a useful tool to express generic requirements independent of a particular plant at hand. We can think of the eventually-report-success specification as a building block in a library of specifications which will be imposed together with other requirements on a specific manufacturing process at hand. It is then left to the synthesis procedure to figure whether at all and in which amount of logic time the requirements can be resolved.

Looking at the development of the previous section, we maintain the assumptions imposed on the phenomenon and its interaction with the supervisor. In particular, we still refer to local admissibility, Definition 3, and the consequences reported in Theorem 4. The only technical difference is a topologically not closed specification, and, thus, Lemma 5 does not apply any longer and the supremum $\mathcal{K}_{\text{loc}}^{\uparrow}$ does not allow for an immediate extraction of an according supervisor. This situation is a special case of the study conducted by Thistle and Wonham (1994a) and we reproduce results relevant for our discussion here.

One key contribution of the cited literature is an alternative characterisation of the supremum $\mathcal{K}_{\text{loc}}^{\uparrow}$ based on the following notion of the controllability prefix.

Definition 6. Given an alphabet Σ with uncontrollable events Σ_{uc}, consider a closed local behaviour $\mathcal{L}_{\text{loc}} \subseteq \Sigma^{\omega}$ of a non-terminating process and a sublanguage $\mathcal{K} \subseteq \mathcal{L}_{\text{loc}}$. A string $s \in \text{pre}\,\mathcal{K}$ is in the *controllability prefix* of $\mathcal{K}$ w.r.t. $\mathcal{L}_{\text{loc}}$, if there exists a subset $\mathcal{V}_s \subseteq \mathcal{K}$ with $s \in \text{pre}\,\mathcal{V}_s$ such that

[C0] $\mathcal{V}_s$ is topologically closed, and

[C1] $\text{pre}\,\mathcal{V}_s$ is controllable w.r.t. $\text{pre}\,(\mathcal{L}_{\text{loc}} \cap s\Sigma^{\omega})$.

The controllability prefix of $\mathcal{K}$ is denoted $\text{ctrl}\,\mathcal{K}$. If the controllability prefix equals the ordinary prefix, then $\mathcal{K}$ is said to be *ω-controllable w.r.t.* $\mathcal{L}_{\text{loc}}$. [3] □

To further elaborate the above definition, lets assume that, at some instance

[3] The literature provides different incompatible definitions for ω-controllability. For the specific case of a topologically closed plant, the definition reproduced here is equivalent to the definition proposed by Thistle and Wonham (1994a). It is not equivalent to the definition provided by Ramadge (1989).

of physical time, the phenomenon has generated an event sequence $s \in \operatorname{pre} \mathcal{L}_{\mathrm{loc}}$, regardless whether with or without supervision. If $s \in \operatorname{ctrl} \mathcal{K}$, conditions [C0] and [C1] by Theorem 4 are equivalent with the existence of a supervisor that can control the hypothetical plant $\mathcal{L}_{\mathrm{loc}} \cap s\Sigma^{\omega}$ to evolve within $\mathcal{V}_s \subseteq \mathcal{K}$. Here, the hypothetical plant $\mathcal{L}_{\mathrm{loc}} \cap s\Sigma^{\omega}$ is set up as the original plant model under the additional hypothesis that, for whatever reason, the phenomenon starts by generating the event sequence s. Thus, whenever a string from the controllability prefix has been generated, a controller could take over to operate the plant compliant to eventuality and liveness requirements expressed by $\mathcal{K}$. Thistle and Wonham (1994a) interpret the controllability prefix $\operatorname{ctrl} \mathcal{K}$ as the *winning configurations*. In particular, if $\mathcal{K}$ is non-empty and ω-controllable, the locally admissible controller corresponding to $\mathcal{V}_s$ with $s = \epsilon$ can control the plant $\mathcal{L}_{\mathrm{loc}}$ to satisfy $\mathcal{K}$ unconditionally.

Now consider a string from the controllability prefix $s \in \operatorname{ctrl} \mathcal{K}$ and $\sigma \in \Sigma_{\mathrm{uc}}$ such that $s\sigma \in \operatorname{pre} \mathcal{L}_{\mathrm{loc}}$. Let $\mathcal{V}_s$ denote the associated language $\mathcal{V}_s$ from the above definition. Controllability [C1] implies that $s\sigma \in \operatorname{pre} \mathcal{V}_s$, and, thus $s\sigma \in \operatorname{pre} \mathcal{K}$. In particular, if $\mathcal{K}$ is ω-controllable, then $\operatorname{pre} \mathcal{K}$ is controllable. From a practical perspective, ω-controllability of a candidate closed-loop behaviour $\mathcal{K}$ implies the existence of a controller that operates the plant within $\operatorname{pre} \mathcal{K}$ with the persistent option to activate a more restrictive controller that enforces $\mathcal{K}$.

As it turns out, ω-controllability is not only retained under arbitrary union but also leads to exactly the same supremum as the discussion in the previous section.

Theorem 7. Given an alphabet Σ with uncontrollable events Σ_{uc}, consider a closed local behaviour $\mathcal{L}_{\mathrm{loc}} \subseteq \Sigma^{\omega}$ of a non-terminating process and a not-necessarily closed specification $\mathcal{E} \subseteq \mathcal{L}_{\mathrm{loc}}$. Then

$$\mathcal{K}^{\uparrow} := \sup\{\, \mathcal{K} \subseteq \mathcal{E} \mid \mathcal{K} \text{ is } \omega\text{-controllable w.r.t. } \mathcal{L}_{\mathrm{loc}} \,\} \tag{5}$$

$$= \sup\{\, \mathcal{K}_{\mathrm{loc}} \subseteq \mathcal{E} \mid \mathcal{K}_{\mathrm{loc}} \text{ satisfies [K0] and [K1] from Theorem 4} \,\} \tag{6}$$

itself is ω-controllable w.r.t $\mathcal{L}_{\mathrm{loc}}$.

Proof. This is a special case of the more general situation addressed by Proposition 5.2 and Corollary 5.4 in (Thistle and Wonham, 1994a). We reproduce a proof later on in this report for general case; see Theorem 14. □

For regular input data, the synthesis procedure given by Thistle and Wonham (1994b) computes a realisation of $\mathcal{K}^{\uparrow}$ by iterating on a deterministic automaton that initially generates $\mathcal{L}_{\mathrm{loc}}$ and accepts $\mathcal{E}$ with a Rabin acceptance condition. For easy reference, an outline of the algorithm for the special case of deterministic Büchi automata is given in the appendix of this report, including an indication on how to obtain an admissible controller that enforces the specification.

Example: stack feeder, capacity 2, liveness specification. Lets assume that the stack feeder, Figure 1, eventually needs some maintenance which can only be performed when it is empty, i.e., we specify that regardless the past event sequence

the empty event e shall eventually occur. This is expressed independently of the stack capacity by the ω-language $\mathcal{E}$ accepted by the automaton in Figure 4 on the left. Regarding the reload r controllable, we obtain the supremal ω-controllable sublanguage $\mathcal{K}^{\uparrow}$ accepted by the automaton in Figure 4 on the right. Note that $\mathcal{K}^{\uparrow}$ is not topologically closed and can therefore not be implemented by a locally admissible supervisor: for a physical implementation the liveness requirement to eventually disable reload r to provoke the empty event e needs to be somehow resolved. One possibility is to only enable reload as an immediate successor of the empty event.

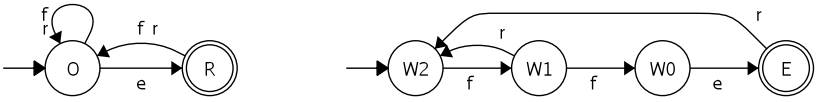

Figure 4: liveness specification and supremal controllable sublanguage

3 When the Plant Exhibits Liveness Properties

We turn to the situation where the plant possesses liveness properties other than not to deadlock. More specifically, we revise the modelling framework summarised in Section 1 to address plants that are representable as the limit of a not-necessarily prefix closed $*$-language. Given the prefix closed and complete local behaviour $L_{\mathrm{loc}} \subseteq \Sigma^*$ of a non-terminating process subject to the assumptions imposed earlier in this report, let $M \subseteq \Sigma^*$ denote the so called *accepted behaviour*. Then, L_{loc} is said *not to livelock* w.r.t. M if every string from the local behaviour can be extended to an accepted string, i.e., if

$$(\forall\, s \in L_{\mathrm{loc}}\, \exists\, t \in \Sigma^*)[\, st \in L_{\mathrm{loc}} \cap M\,]\,. \tag{7}$$

For the sake of a concise notation, we will assume the local behaviour L_{loc} not to livelock w.r.t. a given accepted behaviour M and let $L := L_{\mathrm{loc}} \cap M$. This amounts to $L_{\mathrm{loc}} = \mathrm{pre}\, L$ and, to this end, we can consider the *discrete-event system* L as a formal model of the phenomenon. [4]

As with deadlocks, the technical requirement (7) imposed on the plant model does not suffice to conclude a liveness property of the underlying phenomenon. This is addressed by another hypothesis regarding the non-terminating process.

[4] For the purpose of this report, where we are concerned with controller synthesis, the assumption of the model not to livelock is not restrictive: if L_{loc} did livelock, the conflict could be formally resolved by introducing an explicit locking event to be generated after those strings $s \in L_{\mathrm{loc}}$ that livelocks to attain a formally accepted string; for a subsequent controller synthesis the locking event would be regarded uncontrollable and a safety specification would be put into place to forbid the locking event.

Hypothesis — non-terminating process with accepted behaviour: if, at any instance of physical time, the accepted behaviour indicates that the event sequence generated so far can be extended to an accepted string, then the phenomenon, for any configuration that is compatible with the event sequence generated so far, will eventually generate an accepted string.

Consequence — for a non-livelocking model, an unbounded monotone sequence of accepted strings will be generated during the elapse of an infinite physical duration.

With this hypothesis, the limit $\mathcal{L} := \lim L$ corresponds to the set of infinite signals that can be possibly generated during the elapse of infinite physical time. Regarding the internal configuration, we again refer to a deterministic automaton realisation. Then, the above assumption amounts to transitions being executed such that an accepted configuration is eventually attained. This assumption goes further than the one made in Section 1 to address not to deadlock: the phenomenon must include some mechanism that decides which transitions are to be executed in order to eventually reach a marked state.

Example: generic stack feeder with finite capacity. The *generic stack feeder* is obtained as an abstraction of the specific stack feeder with some unknown but finite capacity. Technically, we take the union over automata realisations for each specific capacity, modelled in analogy to the capacity two example in Figure 1. This leads to an infinite state set and to an unknown initial state. Both are addressed by merging states according to a simulation relation, with the resulting abstraction given in Figure 5. By construction, the model is a suitable basis for the verification of properties that are stated as an upper bound on the behaviour, including liveness. In particular, any sequence of successive feed events f will be eventually followed by either r or e, and this is indeed the case for any specific instance of the generic stack feeder [5]. The question of whether the model is also a suitable basis for controller synthesis is addressed further below; see also Section 5.

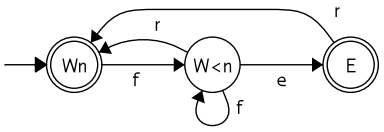

Figure 5: generic stack feeder with finite capacity

.

For the purpose of supervisory control we employ the same interface as motivated in the previous sections, i.e., a supervisor represented as a closed language to generate control patterns from the past event sequence as feed back to disable controllable events. As an additional requirement, the supervisor shall not prevent

[5] Volksweisheit: “Jedes Gefäß das leckt ist irgendwann einmal leer”.

the plant from attaining an accepted state and we adapt the admissibility criteria from Section 1 accordingly; see below condition [H3].

Definition 8. Given an alphabet Σ with uncontrollable events Σ_{uc}, a language H is an *admissible controller* for the accepted behaviour $L \subseteq \Sigma^*$ of a non-terminating process, if it is a locally admissible controller for $L_{\text{loc}} := \operatorname{pre} L$ and if in addition

[H3] L and H are non-conflicting, i.e., $\operatorname{pre}(L \cap H) = (\operatorname{pre} L) \cap (\operatorname{pre} H)$. □

Admissibility as defined above technically matches the original setting proposed by Ramadge and Wonham (1987) with the further development for non-terminating processes by Ramadge (1989). In particular, we recover exactly the same characterisation of achievable closed-loop behaviours as originally identified by Ramadge (1989).

Theorem 9. Consider an alphabet Σ with uncontrollable events Σ_{uc}. For the accepted behaviour $L \subseteq \Sigma^*$ of a non-terminating process an admissible controller $H \subseteq \Sigma^*$ let $K = L \cap H$. Then

[K0'] K is relatively prefix-closed w.r.t. L,

[K1] K is controllable w.r.t. L,

[K2] K is complete.

Moreover, if $L \neq \emptyset \neq H$, then $K \neq \emptyset$. Vice versa, if $K \subseteq L$ satisfies [K0'], [K1] and [K2], then there exists an admissible controller H such that $K = L \cap H$.

Proof. Although the claim is essentially covered by the results from Ramadge (1989), we provide a direct proof for documentation purposes. We begin with "[H*]⇒[K*]". Referring to closedness [H0], K is defined by an intersection of a prefixed closed language with L. This implies relative closedness [K0']. Referring to [H3], we observe $\operatorname{pre} K = (\operatorname{pre} L) \cap (\operatorname{pre} H)$ and conclude completeness [K2] by completeness [H2]. Referring first to [H1] and then to [H3], we obtain $((\operatorname{pre} K)\Sigma_{\text{uc}}) \cap (\operatorname{pre} L) \subseteq (H\Sigma_{\text{uc}}) \cap (\operatorname{pre} L) \subseteq H \cap (\operatorname{pre} L) = \operatorname{pre}(L \cap H) = \operatorname{pre} K$, which amounts to controllability [K1]. The reverse direction "[K*]⇒[H*]" is established constructively with the candidate $H := (\operatorname{pre} K)\Sigma_{\text{uc}}^*$, to immediately imply closedness [H0] and controllability [H1]. By relative closedness [K0], we have $K \subseteq L$ to establish $\operatorname{pre} K \subseteq (\operatorname{pre} L) \cap H$. For the converse inclusion, pick an arbitrary $s \in (\operatorname{pre} L) \cap H$. By the choice of the candidate H, we can rewrite $s = rt$ with $r \in \operatorname{pre} K$ and $t \in \Sigma_{\text{uc}}^*$. If $t = \epsilon$, we conclude $s \in \operatorname{pre} K$. Otherwise, let $\sigma \leq t$ denote the first symbol of t. Referring to controllability [K1], we obtain $r\sigma \in \operatorname{pre} K$. Repreating this argument for the remaining symbols of t, this implies $s = rt \in \operatorname{pre} K$. Since s was chosen arbitrarily, we conclude

$$\operatorname{pre} K = (\operatorname{pre} L) \cap (\operatorname{pre} H)\,. \tag{8}$$

With Eq. (8), completeness [H2] is implied by [K2]. Again referring to Eq. (8), now together with [K0], we obtain $K = (\operatorname{pre} K) \cap L = (\operatorname{pre} L) \cap H \cap L = L \cap H$, which by Eq. (8) also establishes [H3]. □

All closed-loop properties [K0'], [K1] and [K2] are retained under arbitrary union. For a given upper bound $E \subseteq L$, the supremal achievable closed-loop behaviour $K^{\uparrow} \subseteq E$ exists uniquely and can be used to extract a minimal restrictive controller $H^{\uparrow} := (\mathrm{pre}\, K^{\uparrow})\Sigma_{\mathrm{uc}}^{*}$. By [K0'], we can replace E by its supremal relatively closed sublanguage without to affect the solutions of the synthesis problem. In other words, the objective to restrict L to satisfy E can be equivalently expressed by a closed upper bound imposed on the local behaviour $L_{\mathrm{loc}} := \mathrm{pre}\, L$. Thus, although L and E are not closed, the overall situation can still be interpreted as the synthesis of safety properties as opposed to the synthesis of liveness propertries.

In analogy to the basic case, Section 1, we derive the equivalent synthesis problem for ω-languages.

Definition 10. Given an alphabet Σ with uncontrollable events Σ_{uc}, a language $\mathcal{H} \subseteq \Sigma^{\omega}$ is an *admissible controller* for the not-necessarily closed model $\mathcal{L} \subseteq \Sigma^{\omega}$ of a non-terminating process, if

[H0] $\mathcal{H}$ is closed,

[H1] $(\mathrm{pre}\,\mathcal{H})\Sigma_{\mathrm{uc}} \subseteq \mathrm{pre}\,\mathcal{H}$, and

[H2'] $\mathcal{L}$ and $\mathcal{H}$ are non-conflicting. □

Note that the stronger condition [H2'] implies local non-conflictingness [H2].

Theorem 11. Consider an alphabet Σ with uncontrollable events Σ_{uc}. For the not-necessarily closed model $\mathcal{L} \subseteq \Sigma^{\omega}$ of a non-terminating process with admissible controller $\mathcal{H} \subseteq \Sigma^{\omega}$ let $\mathcal{K} = \mathcal{L} \cap \mathcal{H}$. Then

[K0'] $\mathcal{K}$ is relatively topologically closed w.r.t. $\mathcal{L}$.

[K1] $\mathrm{pre}\,\mathcal{K}$ is controllable w.r.t. $\mathrm{pre}\,\mathcal{L}$,

Moreover, if $\mathcal{L} \neq \emptyset \neq \mathcal{H}$, then $\mathcal{K} \neq \emptyset$. Vice versa, if $\mathcal{K}$ satisfies [K0'] and [K1], then there exists a locally admissible controller $\mathcal{H}$ such that $\mathcal{K} = \mathcal{L} \cap \mathcal{H}$.

Proof. The claim is covered by the results from (Ramadge, 1989). For a direct proof of "[H*]⇒[K*]", recall that the intersection of a topologically closed language with another language is relatively topologically closed to obtain [K0'] by [H0]. Regarding [K1], we first observe that $\mathrm{pre}\,\mathcal{K} = \mathrm{pre}\,(\mathcal{L} \cap \mathcal{H}) = (\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H})$, where the last equality is by [H2']. Then, by [H1], we obtain $((\mathrm{pre}\,\mathcal{K})\Sigma_{\mathrm{uc}}) \cap (\mathrm{pre}\,\mathcal{L}) \subseteq ((\mathrm{pre}\,\mathcal{H})\Sigma_{\mathrm{uc}}) \cap (\mathrm{pre}\,\mathcal{L}) \subseteq (\mathrm{pre}\,\mathcal{H}) \cap (\mathrm{pre}\,\mathcal{L}) = \mathrm{pre}\,\mathcal{K}$ to conclude [K1]. The reverse direction "[K*]⇒[H*]" is established constructively with the candidate $\mathcal{H} := \lim((\mathrm{pre}\,\mathcal{K})\Sigma_{\mathrm{uc}}^{*})$, to immediately imply [H0] and [H1]. Note also that $\mathrm{pre}\,\mathcal{K} \subseteq (\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H})$. To establish the preliminary result

$$\mathrm{pre}\,\mathcal{K} = (\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H})\,, \tag{9}$$

pick an arbitrary $s \in (\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H})$. By the choice of the candidate $\mathcal{H}$, we can rewrite $s = rt$ with $r \in \mathrm{pre}\,\mathcal{K}$ and $t \in \Sigma_{\mathrm{uc}}^{*}$. If $t = \epsilon$, we conclude $s \in \mathrm{pre}\,\mathcal{K}$.

Otherwise, let $\sigma \leq t$ denote the first symbol of t. Referring to controllability [K1], we obtain $r\sigma \in \operatorname{pre}\mathcal{K}$ and repeating this argument for the remaining symbols of t this implies $s = rt \in \operatorname{pre}\mathcal{K}$. Since s was arbitrary, we conclude $(\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\mathcal{H}) \subseteq \operatorname{pre}\mathcal{K}$ and, hence, conclude the proof of Eq. (9). Taking limits we obtaine $\operatorname{clo}\mathcal{K} \subseteq (\operatorname{clo}\mathcal{L}) \cap (\operatorname{clo}\mathcal{H})$. For the converse inclusion, pick any w from the righthandsside to observe $\operatorname{pre} w \subseteq (\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\mathcal{H})$, and, by Eq. (9), $w = \lim \operatorname{pre} w \subseteq \lim \operatorname{pre}\mathcal{K} = \operatorname{clo}\mathcal{K}$, and we conclude

$$\operatorname{clo}\mathcal{K} = (\operatorname{clo}\mathcal{L}) \cap (\operatorname{clo}\mathcal{H})\,. \tag{10}$$

Referring to relative topologically closedness [K0'] we obtain $\mathcal{K} = (\operatorname{clo}\mathcal{K}) \cap \mathcal{L} = (\operatorname{clo}\mathcal{L}) \cap (\operatorname{clo}\mathcal{H}) \cap \mathcal{L} = \mathcal{L} \cap \mathcal{H}$. This also implies [H2']. □

Given an achievable closed-loop behaviour as language $K \subseteq L \subseteq \Sigma^*$, the limit $\mathcal{K} = \lim K$ is verified to satisfy the above conditions for $\mathcal{L} = \lim L$. Vice versa, if an ω-language $\mathcal{K}$ satisfies the above conditions for $\mathcal{L} = \lim L$, then $(\operatorname{pre}\mathcal{K}) \cap L$ is an achievable closed-loop behaviour for L. Therefore, the two perspectives are identified as effectively equivalent for plants that can be represented as the limit of a $*$-language. Regarding the synthesis problem for a given specification $\mathcal{E} = \lim E$, recall that we may assume E to be relatively prefix closed w.r.t. L without affecting the supremal closed-loop behaviour in the $*$-language setting. For the respective ω-languages, this implies that $\mathcal{E}$ is relatively topologically closed w.r.t. $\mathcal{L}$. Under this condition the supremal solution to the synthesis problem in terms of ω-languages is known to uniquely exist.

Lemma 12. Consider an alphabet Σ with uncontrollable events Σ_{uc}, a not necessarily closed model $\mathcal{L} \subseteq \Sigma^\omega$ of a non-terminating process and a specification $\mathcal{E} \subseteq \mathcal{L}$ that is relatively closed w.r.t. $\mathcal{L}$. Then the supremum

$$\mathcal{K}^\uparrow := \sup\{\, \mathcal{K} \subseteq \mathcal{E} \mid \mathcal{K} \text{ satisfies [K0'] and [K1]} \,\} \tag{11}$$

itself satisfies [K0'] and [K1], as identified in Theorem 11.

Proof. The claim amounts to Proposition 3.2 from (Ramadge, 1989). We provide a self contained proof for documentation purposes. The supremum $\mathcal{K}^\uparrow$ itself is well defined and we consider the relatively closed superset $\mathcal{M} := (\operatorname{clo}\mathcal{K}^\uparrow) \cap \mathcal{L}$. Note that the above construction does not affect the prefix, i.e., $\operatorname{pre}\mathcal{M} = \operatorname{pre}\mathcal{K}^\uparrow$. In particular, $\mathcal{M}$ satisfies [K0] and [K1]. Moreover, from $\mathcal{K}^\uparrow \subseteq \mathcal{E}$ we obtain by monotonicity $\mathcal{M} \subseteq (\operatorname{clo}\mathcal{E}) \cap \mathcal{L}$ and, by relative closedness, $\mathcal{M} \subseteq \mathcal{E}$. Then, supremality of $\mathcal{K}^\uparrow$ implies $\mathcal{M} \subseteq \mathcal{K}^\uparrow$. The converse inclusion holds by construction of $\mathcal{M}$ and we conclude equality $\mathcal{K}^\uparrow = \mathcal{M}$. □

We are now in the position to interpret the liveness properties of the closed-loop configuration. For the local closed-loop behaviour $K_{\text{loc}} := (\operatorname{pre} L) \cap (\operatorname{pre} H)$, we refer to the discussion in Section 1. The additional condition [H3] amounts to $K_{\text{loc}} = \operatorname{pre} K$ and, therefore K_{loc} does not livelock w.r.t. the accepted behaviour

K. Thus, we may consider K an adequate model of the closed-loop configuration. Here, the same line of thought as for the model L of the non-terminating process is applicable. However, the initial hypothesis regarding liveness of the non-terminating process does not automatically imply that during infinite time an unbounded sequence of accepted strings will be generated in a physical implementation of the closed-loop configuration. This is because the non-terminating process may decide in advance to generate events in a way that imposes restrictions on controllable events that must be eventually enabled to attain an accepted string. At the same time the controller may implement a particular control strategy that is incompatible with the a-priory choice taken by the plant. Without referring to additional technical requirements, this issue can only be resolved by assuming some sort of cooperation. Note the fundamental difference to the situation reviewed in the previous Section 2, where we can extract a closed controller that will resolve liveness requirements imposed by the specification for a closed plant.

Example: need for cooperation. The liveness properties encoded in the automaton Figure 6 require the controller to eventually enable a after A or b after B, i.e., the controller is meant to eventually agree with the plant by enabling the lower case event that matches the respective upper case predecessor. Now consider a controller that never enables a, e.g., because a safety specification regards a as unsafe. To attain an accepted string, the plant must eventually generate B. However, to eventually generate B is not a liveness property encoded in the plant model. Moreover, an alternative controller designed for some other specification may disable b just when the plant decided to generate B. If, on the other hand, the plant knows the control strategy, it can cooperate and eventually generate either A or B as required. An alternative approach to resolve this issue is to require the controller to cooperate. However, the ability of the controller to cooperate is not implied by the technical admissibility conditions under consideration so far.

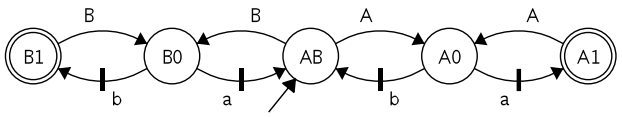

Figure 6: liveness properties that suggest cooperative supervision.

Hypothesis — cooperative supervision of a non-terminating process with accepted behaviour: if there is the persistent option for the model and the supervisor to agree on how to attain an accepted string, then the phenomenon and the implementation of the supervisor will eventually do so.

Consequence — under admissible control, an unbounded monotone sequence of accepted strings will be generated during the elapse of an infinite physical duration.

Example: generic stack feeder with finite capacity and safety specification. The abstract model of the stack feeder from Figure 5 is also suitable for the synthesis of supervisors that enforce safety properties, however, care must be taken regarding liveness. Even if the interface to the actual phenomenon supports that the events f and r are controllable, the controller must not disable both events in state W<n because this will deadlock an actual stack feeder that could be in state W1. In other words, the generic stack feeder as a model of some particular stack feeder does not comply with the hypothesis used in this section: the model when in state W<n indicates the ability to generate e but the actual phenomenon may be in a configuration in which it can not agree to generate e. For the particular example there are two options to circumvent this issue. First, we can consider f as uncontrollable, interpreting the feed as initiated by the plant. Then, for the safety specification not to reload unless the stack is empty the supremal closed-loop behaviour is obtained from the plant behaviour by disabling r in state W<n. The corresponding controller is not only admissible to the generic stack feeder, but also to any particular instance. Second, considering both events f and r controllable, we can refine the generic stack feeder to include the original states W1 and W0 with an additional unobservable transition from W<n to W1; Figure 7. Then, a supervisor designed for partial observation can not distinguish the states W<n and W1 and, hence, will in neither state disable f and r simultaneously. Both strategies are based on further inspection of how the specific instance of a stack feeder relates to its abstraction, the generic stack feeder. We come back to this point in Section 5.

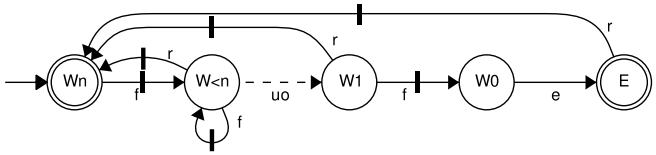

Figure 7: generic stack feeder with finite capacity and unobservable event uo.

4 The General Case

For the most general situation reviewed in this report we consider a plant that exhibits liveness properties and allow for the specification to impose further liveness conditions on the closed-loop behaviour. A comprehensive study of this situation has been presented by Thistle and Wonham (1992, 1994a,b) and we have already recovered the main results for a closed plant, Section 2, as a systematic consequence of the basic case, Section 1, by dropping the requirement of a closed specification. The general case considered in the present section can be obtained similarly, but now starting with the setting of the previous section where the plant is not necessarily closed and by dropping the requirement of a relatively closed specification.

Technically, we still insist in admissibility, Definition 10, and thus are bound to the consequences reported in Theorem 11, i.e., the closed loop is relatively closed [K0'] with a controllable prefix [K1]. However, dropping the requirement of a relatively closed specification, Lemma 12 does not apply any longer and the supremum over all achievable closed loop behaviours $\mathcal{K}^{\uparrow}$ itself may not be achievable because for ω-languages relative closedness is not retained under arbitrary union. As with the special case of a closed plant, an alternative characterisation of $\mathcal{K}^{\uparrow}$ provides additional insight to the situation at hand. To this end, we extend the scope of Definition 6 to define a notion of ω-controllability for not necessarily closed plants.

Definition 13. Given an alphabet Σ with uncontrollable events Σ_{uc}, consider the behaviour $\mathcal{L} \subseteq \Sigma^{\omega}$ of a non-terminating process and a sublanguage $\mathcal{K} \subseteq \mathcal{L}$. A string $s \in \operatorname{pre}\mathcal{K}$ is in the *controllability prefix* of $\mathcal{K}$ w.r.t. $\mathcal{L}$, if there exists a subset $\mathcal{V}_s \subseteq \mathcal{K}$ with $s \in \operatorname{pre}\mathcal{V}_s$ such that

[C0'] $\mathcal{V}_s$ is relatively topologically closed w.r.t. $\mathcal{L}$, and

[C1] $\operatorname{pre}\mathcal{V}_s$ is controllable w.r.t. $\operatorname{pre}(\mathcal{L} \cap s\Sigma^{\omega})$.

The controllability prefix of $\mathcal{K}$ is denoted $\operatorname{ctrl}\mathcal{K}$. If the controllability prefix equals the ordinary prefix, then $\mathcal{K}$ is said to be *ω-controllable w.r.t. $\mathcal{L}$*. □

Except for relative closedness in [C0'], this definition literally matches Definition 6, Section 2, stated for closed plant behaviours. Since relative closedness w.r.t. a closed behaviour is equivalent to closedness, the above definition is technically consistent with the variant Definition 6. It also has the same interpretation: once a string within the controllability prefix is attained, we refer to $\mathcal{V}_s \subseteq \mathcal{K}$ as a closed-loop candidate for the future behaviour. Then, [C0'] and [C1] match the closed-loop characterisation by [K0'] and [K1] and there exists a controller that implements $\mathcal{V}_s$ from s onwards. In particular, this resolves any additional liveness properties expressed by $\mathcal{K}$. Again, ω-controllability implies a controllable prefix. Thus, if $\mathcal{K}$ is ω-controllable, then there exists a controller that operates the plant within $\operatorname{clo}\mathcal{K}$ with the persistent option to activate a more restrictive controller that enforces $\mathcal{K}$.

The following theorem summarises the main result from (Thistle and Wonham, 1994a), Section 5, for the purpose of this report.

Theorem 14. Given an alphabet Σ with uncontrollable events Σ_{uc}, consider the behaviour $\mathcal{L} \subseteq \Sigma^{\omega}$ of a non-terminating process and a specification $\mathcal{E} \subseteq \mathcal{L}$. Then

$$\mathcal{K}^{\uparrow} := \sup\{\, \mathcal{K} \subseteq \mathcal{E} \mid \mathcal{K} \text{ is } \omega\text{-controllable w.r.t. } \mathcal{L} \,\} \tag{12}$$

$$= \sup\{\, \mathcal{K} \subseteq \mathcal{E} \mid \mathcal{K} \text{ satisfies [K0'] and [K1] from Theorem 11} \,\} \tag{13}$$

itself is ω-controllable w.r.t $\mathcal{L}$.

Proof. The claim follows as an immediate consequences from Proposition 5.2 and Corollary 5.4 in (Thistle and Wonham, 1994a). The original literature provides a

proof via various propositions that explicitly address further aspects of the situation at hand. We give a concise direct proof for documentation purposes. Consider a family $(\mathcal{K}_a)_{a\in A}$ of ω-controllable languages and denote the union $\mathcal{K} := \cup_{a\in A}\mathcal{K}_a$. To establish ω-controllability of $\mathcal{K}$, pick an arbitrary $s \in \operatorname{pre}\mathcal{K}$. Since the prefix operator and union commutes, we can pick $a \in A$ such that $s \in \operatorname{pre}\mathcal{K}_a$. By ω-controllability of $\mathcal{K}_a$ we have $s \in \operatorname{ctrl}\mathcal{K}_a$ and choose $\mathcal{V}_s \subseteq \mathcal{K}_a$ with $s \in \operatorname{pre}\mathcal{V}_s$ and according to requirements [C0'] and [C1]. In particular, we have $\mathcal{V}_s \subseteq \mathcal{K}$ and can use the same language to $\mathcal{V}_s$ establish $s \in \operatorname{ctrl}\mathcal{K}$. Since s was chosen arbitrarily, we obtain $\operatorname{pre}\mathcal{K} = \operatorname{ctrl}\mathcal{K}$ to conclude that the union $\mathcal{K}$ is also ω-controllable. For the equality of the two suprema, pick any $\mathcal{K}$ that satisfies [K0'] and [K1] from Theorem 11. By the uniform choice of $\mathcal{V}_s = \mathcal{K}$ we satisfy [C0'] and [C1] for any $s \in \operatorname{pre}\mathcal{K}$. Therefore, $\mathcal{K}$ is ω-controllable. This implies that the second supremum (13) is a subset of the first supremum (12). To establish the converse inclusion, pick an arbitrary infinite string w from the supremum (12). It remains to show that there exists a closed-loop behaviour $\mathcal{K}$ that satisfies [K0'] and [K1] and that includes w. We do so by construction. For each prefix $s < w$, denote $\mathcal{V}_s \subseteq \mathcal{K}$ the language that satisfies $s \in \operatorname{pre}\mathcal{V}_s$, [C0'] and [C1]. Let

$$\mathcal{K}_s := \{v \in \mathcal{V}_s |\, \exists\, \sigma \in \Sigma : s\sigma < v \text{ and } s\sigma \not< w\} \tag{14}$$

and consider the candidate $\mathcal{K} := (\cup_{s<w}\mathcal{K}_s) \cup \{w\}$. Clearly, we have $w \in \mathcal{K}$ and are left to establish relative closedness [K0'] and controllability [K1]. Observe that by our construction the latter union is disjoint and that, for $v \in \mathcal{K}$, $v \neq w$ we have $v \in \mathcal{K}_r$, where $r < v$ is the maximal prefix with $r < w$. Thus, any monotone unbounded sequence in $\operatorname{pre}\mathcal{K}$ has either the limit w or all prefixes in exactly one component $\operatorname{pre}\mathcal{K}_s$. This can be used to establish relative closedness [K0']. Pick any $v \in (\operatorname{clo}\mathcal{K}) \cap \mathcal{L}$; if $v = w$ we trivially have $v \in \mathcal{K}$; else, pick s such $\operatorname{pre} v \subseteq \operatorname{pre}\mathcal{K}_s$. This implies $v \in \operatorname{clo}\mathcal{K}_s$; hence $v \in (\operatorname{clo}\mathcal{K}_s) \cap \mathcal{L} = \mathcal{K}_s \subseteq \mathcal{K}$. This concludes the proof of [K0'] and we turn to controllability [K1]. Pick $s \in \operatorname{pre}\mathcal{K}$ and $\sigma \in \Sigma_{\mathrm{uc}}$ with $s\sigma \in \operatorname{pre}\mathcal{L}$. If $s\sigma < w$, then $w \in \mathcal{K}$ implies $s\sigma \in \operatorname{pre}K$. If $s\sigma \not< w$, we pick the maximal prefix $r < s\sigma$ with $r < w$. Observe that $s \in \operatorname{pre}\mathcal{K}_r \subseteq \operatorname{pre}\mathcal{V}_r$, and, by controllability [C1] of $\operatorname{pre}\mathcal{V}_r$, $s\sigma \in \operatorname{pre}\mathcal{V}_r$. The latter together with $r < s\sigma$ implies $s\sigma \in \operatorname{pre}\mathcal{K}_r$. This concludes the proof of controllability of [K1]. □

A procedure for the computation of the controllability prefix for regular input data is given in Thistle and Wonham (1992), Section 8., with subsequent transformations to obtain the supremal ω-controllable sublanguage $\mathcal{K}^{\uparrow}$. It is based on a realisation of $\mathcal{L}$ and $\mathcal{E}$ by a single deterministic automaton with a Büchi acceptance condition to represent $\mathcal{L}$ and a Rabin acceptance condition for $\mathcal{E}$. For illustration purposes, we give a concise outline in the appendix of this report, restricted to the special case where both $\mathcal{L}$ and $\mathcal{E}$ are represented by one deterministic Büchi automata each.

Example: generic stack feeder with finite capacity and liveness specification. We refer to the abstract model of the stack feeder from Figure 5 with r the only controllable event and ask for an admissible supervisor that eventually empties the stack; see Figure 8 on the left for a realisation of the specification. The abstract

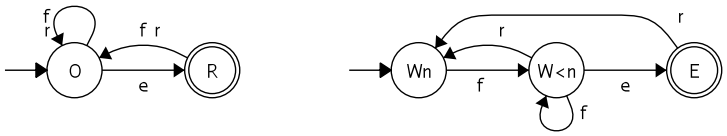

Figure 8: liveness specification and supremal ω-controllable sublanguage.

stack feeder model includes the information that when the reload is disabled, the stack well indeed become eventually empty. This is relevant for the synthesis task at hand: if we used the closure of the behaviour of the generic stack feeder as a model, no supervisor would be be found. The supremal ω-controllable sublanguage for the given input data as realised by the automaton in Figure 8 on the right. This matches the plant model expect that the marking in the initial state has been removed. The interpretation is that an admissible controller must eventually disable the reload r to enforce that after a finite number of further feed events f the stack indeed become empty. As before, one option to achieve this is by disabling r after the first feed, however, one could also do so after an arbitrary finite number of feeds.

Example: technical example to discriminate the two variants. For the purpose of illustration, we introduce the variation of the generic stack feeder given in Figure 9 with the additional capability to emit an arbitrary finite number of empty events e when in state E until the reload r will occur. This can be motivated by periodic sensor readings and an unspecified finite physical time for the reload mechanism to become ready for operation. Regarding the reload r conrollable, the generic plant model $\mathcal{L}$ now requires that r must be enabled eventually. However, a controller designed for the local behaviour $\mathcal{L}_{\text{loc}} := \text{clo}\,\mathcal{L}$ may insist in not to do so and control the substitute plant to exhibit an infinite sequence of e events. This violates the assumption $\mathcal{A} = \mathcal{L}$ and therefore formally satisfies any implied guarantee. Clearly, such a controller is not admissible to $\mathcal{L}$ and, hence, is guaranteed to not be the outcome when applying the methods from this section directly to the generic plant $\mathcal{L}$.

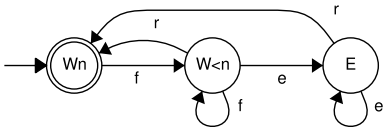

Figure 9: variation of the generic stack feeder

.

5 Abstraction Based Synthesis

In our review of the situation in which the plant exhibits liveness properties other than not to deadlock, Section 3, we imposed the hypothesis that the phenomenon will somehow generate an unbounded monotone sequence of strings accepted by the model. Thinking of a physical implementation as a transition system, there must be some additional mechanism that actually decides on which transitions are taken and this decision must somehow resolve the requirement to eventually attain accepted strings, with *hybrid systems* as a prominent example; see Henzinger (1996); Tabuada (2009). For the closed-loop configuration, we imposed the additional hypothesis that the decision mechanism will cooperate with any control strategy and that it therefore is sufficient to require that a supervisor leaves the persistent chance to attain an accepted string; see also the technical condition [H3] for $*$-languages and [H2'] for ω-languages, respectively. However, cooperation may in this form not be a satisfactory concept for an application at hand. Considering the generic stack feeder, one may encounter situations in which an abstract plant model effectively decides by the choice of a particular instance how to resolve liveness properties befor any events are generated and, thus, leaving no degrees of freedom for cooperative behaviour. Rather than a persistent chance to resolve liveness, this suggests to require the controller to have no chance to conflict, regardless how the plant attempts to attain accepted strings.

To avoid the need for cooperation we may impose an alternative hypothesis on the non-terminating process and discuss corresponding formal properties of the model.

> *Hypothesis — supervised non-terminating process with accepted behaviour: any a-priory choice of events to be generated by the phenomenon that is consistent with the accepted behaviour, must not restrict the future applicability of control patterns in order to attain an accepted string.*
>
> *Consequence — under admissible control, an unbounded monotone sequence of accepted strings will be generated during the elapse of an infinite physical duration.*

The following lemma directly addresses the situation in which the plant resolves *all* liveness properties by an a-priory choice of events to generate. Here, the phenomenon can be represented as a closed language and controller synthesis can be based on a not necessarily closed abstraction; see also Bai and Moor (2017), which also addresses a converse implication.

Lemma 15. Given a plant behaviour $\mathcal{L}$ and an abstraction $\mathcal{L}'$, $\mathcal{L} \subseteq \mathcal{L}'$, subject to the consistency conditions

[C1] consistent admissible control patterns that do not deadlock,

$$(\forall\, s \in \operatorname{pre}\mathcal{L})[\,(s\Sigma_{\mathrm{uc}}) \cap \operatorname{pre}\mathcal{L} = \emptyset \;\;\Rightarrow\;\; (s\Sigma) \cap \operatorname{pre}\mathcal{L} = (s\Sigma) \cap \operatorname{pre}\mathcal{L}'\,],$$

[C2] absence of liveness properties other than not to deadlock,

$\mathcal{L}$ is closed,

then admissibility of a controller $\mathcal{H}$ to the abstraction $\mathcal{L}'$ implies admissibility of $\mathcal{H}$ to the actual plant $\mathcal{L}$.

Proof. The only admissibility condition that depends on the plant behaviour is [H2']. Pick an arbitrary $s \in (\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\mathcal{H})$. If there exists an event $\sigma \in \Sigma_{\mathrm{uc}}$ such that $s\sigma \in \operatorname{pre}\mathcal{L}$ we obtain $s\sigma \in \operatorname{pre}\mathcal{H}$ by controllability [H1]. Else, we have $(s\Sigma_{\mathrm{uc}}) \cap \operatorname{pre}\mathcal{L} = \emptyset$ and appeal to [C1] to obtain $(s\Sigma) \cap \operatorname{pre}\mathcal{L} = (s\Sigma) \cap \operatorname{pre}\mathcal{L}'$. By non-conflictingness [H2'] with the abstraction $\mathcal{L}'$ we can again choose $\sigma \in \Sigma$ such that $s\sigma \in (\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\mathcal{H})$. To this end, we have extended an arbitrary string from the local closed-loop behaviour by one more event. Thus, $\mathcal{L}$ and $\mathcal{H}$ are locally non-conflicting and $\mathcal{H}$ is locally admissible to $\mathcal{L}$. Referring to [C2] and [H0] both languages are closed. Thus, non-conflictingness is implied by local non-conflictingness. □

Condition [C1] requires that, whenever the actual plant $\mathcal{L}$ suggests that at least one controllable event has to be enabled, this is also the case for the abstraction $\mathcal{L}'$ and that whatever control pattern the controller may choose, not to deadlock the abstraction implies not to deadlock the actual plant. By closedness [C2], a non-conflicting closed-loop configuration is obtained.

Example: generic stack feeder with finite capacity. We interpret the model from Figure 5 as abstraction $\mathcal{L}'$ of all particular instance $\mathcal{L}_{\mathrm{n}}$ with capacity $n \in \mathbb{N}$; i.e., $\cup_{n\in\mathbb{N}}\mathcal{L}_{\mathrm{n}} \subseteq \mathcal{L}'$. Clearly, if we pick a particular instance $\mathcal{L}_{\mathrm{n}}$ we have $\mathcal{L}_{\mathrm{n}} \subseteq \mathcal{L}'$ and $\mathcal{L}_{\mathrm{n}}$ is closed [C2]. If in addition r is the only controllable event, [C1] is satisfied. Then, any controller designed to be admissible to the abstraction $\mathcal{L}'$ is also admissible to any particular stack feeder $\mathcal{L}_{\mathrm{n}}$. If, on the other hand, r and f are controllable, then [C1] is violated.

The following weaker requirement [A2] is proposed by Moor et al. (2011) to replace closedness [C2] in the context of alternating inputs and outputs.

Lemma 16. Given a plant behaviour $\mathcal{L}$ and an abstraction $\mathcal{L}'$, $\mathcal{L} \subseteq \mathcal{L}' \subseteq (UY)^{\omega}$, subject to the alternating-input-output conditions

[A1] locally free input,

$$(\forall\, s \in \operatorname{pre}\mathcal{L},\ \mu, \mu' \in U\,)[\; s\mu \in \operatorname{pre}\mathcal{L} \;\;\Rightarrow\;\; s\mu' \in \operatorname{pre}\mathcal{L}\,],$$

[A2] strong non-anticipation,

$\mathcal{L}$ is ω-controllable w.r.t. clo $\mathcal{L}$ with Y taking the role of the uncontrollable (!) events.

then admissibility of a controller $\mathcal{H}$ to the abstraction $\mathcal{L}'$ implies admissibility of $\mathcal{H}$ to the actual plant $\mathcal{L}$.

Proof. The claim is from (Moor et al., 2011), Proposition 13, however, stated there without a proof. Although the author advertises a technical report that provides a proof, we reproduce it here for easy reference. The only admissibility condition that depends on the plant behaviour is [H2']. Pick an arbitrary $s \in (\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H})$. According to Theorem 7 and condition [A2] $\mathcal{L}$ can be represented as a union $\mathcal{L} = \cup_{a\in A}\mathcal{L}_\mathrm{a}$, where the components $\mathcal{L}_\mathrm{a}$ are closed and have an controllable prefix with U regarded uncontrollable. In particular, the locally free input of $\mathcal{L}$ implies that each union component has also a locally free input. Since the prefix operator distributes over arbitrary unions, we can pick an $a \in A$ with $s \in \mathrm{pre}\,\mathcal{L}_\mathrm{a}$. Now assume that we have extended s by $t \in \Sigma^*$ such that $st \in (\mathrm{pre}\,\mathcal{L}_\mathrm{a}) \cap (\mathrm{pre}\,\mathcal{H})$. If there exists an event $\nu \in Y$ such that $st\nu \in \mathrm{pre}\,\mathcal{L}_\mathrm{a}$ we obtain $st\nu \in \mathrm{pre}\,\mathcal{H}$ by controllability [H1]. Else, we must have $st\mu \in \mathrm{pre}\,\mathcal{L}_\mathrm{a} \subseteq \mathrm{pre}\,\mathcal{L}'$ for some $\mu \in U$, and, referring to $\mathcal{L}' \subseteq (UY)^\omega$, $stY \cap \mathrm{pre}\,\mathcal{L} = \emptyset$. Since $\mathcal{L}'$ and $\mathcal{H}$ are non-conflicting [H2'] this implies $st\mu' \in \mathrm{pre}\,\mathcal{H}$ for some $\mu' \in U$. Together with the locally free input this implies $st\mu \in \mathrm{pre}\,\mathcal{L}_\mathrm{a}$. Thus, for both cases, we have extended the string st by one more event within $(\mathrm{pre}\,\mathcal{L}_\mathrm{a}) \cap (\mathrm{pre}\,\mathcal{H})$. Applying this argument iteratively, we obtain an infinite extension w such that $sw \in (\mathrm{clo}\,\mathcal{L}_\mathrm{a}) \cap (\mathrm{clo}\,\mathcal{H})$. Since both languages are closed, we conclude $sw \in \mathcal{L}_\mathrm{a} \cap \mathcal{H} \subseteq \mathcal{L}' \cap \mathcal{H}$. □

We first comment for the case that $\mathcal{L}$ is additionally closed and recover the previous Lemma 15 as a special case. Condition [A1] together with the alternation of input events U with output events Y correspond to a *free input* and a *non-anticipating* output as originally proposed by Willems (1991), while [A3] is trivially satisfied for closed plants $\mathcal{L}$. Alternation can be introduced to the common setting by interpreting the choice of the control pattern as input, i.e., $U = \Gamma$, and the execution of a transition by the phenomenon as output, i.e., $Y = \Sigma$. Then, the locally free input [A1] requires that any control pattern can be applied after any output event. If for particular strings $s \in \mathrm{pre}\,\mathcal{L} \cap (UY)^*$ certain control patterns $\gamma \in \Gamma$ would deadlock the phenomenon, these deadlocks are to be made explicit by a distinguished error event in order to formally satisfy [A1]. Referring to the inclusion requirement $\mathcal{L} \subseteq \mathcal{L}'$, the distinguished error event must show in the abstraction $\mathcal{L}'$. In other words: the abstraction must hold the information of which control patterns are applicable to which generated strings without risking a deadlock. This effectively amounts to the above condition [C1].

For not-necessarily closed behaviours, [A2] requires the plant to be able to satisfy its own liveness properties by a strategic choice of transitions that does not impose a constraint on the infinite future of applicable input events. This is a somewhat subtle condition that is slightly stronger than a non-anticipating output proposed by Willems (1991). Effectively, it puts the plant in the position to uniformly cooperate with any controller. The example provided for the need for cooperation in Section 3 fails to satisfy [A3].

The converse implication may seem provocative but it is also of interest. The assumptions in the following lemma are rather restrictive — I hope to be able to identify variants with weaker assumptions in due course.

Lemma 17. Given a generic plant $\mathcal{L}'$ that is consistent with the union construct over a family of specific closed plant behaviours $\mathcal{L}_\mathrm{a}$, $a \in A$ in the sense of $\cup_{a\in A}\mathcal{L}_\mathrm{a} \subseteq \mathcal{L}'$ and tight in the sense of $\mathrm{clo}\,\mathcal{L}' = \mathrm{clo}\cup_{a\in A}\mathcal{L}_\mathrm{a}$. Then any controller $\mathcal{H}$ that fails to be admissible to $\mathcal{L}'$ also fails to be admissible to at least one component $\mathcal{L}_\mathrm{a}$.

Proof. If $\mathcal{H}$ violates either [H0] or [H1], it is not admissible regardless the plant under consideration. Else $\mathcal{H}$ must violate [H2'], i.e., there exists $s \in (\mathrm{pre}\,\mathcal{L}') \cap (\mathrm{pre}\,\mathcal{H})$ such that any extension w with $sw \in \mathcal{H}$ is not in $\mathrm{pre}\,\mathcal{L}'$, and, thus, neither in any $\mathcal{L}_\mathrm{a}$. However, tightness implies $\mathrm{pre}\,\mathcal{L}' = \cup_{a\in A}\,\mathrm{pre}\,\mathcal{L}_\mathrm{a}$. Therefore, we can choose $a \in A$ such that $s \in (\mathrm{pre}\,\mathcal{L}_\mathrm{a}) \cap (\mathrm{pre}\,\mathcal{H})$. This established a conflict between $\mathcal{L}_\mathrm{a}$ and $\mathcal{H}$. □

Conclusion

In the context of a model based controller design, the plant model can be seen to formally represent assumptions justified by the phenomenon under consideration, and the controller can utilise these assumptions to provide the guarantee that prescribed closed-loop objectives are indeed achieved. In this sense, the plant supports the controller in meeting the design objectives. However, the plant also imposes restrictions on what is regarded an admissible controller. Controllability is one such restriction and it is well understood. Liveness properties possessed by the plant deserve a more detailed discussion to identify their consequences for the admissibility of a controller. The common setting for supervisory control is to persistently maintain the option that relevant liveness properties *can* be resolved eventually. Depending on the application at hand this may not be sufficient. In this report, I propose to interpret the situation of liveness properties represented by a not-topologically-closed plant as the result of model abstraction introduced when recovering generic properties for a class of plants, i.e., I make explicit that the plant is an assumption n the phenomenon and not the phenomenon itself. Technically, the abstraction is (a superset of) the union of infinitely many closed behaviours which, for automata realisations, amounts to an unknown initial condition to encode how liveness properties are resolved for each specific plant under consideration. This setting motivates additional technical conditions that allow for abstraction-based synthesis, i.e., conditions by which admissibility to the actual plant is implied by admissibility to the abstraction. To this end, the conjecture is that technical conditions in support of abstraction-based supervisory control should be essentially the same as those discussed in the context of reactive synthesis with a formal assume-guarantee approach — and this conjecture is followed up in Schmuck et al. (2017),

leaving the present report as a somewhat biased introduction to the supervision of sequential behaviours with particular focus on liveness properties.

References

X. Bai and T. Moor. Consistent abstractions for the supervision of sequential behaviours. In *IEEE Proceedings of the 56nd Conference on Decision and Control, CDC2017*, pages 565–571, 2017.

T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 278–292. IEEE Computer Society Press, 1996.

R. Kumar, V. Garg, and S. I. Marcus. On supervisory control of sequential behaviors. *IEEE Transactions on Automatic Control*, 37:1978–1985, 1992.

`libFAUDES`. Software library for discrete event systems. `http://www.rt.eei.uni-erlangen.de/FGdes/faudes`. Accessed: 2017-07-15.

T. Moor, K. Schmidt, and Th. Wittmann. Abstraction-based control for not necessarily closed behaviours. *Proceedings of the 18th IFAC World Congress*, pages 6988–6993, 2011.

T. Moor, Ch. Baier, T.-S. Yoo, F. Lin, and S. Lafortune. On the computation of supremal sublanguages relevant to supervisory control. *Workshop on Discrete Event Systems (WODES)*, pages 175–180, 2012.

P. J. Ramadge. Some tractable supervisory control problems for discrete-event systems modeled by büchi automata. *IEEE Transactions on Automatic Control*, 34:10–19, 1989.

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

P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77:81–98, 1989.

A.-K. Schmuck, T. Moor, and R. Majumdar. On the relation between reactive synthesis and supervisory control of non-terminating processes. Technical report, Max-Planck-Institut für Softwaresysteme, Kaiserslautern, Germany, 2017.

P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer-Verlag, 2009.

J. G. Thistle and W. M. Wonham. Control of omega-automata, church's problem, and the emptiness problem for tree omega-automata. *Proceedings of the 5th Workshop on Computer Science Logic*, pages 367–382, 1992.

J. G. Thistle and W. M. Wonham. Supervision of infinite behavior of discrete event systems. *SIAM J. Control and Optimization*, 32:1098–1113, 1994a.

J. G. Thistle and W. M. Wonham. Control of infinite behavior of finite automata. *SIAM J. Control and Optimization*, 32:1075–1097, 1994b.

J. C. Willems. Paradigms and puzzles in the theory of dynamic systems. *IEEE Transactions on Automatic Control*, 36:258–294, 1991.

Appendix: Synthesis Algorithms

We outline the algorithms presented by Thistle and Wonham (1992, 1994b) to compute the supremal ω-controllable sublanguage for the special case that the input data is provided by deterministic Büchi automata; see also `libFAUDES` for a freely available software implementation.

1: Closed plant behaviours

We begin with a closed plant behaviour, i.e., the situation of Section 2, and effectively assume that $\mathcal{L}_{\mathrm{loc}} = \lim L_{\mathrm{loc}}$ and $\mathcal{E} = \lim E \subseteq \mathcal{L}_{\mathrm{loc}}$ for regular languages $L_{\mathrm{loc}} \subseteq \Sigma^*$, $L_{\mathrm{loc}} = \mathrm{pre}\, L_{\mathrm{loc}}$ and $E \subseteq \Sigma^*$, which in turn are accepted by one automaton each. For technical reasons, we assume that the specification automaton generates the full language Σ^*, which is not restrictive. In particular, we can construct a product $G := (Q, \Sigma, \delta, Q_{\mathrm{o}}, Q_{\mathrm{E}})$ of both automata to generate $\mathcal{L}_{\mathrm{loc}}$ and to accept $\mathcal{E}$. The key observation is that if for any two finite strings s' and s'' with $s' \,[\equiv_L]\, s''$ and $s' \,[\equiv_E]\, s''$ one is in the controllability prefix, then so is the other. Thus, we can represent the controllability prefix $\mathrm{ctrl}\,\mathcal{E}$ as a set of states $Q_{\mathrm{ctrl}} \subseteq Q$.

The algorithm to obtain the set $Q_{\mathrm{ctrl}} \subseteq Q$ is stated as a two-level nested fixpoint iteration built on the *one-step backward controlled reachability operator* θ that maps a set of target states $T \subseteq Q$ to its predecessor as follows:

$$\theta(T) := \{\, q \in Q \mid (\exists\, \sigma \in \Sigma : \emptyset \neq \delta(q,\sigma) \subseteq T) \text{ and } (\forall\, \sigma \in \Sigma_{\mathrm{uc}} : \delta(q,\sigma) \subseteq T) \,\}. \quad (15)$$

Thus, $q \in \theta(T)$ corresponds to the existence of a control pattern that, when applied to q, enforces that some transition will be executed with successor state within the target T. By iterating this operator, one can identify all those states, that can be forced to enter the target by any finite number of transitions: the *star-step backward controlled reachability operator* p is defined by the following iteration.

1: **procedure** $p(T)$
2: $\quad T_{\mathrm{it}} \leftarrow \emptyset$
3: $\quad$ **repeat**

```
4:          T_it ← T_it ∪ θ(T_it ∪ T)
5:      until T_it attains a fixpoint
6:      return T_it
7:  end procedure
```

Technically, the above pseudo-code computes the smallest fixpoint of the monotone operator $\theta(\,\cdot\,\cup T)$. A framework for the discussion of fixpoints of monotone operators on sets is provided by the so called *μ-calculus*. A concise introduction is given in (Thistle and Wonham, 1994a). The above iteration is represented by the μ-calculus formula

$$p(T) := \mu T_{\mathrm{it}} \,.\, \theta(T_{\mathrm{it}} \cup T)\,, \tag{16}$$

where "μT_{it} ." reads the smallest fixpoint of the expression after the " . " interpreted as an operator with argument "T_{it}". In this appendix, we use μ-calculus formulae as a short form to represent iterations.

If we restrict the original target T by $p(T)$ and apply p again, we obtain states $p(T \cap p(T))$ that can be controlled to the restricted target $T \cap p(T)$, i.e., we have reached a target state that can once again be controlled to reach the target. Repeating this argument to iteratively restrict the target until a fixpoint is attained, we obtain the set of states that can be controlled to reach the target infinitely often. The corresponding operator C is defined by the below pseudo-code.

```
1:  procedure C(T)
2:      R_it ← Q
3:      repeat
4:          R_it ← R_it ∩ p(R_it ∩ T)
5:      until R_it attains a fixpoint
6:      return R_it
7:  end procedure
```

Again, the iteration can be stated as a μ-calculus formula:

$$C(T) := \nu R_{\mathrm{it}} \,.\, p(R_{\mathrm{it}} \cap T)\,, \tag{17}$$

$$= \nu R_{\mathrm{it}} \,.\, \mu T_{\mathrm{it}} \,.\, \theta(T_{\mathrm{it}} \cup (T \cap R_{it}))\,, \tag{18}$$

where "ν" reads "largest fixpoint" and $C(T)$ is indeed the largest fixpoint of the monotone operator $p(\,\cdot\,\cap T)$.

Applying this formula to the set of accepted states we obtain a state-set representation of the controllability prefix, i.e., with $Q_{\mathrm{ctrl}} := C(Q_E)$ we have $s \in \mathrm{ctrl}\,\mathcal{E}$ if and only if $\delta(Q_0, s) \in Q_{\mathrm{ctrl}}$. In other words, if we use Q_{ctrl} as marked states, we obtain an automaton that accepts $\mathrm{ctrl}\,\mathcal{E}$. Once the controllability prefix $\mathrm{ctrl}\,\mathcal{E}$ is established, the supremum $\mathcal{K}^{\uparrow}$ is obtained as the intersection of $\mathcal{E}$ with the limit of the supremal closed and controllable subset of $\mathrm{ctrl}\,\mathcal{E}$ and there are various options to extract a specific admissible controller $\mathcal{H}$ that enforces $\mathcal{E}$. For example, one can record the control patterns found in the above iteration to obtain a controller that chooses the shortest path to reach a marked state. Alternatively, one may start with any closed subset $\mathcal{K}$ of $\mathcal{K}^{\uparrow}$ to observe that the infimal closed and controllable

superset lies within $\mathrm{ctrl}\,\mathcal{K}^{\uparrow}$ and, thus, $\mathcal{K}$ can be established as lower bound specification. An option that is not considered in the cited literature is to aim for an implementation of $\mathcal{K}^{\uparrow}$ itself as not-topologically closed supervisor — this could be of interest in the context of cooperative supervision.

2: Not-necessarily closed plant behaviours

We now turn to the case of a not-necessarily closed plant behaviour $\mathcal{L}$, i.e., the situation of Section 4, and we restrict the presentation to the case where both $\mathcal{L}$ and $\mathcal{E} \subseteq \mathcal{L}$ are represented by one deterministic Büchi automaton each, i.e., $\mathcal{L} = \lim L$ and $\mathcal{E} = \lim E$ for regular languages L and E. Similar to the situation with a closed plant, we can construct a product automaton $G := (Q, \Sigma, \delta, Q_{\mathrm{o}}, Q_{\mathrm{E}})$ that generates the local plant behaviour $\mathrm{pre}\,\mathcal{L}$ and such that we can choose marked states Q_{E} to accept $\mathcal{E}$. Regarding $\mathcal{L}$, we retrieve an additional set of marked states Q_{L} such that $\mathcal{L} = \lim\{s \mid \delta(Q_0, s) \cap Q_{\mathrm{L}} \neq \emptyset\}$. If for any two finite strings s' and s'' with $s' \,[\equiv_L]\, s''$ and $s' \,[\equiv_E]\, s''$ one is in the controllability prefix, then so is the other. Thus, the controllability prefix can again be represented as a set of states $Q_{\mathrm{ctrl}} \subseteq Q$.

As in the situation of a closed plant, the computation of Q_{ctrl} can be based on an iteration of a backward reachability operator. Here, we need to consider that the liveness properties of the plant can be utilised when satisfying the specification: if, e.g., an unmarked state $q \notin Q_{\mathrm{L}}$ has an uncontrollable selfloop, the procedure can trust the plant to eventually exit this loop; if in addition a transition to a target state $q' \in T \subseteq Q$ is possible, the procedure shall identify q as a state that can be controlled to reach T. This line of thought extends to unmarked strictly connected components. To account for such situations, the *one-step backward controlled reachability operator* θ from Section 2 is replaced by a strategically designed additional iteration.

The *conditional one-step backward controlled reachability operator* θ' maps a set of target states $T \subseteq Q$ w.r.t. a constraint $D \subseteq Q$ to its predecessor as follows:

$$\theta'(T, D) := \{\, q \in Q \mid \\ (\exists\, \sigma \in \Sigma : \emptyset \neq \delta(q, \sigma) \subseteq T) \text{ and } (\forall\, \sigma \in \Sigma_{\mathrm{uc}} : \delta(q, \sigma) \subseteq T \cup D)\,\}\,. \quad (19)$$

Thus, $q \in \theta(T)$ corresponds to the existence of a control pattern under which the target T is reachable with the guarantee that the target is indeed attained under the assumption that the domain constraint D is violated. The intention is to apply a domain constraint that is disjoint to Q_{L}, so we know that it will be eventually violated by the plant. More precisely, we consider the following fixpoint for a given target $T \subseteq Q$ and a constraint $D \subseteq Q$ (think of $D = Q$ for the first reading):

$$\theta''(T, D) := \mu T_{\mathrm{it}}\,.\, \theta'((T_{\mathrm{it}} - Q_{\mathrm{L}}) \cup T,\, D - Q_{\mathrm{L}})\,. \quad (20)$$

From the first argument to θ' we conclude for all $q \in \theta''(T, D)$ that there exists a path to the target without passing a marking Q_{L} in between. Regardless the choice of D, we conclude by the second argument to θ' that, under suitable control, from

any $q \in \theta''(T,\ D)$ a marking can only be reached by entering or passing the target. Regarding the parameter D, any state in $q \in \theta''(T,\ D)$ can be controlled to exit D only via a marking. This is not sufficient for our purposes: if D is a strict superset of $\theta''(T,\ D)$, the system may leave $\theta''(T,\ D)$ by an uncontrollable event without having passed T. This is prevented by the choice of D that satisfies $D = \theta''(T,\ D)$ with the largest fixpoint of $\theta''(T,\ \cdot\,)$ as the natural candidate:

$$\theta'''(T) := \nu D_{\mathrm{it}}\,.\,\mu T_{\mathrm{it}}\,.\,\theta'((T_{\mathrm{it}} - Q_{\mathrm{L}}) \cup T,\ D_{\mathrm{it}} - Q_{\mathrm{L}})\,. \tag{21}$$

Summarising the construction so far, $\theta'''(T)$ is the set of states that can be controlled such that a marking Q_{L} can only be reached by entering or passing T. In particular, $\theta'''(Q_{\mathrm{E}})$ are those states, that can be controlled such that passing a plant marking once implies that a specification marking was passed. Likewise, states in $\theta'''(\theta'''(Q_{\mathrm{E}}))$ can be controlled such that passing a plant marking twice implies that a specification marking must have been passed. This argument is extended for higher powers of θ''': the fixpoint iteration

$$p'(T) := \mu T_{\mathrm{it}}\,.\,\theta'''(T_{\mathrm{it}} \cup T)\,. \tag{22}$$

yield the states controllable to enter or pass the target T *once* under the assumption that the plant marking is passed sufficiently often. If we happen to have $P(T) = T$, then the local plant behaviour can be controlled such that closed-loop trajectories that satisfy the plant liveness properties are guaranteed to pass the target T infinitely often. Thus, we are looking for the largest restriction of T that is a fixpoint of P. This amounts to

$$C'(T) := \nu R_{\mathrm{it}}\,.\,p'(R_{\mathrm{it}} \cap T)\,. \tag{23}$$

Finally, using the specification marking Q_{E} as a target, the controllability prefix is obtained by $Q_{\mathrm{ctrl}} = C'(Q_{\mathrm{E}})$ and one continuous exactly as in the situation of a closed plant, i.e., one computes the supremal closed and controllable sublanguage of $\mathrm{ctrl}\,\mathcal{E}$, takes the limit and intersects with $\mathcal{E}$ to obtain $\mathcal{K}^{\uparrow}$. In particular, representability by deterministic Büchi automata is retained and $\mathcal{K}^{\uparrow}$ is the limit of a regular $*$-language. Again, one possibility to obtain a controller is to record the control patterns found in the above iteration when evaluating θ'.