Consistent Abstractions for the Supervision of Sequential Behaviours

Xiaoying Bai* Thomas Moor*

***Abstract*— A common approach to controller design of large scale discrete-event systems or hybrid systems is to apply synthesis procedures on an abstraction that is realised on a significantly smaller state set. However, for every abstraction-based controller design, an inherent problem is to guarantee that the controller is also admissible to the actual plant. This paper addresses abstraction-based supervisory control for not-necessarily topologically closed ω-languages and upper-bound language-inclusion specifications. We refer to an abstraction as *consistent for the purpose of controller design*, if any controller obtained for the abstraction is also admissible to the actual plant. The main results of our study are sufficient and necessary conditions to characterise consistency.**

***Index Terms*— discrete-event systems, supervisory control, sequential behaviours, ω-languages.**

I. INTRODUCTION

A model that represents the plant dynamics in all detail often turns out too complex for an efficient controller design. Here, a common approach is to apply synthesis methods to a less complex abstraction. Depending on the control objectives, the abstraction must satisfy certain conditions such that the resulting controller is also applicable to the actual plant and such that the closed loop performs satisfactionary.

The control problem considered in this paper is the supervision of non-terminating processes that can be adequately represented as languages of infinite words, also referred to as *ω-languages* or *sequential behaviours*. Given an upper-bound language-inclusion specification, the corresponding synthesis problem was originally proposed by [11], with solution procedures provided in [3] and, with a particular focus on liveness properties, in [15, 16]. In order to address an upper-bound specification based on an abstraction, the latter must account for all possible sequences from the actual plant, and we therefore consider supersets of the plant as abstraction candidates. A crucial question here is how to guarantee that an abstraction-based controller does not livelock or deadlock when applied to the actual plant.

Abstraction-based supervisory control has been intensively studied in the context of hierarchical control of regular $*$-languages. There, the abstraction is commonly obtained by observation maps or, as a special case, by natural projections; see e.g. [2, 4, 17]. The so called *observer property*, developed in [17], is a sufficient condition guaranteeing that any non-conflicting control exercised on the abstraction will also be non-conflicting for the actual plant. By explicitly referring to controllable restrictions, [5] elaborates a less restrictive criterion for the particular purpose of abstraction-based control. In [1], the conditions for abstraction-based control of I/O-systems proposed in [7] are further developed to address a hierarchical control architecture for ω-languages.

Another domain of abstraction-based control addresses hybrid plant models, where finite state abstractions are of a particular interest. A common approach here is to obtain the abstraction by use of a strategically constructed quotient state set. Here, the abstraction can turn out non-deterministic, through a *static quantiser* in [13], or deterministic with an *equivalence relation* in [14]; see also [8, 9]. In [10], a deterministic finite abstraction is constructed from a cover of the state set based on bounded-length external strings. As with the situation of regular $*$-languages, one challenge is to guarantee that control designed for the abstraction does not lead to a blocking closed loop when applied to the original hybrid system. All of the given references address this issue, either by construction or by additional conditions.

The study conducted in the present paper does not refer to a particular class of realisations but discusses the problem of abstraction-based supervisory control entirely in terms of ω-languages. In this setting, we define an abstraction as *consistent for the purpose of controller design* if any supervisor obtained for the abstraction does neither livelock nor deadlock when applied to the actual plant. We provide necessary and sufficient conditions for consistency to address the special case of topologically closed plant behaviours as well as for the general case of not-necessarily topologically closed plants. Naturally, the results obtained in this general setting need further refinement to be of practical use for specific classes of realisations. To this end, we demonstrate how our conditions can be evaluated for ω-regular languages.

The paper is organised as follows. Preliminaries and a problem statement are given in Sections II and III, respectively, including a concise summary on the supervision of ω-languages and a formal definition of our notion of consistency. Addressing closed languages, Section IV provides our first sufficient and necessary condition for consistent abstractions. In Section V, we turn to the general case and continue the discussion without assuming that the plant is topologically closed. Finally, Section VI accounts for regular languages and outlines how the provided criteria can be computationally evaluated.

II. PRELIMINARIES AND NOTATION

Given an *alphabet* Σ, the set of *finite strings* $s = \sigma_1\sigma_2\cdots\sigma_n$ with $\sigma_i \in \Sigma$, $1 \leq i \leq n$, $n \in \mathbb{N}$, is denoted Σ^*, including the *empty string* $\epsilon \in \Sigma^*$, $\epsilon \notin \Sigma$. For two

* Lehrstuhl für Regelungstechnik, Friedrich-Alexander Universität Erlangen-Nürnberg, Germany. Email lrt@fau.de.

strings $s = \sigma_1\sigma_2\cdots\sigma_n$, $t = \tau_1\tau_2\cdots\tau_k \in \Sigma^*$, let $st := \sigma_1\sigma_2\cdots\sigma_n\tau_1\tau_2\cdots\tau_k \in \Sigma^*$ denote the *concatenation* with $s\epsilon := s =: \epsilon s$ for the empty string. The set of all countably *infinite sequences* with symbols from Σ is denoted Σ^ω. For $s = \sigma_1\sigma_2\cdots\sigma_n \in \Sigma^*$ and $v = \tau_1\tau_2\tau_3\cdots \in \Sigma^\omega$ denote $sv \in \Sigma^\omega$ the concatenation, i.e., $sv := \sigma_1\sigma_2\cdots\sigma_n\tau_1\tau_2\tau_3\cdots$. Given a string $w \in \Sigma^* \cup \Sigma^\omega$, its *prefix* is given by $\operatorname{pre} w := \{s \in \Sigma^* \mid \exists\, v \in \Sigma^* \cup \Sigma^\omega : sv = w\}$. For $s \in \operatorname{pre} w$ we say that s is *a prefix of* w and write $s \leq w$. If in addition $s \neq w$, we write $s < w$.

Subsets $L \subseteq \Sigma^*$ and $\mathcal{L} \subseteq \Sigma^\omega$ are referred to as *$*$-languages* and *ω-languages*, respectively. We write $\operatorname{pre} L$ and $\operatorname{pre}\mathcal{L}$ to denote the respective image under the prefix operator. For $*$-languages, we have $L \subseteq \operatorname{pre} L$, and in the case of equality L is said to be *prefix-closed*. Given $L \subseteq \Sigma^*$, the *limit* is defined $\lim L := \{w \in \Sigma^\omega \mid w \text{ has infinitely many prefixes in } L\}$. For $\mathcal{L} \subseteq \Sigma^\omega$, the *topological closure* is defined $\operatorname{clo}\mathcal{L} := \lim\operatorname{pre}\mathcal{L}$, and $\mathcal{L}$ is said to be *topologically closed* if $\mathcal{L} = \operatorname{clo}\mathcal{L}$. This notion of a closure indeed defines a topology on Σ^ω.

Both notions of closedness are generalised for interpretation relative to a restricted domain. For $L, M \in \Sigma^*$, we say L is *relatively prefix-closed w.r.t. M* if $L = (\operatorname{pre} L) \cap M$. Likewise, $\mathcal{L} \subseteq \Sigma^\omega$ is *relatively topologically closed w.r.t. $\mathcal{M} \subseteq \Sigma^\omega$* if $\mathcal{L} = (\operatorname{clo}\mathcal{L}) \cap \mathcal{M}$. In either variant, closedness is equivalent to relative closedness w.r.t. a closed language.

A $*$-language $L \subseteq \Sigma^*$ is said to be *complete*, if for all $s \in L$ there exists $\sigma \in \Sigma$ such that $s\sigma \in \operatorname{pre} L$; this is a particular form of *liveness*, also referred to as the *absence of deadlocks*. For any $\mathcal{L} \subseteq \Sigma^\omega$, the prefix $\operatorname{pre}\mathcal{L}$ is complete and prefix-closed. For any $L \subseteq \Sigma^*$, we have $\operatorname{pre}\lim L = L$ if and only if L is complete and prefix-closed.

The prefix-operator distributes over arbitrary unions of languages. In particular, the union over (relatively) prefix-closed $*$-languages is (relatively) prefix-closed. The limit operator in general distributes only over finite unions. In particular, finite unions of topologically closed ω-languages are closed, however, the union over an infinite family of (relatively) topologically closed ω-languages is not necessarily (relatively) topologically closed. Regarding intersections, we have $\operatorname{pre}(L \cap H) \subseteq (\operatorname{pre} L) \cap (\operatorname{pre} H)$ and $\operatorname{pre}(\mathcal{L} \cap \mathcal{H}) \subseteq (\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\mathcal{H})$, where $L, H \subseteq \Sigma^*$ and $\mathcal{L}, \mathcal{H} \subseteq \Sigma^\omega$. In the case of equality, the respective languages are said to be *non-conflicting*. Note that non-conflictingness of the ω-languages $\mathcal{L}$ and $\mathcal{H}$ implies that $(\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\mathcal{H})$ is complete. The converse implication is true for topologically closed ω-languages; see also Proposition 5.

Given $\Sigma_o \subseteq \Sigma$, the *natural projection* $p_o : \Sigma^* \to \Sigma_o^*$ is iteratively defined by (a) let $p_o\,\epsilon := \epsilon$; and (b) with $s \in \Sigma^*$, $\sigma \in \Sigma$, let $p_o(s\sigma) := (p_o\, s)\sigma$ if $\sigma \in \Sigma_o$, or else $p_o(s\sigma) := p_o\, s$. The set-valued inverse is defined by $p_o^{-1} r := \{s \in \Sigma^* \mid p_o(s) = r\}$ for $r \in \Sigma_o^*$. For infinite sequences, the natural projection is defined as a map $p_o^\omega : \Sigma^\omega \to (\Sigma^* \cup \Sigma^\omega)$. Given $w \in \Sigma^\omega$, consider the set $p_o \operatorname{pre} w$ arranged as a monotone sequence. If the sequence exhibits a maximal element $u \in \Sigma_o^*$, let $p_o^\omega w := u$. If there is no maximal element, then there uniquely exists $v \in \Sigma_o^\omega$ such that $p_o \operatorname{pre} w = \operatorname{pre} v$, and we define $p_o^\omega w := v$. The set-valued inverse of p_o^ω is defined $p_o^{-\omega} v := \{w \in \Sigma^\omega \mid p_o^\omega w = v\}$ for $v \in (\Sigma^* \cup \Sigma^\omega)$.

III. PROBLEM STATEMENT

We begin with a concise review of supervisory control for sequential behaviours as originally discussed in [11], however, in a notation adapted to the particular purpose of the present paper. In our setting, both the plant and the controller are represented as ω-languages and the closed-loop behaviour is obtained by intersection. For a well-posed closed-loop configuration, we introduce the following admissibility requirements.

Definition 1. Given a plant $\mathcal{L} \subseteq \Sigma^\omega$ with uncontrollable events $\Sigma_{uc} \subseteq \Sigma$, a controller $\mathcal{H} \subseteq \Sigma^\omega$ is *admissible* to the plant, if

[H0] $\mathcal{H} = \operatorname{clo}\mathcal{H}$,

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

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

Condition [H0] implies that the controller can be implemented as a causal feedback map, which by condition [H1] never disables uncontrollable events. Condition [H2] ensures that the closed loop will neither run into deadlocks nor will it run into livelocks. By the following proposition, the achievable closed-loop behaviours in our formal setting indeed match the results from the original literature [11].

Proposition 2. Given Σ, denote $\Sigma_{uc} \subseteq \Sigma$ the uncontrollable events and consider the plant $\mathcal{L} \subseteq \Sigma^\omega$. If a controller $\mathcal{H} \subseteq \Sigma^\omega$ is admissible to $\mathcal{L}$, then the closed-loop behaviour $\mathcal{K} := \mathcal{L} \cap \mathcal{H}$ satisfies [K0] and [K1]:

[K0] $\mathcal{K}$ is relatively topologically closed w.r.t. $\mathcal{L}$, and

[K1] $((\operatorname{pre}\mathcal{K})\Sigma_{uc}) \cap (\operatorname{pre}\mathcal{L}) \subseteq \operatorname{pre}\mathcal{K}$.

Conversely, let $\mathcal{K} \subseteq \Sigma^\omega$ be a candidate closed-loop behaviour that satisfies [K0] and [K1]. Then $\mathcal{H} := \lim((\operatorname{pre}\mathcal{K})\Sigma_{uc}^*)$ is admissible to $\mathcal{L}$ with $\mathcal{K} = \mathcal{L} \cap \mathcal{H}$.

Proof. A self-contained proof is provided in the appendix. □

Given a plant $\mathcal{L} \subseteq \Sigma^\omega$ and an upper-bound specification $\mathcal{E} \subseteq \mathcal{L}$, the prototypical synthesis problem is to construct a candidate closed-loop behaviour $\mathcal{K} \subseteq \mathcal{E}$ that satisfies conditions [K0] and [K1]. While $\mathcal{K} = \emptyset$ always formally solves this problem, practical reasons suggest to ask for a *maximal permissive* solution, i.e., to consider the candidate

$$\mathcal{K}^\uparrow := \cup\{\mathcal{K} \subseteq \mathcal{E} \mid \mathcal{K} \text{ satisfies [K0] and [K1]}\}. \qquad (1)$$

Except for referring to ω-languages, property [K1] is literally identical to the notion of *controllability* commonly used for $*$-languages; see [12]. In particular, [K1] is retained under arbitrary union, and, thus, $\mathcal{K}^\uparrow$ itself satisfies [K1]. However, and in contrast to the situation with $*$-languages, relative topological closedness is in general not retained under arbitrary union and, hence, $\mathcal{K}^\uparrow$ is not expected to satisfy [K0]. A notable exception here is when $\mathcal{E}$ happens to be relatively topologically closed w.r.t. $\mathcal{L}$; see [11]. The general case is considerably more involved and the reader is kindly referred to [15, 16] for a detailed discussion. The results reported in the cited literature include an alternative

characterisation of $\mathcal{K}^{\uparrow}$ in terms of the so called *controllability prefix*, which in turn leads to a computational procedure that effectively solves the synthesis problem for finite state representations.

We now turn to the discussion of abstraction-based supervisory control, in which we are given the actual plant $\mathcal{L} \subseteq \Sigma^\omega$ and an abstraction $\mathcal{L}' \subseteq \Sigma^\omega$. For the purpose of this paper, we require the abstraction to account for any possible behaviour of the actual plant and not to render events that are actually uncontrollable as controllable. This amounts to the following technical conditions.

Definition 3. Given a plant $\mathcal{L} \subseteq \Sigma^\omega$ with uncontrollable events Σ_{uc}, an *abstraction* is a behaviour $\mathcal{L}' \subseteq \Sigma^\omega$ with uncontrollable events Σ'_{uc} such that

[A1] $\mathcal{L} \subseteq \mathcal{L}'$, and

[A2] $\Sigma_{\mathrm{uc}} \subseteq \Sigma'_{\mathrm{uc}}$. □

Condition [A1] ensures that any controller $\mathcal{H}'$ which restricts the abstraction $\mathcal{L}'$ to satisfy an upper bound $\mathcal{E}$ will also do so when applied to the actual plant $\mathcal{L}$; i.e.,

$$\mathcal{L} \cap \mathcal{H}' \subseteq \mathcal{L}' \cap \mathcal{H}' \subseteq \mathcal{E}. \tag{2}$$

This is a common prerequisite when using finite state abstractions of hybrid systems for a subsequent controller synthesis; see e.g. [14] for a comprehensive study, and also [6, 10] for a discussion in terms of sequential behaviours. Condition [A1] is also relevant when using natural projections for the hierarchical control of large-scale discrete-event systems; see e.g. [2, 7], and, specifically addressing ω-languages, [1]. There, low-level events are filtered by projection and a high-level supervisor deliberately exercises no control on low-level events. In the setting of the present paper, this amounts to choosing $\mathcal{L}' := \mathrm{p}_{\mathrm{o}}^{-\omega}\mathrm{p}_{\mathrm{o}}^{\omega}\mathcal{L}$ with $\Sigma'_{\mathrm{uc}} := \Sigma_{\mathrm{uc}} \cup (\Sigma - \Sigma_{\mathrm{o}})$ to comply with [A1] and [A2].

While [A1] guarantees that a prescribed upper bound is satisfied in an abstraction-based design, it is not obvious whether or under which conditions admissibility can be maintained. To this end, we pragmatically impose the following consistency requirement.

Definition 4. Consider a plant $\mathcal{L} \subseteq \Sigma^\omega$ with uncontrollable events Σ_{uc} and an abstraction $\mathcal{L}' \subseteq \Sigma^\omega$ with uncontrollable events Σ'_{uc}. Then the abstraction is *consistent for the purpose of supervisory control* (or short *consistent*) if it satisfies

[A3] $(\forall \mathcal{H}' \subseteq \Sigma^\omega)[\ \mathcal{H}'$ is admissible to $\mathcal{L}'$
$\Rightarrow\quad \mathcal{H}'$ is admissible to $\mathcal{L}\ ]$ □

Provided that a plant $\mathcal{L}$ and an abstraction $\mathcal{L}'$ comply with [A1] and [A2], it is readily observed that any controller $\mathcal{H}'$ that is admissible to $\mathcal{L}'$ also satisfies the admissibility conditions [H0] and [H1] for $\mathcal{L}$: [H0] does not refer to the plant at all, and [H1] is a trivial consequence of [A2]:

$$(\mathrm{pre}\,\mathcal{H}')\Sigma_{\mathrm{uc}} \subseteq (\mathrm{pre}\,\mathcal{H}')\Sigma'_{\mathrm{uc}} \subseteq \mathrm{pre}\,\mathcal{H}'. \tag{3}$$

Thus, to establish consistency of the abstraction, we are left to discuss non-conflictingness [H2] of $\mathcal{L}$ and $\mathcal{H}'$, and we will do so in the following two sections.

IV. CONSISTENCY FOR CLOSED BEHAVIOURS

In this section, we develop a sufficient and necessary condition for consistency in the sense of Definition 4 under the additional assumption that the actual plant is topologically closed. Although we will drop the latter assumption in the subsequent section, we believe that it is instructive to first discuss the less involved special case. In particular, for topologically closed plants livelocks are not an issue and, by the below proposition, non-conflictingness [H2] is equivalent to the absence of deadlocks.

Proposition 5. Consider two ω-languages $\mathcal{L}, \mathcal{H} \subseteq \Sigma^\omega$ and assume both to be topologically closed. Then $\mathcal{L}$ and $\mathcal{H}$ are non-conflicting if and only if $(\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H})$ is complete.

Proof. If $\mathcal{L}$ and $\mathcal{H}$ are non-conflicting then by definition $(\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H})$ equals the prefix of the ω-language $\mathcal{L} \cap \mathcal{H}$ and, therefore, is complete. For the converse implication, assume that $(\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H})$ is complete and pick $s \in (\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H})$. By completeness, we construct a sequence $(v_i)_{i\in\mathbb{N}}$ such that $s < v_i < v_{i+1}$ and $v_i \in (\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H})$ for all $i \in \mathbb{N}$. In particular, $\lim\{v_i \,|\, i \in \mathbb{N}\}$ is a singleton, which we denote $w \in \Sigma^\omega$ to observe $s < w$. Moreover, we have $\{w\} = \lim\{v_i \,|\, i \in \mathbb{N}\} \subseteq \lim((\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H})) \subseteq \lim \mathrm{pre}\,\mathcal{L} = \mathrm{clo}\,\mathcal{L} = \mathcal{L}$. Likewise, we obtain $w \in \mathcal{H}$, and, hence, $s \in \mathrm{pre}\, w \subseteq \mathrm{pre}(\mathcal{L} \cap \mathcal{H})$. □

If a controller $\mathcal{H}'$ is admissible to an abstraction $\mathcal{L}'$, we have that $(\mathrm{pre}\,\mathcal{L}') \cap (\mathrm{pre}\,\mathcal{H}')$ is complete, i.e., after a string $s \in (\mathrm{pre}\,\mathcal{L}') \cap (\mathrm{pre}\,\mathcal{H}')$ has occurred in the closed-loop configuration, the controller must at least enable one event that the abstraction can agree on. For an abstraction-based design, this must imply that there also exists an enabled event that is compliant with the actual plant $\mathcal{L}$. This is the case if, either, the actual plant can execute an uncontrollable event after s, or, else, that actual plant can execute any controllable event that is enabled by the abstraction. Both cases are formally encoded in the below condition [C1], which turns out sufficient and necessary for consistency.

Theorem 6. Consider a topologically closed plant $\mathcal{L} \subseteq \Sigma^\omega$ with uncontrollable events Σ_{uc}, and an abstraction $\mathcal{L}' \subseteq \Sigma^\omega$ with uncontrollable events Σ'_{uc}, subject to conditions [A1] and [A2]. Then $\mathcal{L}'$ is a consistent abstraction if and only if condition [C1] is satisfied:

[C1] $(\forall\, s \in \mathrm{pre}\,\mathcal{L})[\,(s\Sigma'_{\mathrm{uc}}) \cap (\mathrm{pre}\,\mathcal{L}) = \emptyset$
$\Rightarrow\quad (s\Sigma) \cap (\mathrm{pre}\,\mathcal{L}) = (s\Sigma) \cap (\mathrm{pre}\,\mathcal{L}')\,]$.

Proof. For the constructive part of this proof, assume that [C1] is indeed satisfied and pick any controller $\mathcal{H}'$ that is admissible to the abstraction $\mathcal{L}'$. Referring to Proposition 5, we need to show that $(\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H}')$ is complete. Pick an arbitrary $s \in (\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H}')$. If it is the case that there exists $\sigma \in \Sigma'_{\mathrm{uc}}$ such that $s\sigma \in \mathrm{pre}\,\mathcal{L}$, we refer to admissibility of $\mathcal{H}'$ to $\mathcal{L}'$ to obtain $s\sigma \in \mathrm{pre}\,\mathcal{H}'$. This implies $s\sigma \in (\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H}')$ and closed this case. Otherwise, we must have that $(s\Sigma'_{\mathrm{uc}}) \cap (\mathrm{pre}\,\mathcal{L}) = \emptyset$ and we may later refer to the implication in [C1]. By [A1] we observe that $s \in (\mathrm{pre}\,\mathcal{L}') \cap (\mathrm{pre}\,\mathcal{H}')$ and by admissibility of $\mathcal{H}'$ to $\mathcal{L}'$

we can pick $\sigma \in \Sigma$ such that $s\sigma \in (\mathrm{pre}\,\mathcal{L}') \cap (\mathrm{pre}\,\mathcal{H}')$. Now the right-hand-side of [C1] implies $s\sigma \in (\mathrm{pre}\,\mathcal{L})$ to close this case. In both cases, we have extended s within $(\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H}')$ by one more event. Thus, have indeed established completeness, and, referring to Proposition 5, non-conflictingness. This amounts to the implication "[C1] $\Rightarrow$ (the abstraction is consistent)".

We now turn to "($\neg$ [C1]) $\Rightarrow$ (the abstraction is not consistent)", i.e., under the given hypothesis that [C1] is not satisfied, we construct a controller $\mathcal{H}'$ that is admissible to the abstraction $\mathcal{L}'$ but fails to be admissible to the actual plant $\mathcal{L}$. If [C1] is indeed not satisfied, we can pick a witness $s \in \mathrm{pre}\,\mathcal{L}$ such that the left-hand-side of [C1] holds while the right-hand-side of [C1] fails. From the right-hand-side to fail and referring to [A1], we pick $\sigma \in \Sigma$ such that $s\sigma \in \mathrm{pre}\,\mathcal{L}'$ but $s\sigma \notin \mathrm{pre}\,\mathcal{L}$. Based on the choice of s and σ, we define a candidate controller $\mathcal{H}'$ by

$$\mathcal{H}' := \{s(\Sigma'_{\mathrm{uc}} \cup \{\sigma\})w \,|\, w \in \Sigma^\omega\} \;\cup\; \{w \in \Sigma^\omega \,|\, s \notin \mathrm{pre}\, w\}\,. \quad (4)$$

It is readily observed that $\mathcal{H}'$ is admissible to $\mathcal{L}'$ (closedness [H0] follows by the union construct, [H1] and [H2] are verified via elementary case distinction). However, the controller $\mathcal{H}'$ has the property $(s\Sigma) \cap (\mathrm{pre}\,\mathcal{H}') = (s\Sigma'_{\mathrm{uc}}) \cup \{s\sigma\}$ while $(s\Sigma'_{\mathrm{uc}}) \cap (\mathrm{pre}\,\mathcal{L}) = \emptyset$ and $s\sigma \notin \mathrm{pre}\,\mathcal{L}$. Hence, $(s\Sigma) \cap (\mathrm{pre}\,\mathcal{H}') \cap (\mathrm{pre}\,\mathcal{L}) = \emptyset$ to demonstrate a conflict of $\mathcal{L}$ and $\mathcal{H}'$ at s. Therefore, the candidate $\mathcal{H}'$ is not admissible to $\mathcal{L}$ and the abstraction is indeed not consistent. □

V. CONSISTENCY FOR GENERAL BEHAVIOURS

We turn to the general case in dropping the requirement of topological closedness and again derive a characterisation of consistency. Here, the plant is subject to potential livelocks and we can not expect [C1] to be a sufficient condition to prevent such undesirable closed-loop artefacts.

To formally investigate the situation, we observe that if the actual plant $\mathcal{L} \subseteq \Sigma^\omega$ in closed-loop configuration conflicts with an abstraction-based controller $\mathcal{H}' \subseteq \Sigma^\omega$, we can nominate a witness $s \in \Sigma^*$ such that

$$s \in (\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H}')\,, \quad s \notin \mathrm{pre}(\mathcal{L} \cap \mathcal{H}')\,. \quad (5)$$

This is referred to as a *conflict at s*. Assuming for the moment that [C1] is satisfied, $\mathcal{H}'$ effectively controls the actual plant $\mathcal{L}$ to evolve after s within $\mathrm{pre}\,\mathcal{L}$ for infinite time but prevents the achievement of acceptance conditions as expressed by $\mathcal{L}$. This undesirable situation can be formally rephrased as an upper-bound specification:

$$\mathcal{E}_s := \{sw \,|\, w \in \Sigma^\omega,\ sw \notin \mathcal{L}\} \cup \{w \in \Sigma^\omega \,|\, s \notin \mathrm{pre}\, w\}\,. \quad (6)$$

Therefore, we may expect that the "evil" controller $\mathcal{H}' \subseteq \Sigma^\omega$ enforces $\mathcal{E}_s$ to be satisfied and that it will show the witness-of-conflict s in its designated closed-loop behaviour, i.e., $s \in \mathrm{pre}(\mathcal{L}' \cap \mathcal{H}')$. Whether or not such an unfortunate controller exists, can be tested by inspecting the supremum $\mathcal{K}_s$ over all achievable closed-loop behaviours that satisfy $\mathcal{E}_s$, i.e.,

$$\mathcal{K}_s := \cup\,\{\mathcal{K} \subseteq \mathcal{E}_s \cap \mathcal{L}' \,|\, \mathcal{K} \text{ satisfies [K0] and [K1] for the abstraction } \mathcal{L}' \text{ with uncontrollable events } \Sigma'_{\mathrm{uc}}\}\,. \quad (7)$$

The following theorem further elaborates the above line of thought to obtain a characterisation of consistency.

Theorem 7. Consider a plant $\mathcal{L} \subseteq \Sigma^\omega$ with uncontrollable events Σ_{uc}, and an abstraction $\mathcal{L}' \subseteq \Sigma^\omega$ with uncontrollable events Σ'_{uc}, subject to [A1] and [A2]. Then $\mathcal{L}'$ is a consistent abstraction if and only if condition [C2] is satisfied:

[C2] $(\forall\, s \in \mathrm{pre}\,\mathcal{L})[\, s \notin \mathrm{pre}\,\mathcal{K}_s\,]$,

with $\mathcal{K}_s$ as defined by Eqs. (6) and (7).

Proof. We first establish the implication "(the abstraction is not consistent) $\Rightarrow$ ($\neg$[C2])". Assuming that $\mathcal{L}'$ is not a consistent abstraction, we can pick a controller $\mathcal{H}'$ that is admissible to the abstraction $\mathcal{L}'$ but not to the actual plant $\mathcal{L}$. Since we have established [H0] and [H1] by [A1] and [A2] regardless of the particular plant at hand, $\mathcal{L}$ and $\mathcal{H}'$ fail on [H2]. Thus, there is a conflict and we can pick a witness s according to Eq. (5). Here, $\mathcal{L} \subseteq \mathcal{L}'$ [A1] implies that $s \in (\mathrm{pre}\,\mathcal{L}') \cap (\mathrm{pre}\,\mathcal{H}')$, and therefore admissibility to $\mathcal{L}'$ with [H2] ensures that we can pick $w \in \Sigma^\omega$ such that $sw \in \mathcal{L}' \cap \mathcal{H}'$. However, for any such choice of w we must, by Eq. (5), have $sw \notin \mathcal{L}$. Therefore, $\mathcal{K} := \mathcal{L}' \cap \mathcal{H}' \subseteq \mathcal{E}_s$ and $s \in \mathrm{pre}(\mathcal{L}' \cap \mathcal{H}') = \mathrm{pre}\,\mathcal{K}$. By Proposition 2 and admissibility of $\mathcal{H}'$ to $\mathcal{L}'$, the closed loop $\mathcal{K}$ qualifies as a component of the union construct in Eq. (7) and we have $\mathcal{K} \subseteq \mathcal{K}_s$. Applying prefixes, we obtain $s \in \mathrm{pre}\,\mathcal{K} \subseteq \mathrm{pre}\,\mathcal{K}_s$ and thus have established that [C2] is indeed not satisfied.

We now turn to "($\neg$[C2]) $\Rightarrow$ (the abstraction is not consistent)". Here, we can pick $s \in \mathrm{pre}\,\mathcal{L}$ such that $s \in \mathrm{pre}\,\mathcal{K}_s$. Since the prefix operator distributes over arbitrary unions, there must exist $\mathcal{K} \subseteq \mathcal{E}_s$ that satisfies [K0] and [K1] such that $s \in \mathrm{pre}\,\mathcal{K}$. Referring to Proposition 2, we can choose $\mathcal{H}'$ admissible to $\mathcal{L}'$ such that $\mathcal{K} = \mathcal{L}' \cap \mathcal{H}'$. In particular, we have $s \in \mathrm{pre}\,\mathcal{K} \subseteq \mathrm{pre}\,\mathcal{H}'$ and, together with the initial choice of s, also $s \in (\mathrm{pre}\,\mathcal{L}) \cap (\mathrm{pre}\,\mathcal{H}')$. Now assume that we can pick a continuation $w \in \Sigma^\omega$ with $sw \in \mathcal{L} \cap \mathcal{H}'$. Then, by [A1], $sw \in \mathcal{L}' \cap \mathcal{H}' = \mathcal{K} \subseteq \mathcal{E}_s$. By Eq. (6), this implies $sw \notin \mathcal{L}$, and, hence, contradicts the choice of w. Therefore, $sw \notin \mathcal{L} \cap \mathcal{H}'$ for all $w \in \Sigma^\omega$, and hence $s \notin \mathrm{pre}(\mathcal{L} \cap \mathcal{H}')$. Thus, $\mathcal{L}$ and $\mathcal{H}'$ conflict at s and the abstraction $\mathcal{L}'$ is not consistent. □

Although we referred to [C1] in our motivation, the above theorem establishes that [C2] by its own characterises consistency of an abstraction. Hence, we conclude that for topologically closed plants [C1] and [C2] are equivalent.

VI. IMPLEMENTATION OUTLINE

Both conditions [C1] and [C2] are stated as a universal quantification over all prefixes of the actual plant and a test, which is to be applied to each individual prefix. Since there are infinitely many prefixes, a prerequisite for a software

implementation is to establish a finite set of representatives. Technically, we use the conjunction of Nerode equivalences and show that the test either uniformly passes or uniformly fails for all prefixes within the same equivalence classes. For regular languages, a software implementation can then be based on an iteration over all states.

We begin the discussion under the assumption that the actual plant $\mathcal{L} \subseteq \Sigma^\omega$ is topologically closed and target for condition [C1]. By the definition of closedness, we have $\mathcal{L} = \lim L$ with $L{:=}\operatorname{pre}\mathcal{L}$. This suggests to utilise the well studied Nerode equivalence for $*$-languages.

Definition 8. Let Σ denote an alphabet. For $L \subseteq \Sigma^*$, *L-equivalence* $[\equiv_L]$ is defined for $s', s'' \in \Sigma^*$ by $s'[\equiv_L]s''$ if and only if $(\forall\, t \in \Sigma^*)[\, s't \in L \;\leftrightarrow\; s''t \in L\,]$. □

Inspecting [C1], it is readily observed for any two strings $s', s'' \in \Sigma^*$ with $s'[\equiv_{\operatorname{pre}\mathcal{L}}]s''$ and $s'[\equiv_{\operatorname{pre}\mathcal{L}'}]s''$ that

$$\begin{aligned}&(s'\Sigma_{\mathrm{uc}}) \cap (\operatorname{pre}\mathcal{L}) = \emptyset\\ &\qquad\Leftrightarrow\quad (s''\Sigma_{\mathrm{uc}}) \cap (\operatorname{pre}\mathcal{L}) = \emptyset\,, \qquad (8)\\ &(s'\Sigma) \cap (\operatorname{pre}\mathcal{L}) = (s'\Sigma) \cap (\operatorname{pre}\mathcal{L}')\\ &\qquad\Leftrightarrow\quad (s''\Sigma) \cap (\operatorname{pre}\mathcal{L}) = (s''\Sigma) \cap (\operatorname{pre}\mathcal{L}')\,. \qquad (9)\end{aligned}$$

When $\operatorname{pre}\mathcal{L}$ and $\operatorname{pre}\mathcal{L}'$ are provided as deterministic finite automata realisations, one can use the common product construct to obtain an automaton in which each state corresponds to a set of strings that belong to exactly one equivalence class of $[\equiv_{\operatorname{pre}\mathcal{L}}]$ and $[\equiv_{\operatorname{pre}\mathcal{L}'}]$, respectively. Moreover, the sets in the individual tests effectively amount to sets of possible successor events. Thus, the overall condition [C1] can be evaluated by an iteration over all states in the product automaton with an inspection of enabled events in each state.

In the subsequent discussion of [C2], we again refer to an equivalence relation on Σ^*, but now take into account the infinite future w.r.t. an ω-language.

Definition 9. Given an alphabet Σ and an ω-language $\mathcal{L} \subseteq \Sigma^\omega$, *$\mathcal{L}$-equivalence* is defined for $s', s'' \in \Sigma^*$ by $s'[\equiv_{\mathcal{L}}]s''$ if and only if $(\forall\, w \in \Sigma^\omega)[\, s'w \in \mathcal{L} \;\Leftrightarrow\; s''w \in \mathcal{L}\,]$. □

For two $\mathcal{L}$-equivalent strings $s', s'' \in \Sigma^*$, intuitively, any admissible control exercised after s' can also be applied after s''. Moreover, if a controller causes a conflict at s', the same control when exercised at s'' will also cause a conflict. The following two propositions clarify this informal observation.

Proposition 10. Given a plant $\mathcal{L} \subseteq \Sigma^\omega$ with uncontrollable events Σ_{uc}, an admissible controller $\mathcal{H} \subseteq \Sigma^\omega$, and two $\mathcal{L}$-equivalent strings $s', s'' \in \Sigma^*$, $s'[\equiv_{\mathcal{L}}]s''$ with $s' \in \operatorname{pre}\mathcal{H}$, let

$$\tilde{\mathcal{H}} := \{s''w \,|\, w \in \Sigma^\omega,\ s'w \in \mathcal{H}\} \cup \{w \in \Sigma^\omega \,|\, s'' \notin \operatorname{pre} w\}\,. \quad (10)$$

Then $\tilde{\mathcal{H}}$ is admissible to $\mathcal{L}$.

Proof. Regarding [H0], observe that both union components in Eq. (10) are topologically closed, and, hence, so is $\tilde{\mathcal{H}}$.

Regarding [H1], pick any $s \in \operatorname{pre}\tilde{\mathcal{H}}$ and $\sigma \in \Sigma_{\mathrm{uc}}$. Case (a), if $s\sigma \leq s''$, then we refer to the left union component (l.u.c.) to obtain $s\sigma \in \operatorname{pre}\tilde{\mathcal{H}}$. Case (b), we have $s\sigma \notin \operatorname{pre} s''$. Let $t \in \operatorname{pre}(s\sigma)$ denote the longest string such that $t \leq s''$ and write $s\sigma = tu$ with $u \in \Sigma^*$, $u \neq \epsilon$. We distinguish two more sub-cases. Case (b1), if $t = s''$, we have $s'' < s\sigma$ and refer to the l.u.c. together with $(\operatorname{pre}\mathcal{H})\Sigma_{\mathrm{uc}} \subseteq \operatorname{pre}\mathcal{H}$ to obtain $s\sigma \in \operatorname{pre}\tilde{\mathcal{H}}$. For the remaining case (b2) we have $t < s''$. Here, we observe that $s'' \notin \operatorname{pre} tuw$ for any $w \in \Sigma^\omega$, and refer to the right union component (r.u.c.) to obtain $s\sigma w = tuw \in \tilde{\mathcal{H}}$. This concludes the proof of [H1]. For a proof of [H2], we pick an arbitrary $s \in (\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\tilde{\mathcal{H}})$. Then we can choose $w \in \Sigma^\omega$ such that $sw \in \mathcal{L}$. Case (a), if $s'' \notin \operatorname{pre} sw$, the r.u.c. provides us $sw \in \mathcal{H}$. We are left with case (b) where we assume that $s'' < sw$. This implies $s'' \in \operatorname{pre}\mathcal{L}$ and, by $\mathcal{L}$-equivalence, $s' \in \operatorname{pre}\mathcal{L}$. We distinguish two sub cases. For case (b1) assume that $s \leq s''$. By non-conflictingness of $\mathcal{L}$ and $\mathcal{H}$, we choose $v \in \Sigma^\omega$ such that $s'v \in \mathcal{L} \cap \mathcal{H}$. Thus, we have $s''v \in \tilde{\mathcal{H}}$ via the l.u.c. and $s''v \in \mathcal{L}$ by $s'[\equiv_{\mathcal{L}}]s''$. This implies $s \leq s'' \in \operatorname{pre}(\mathcal{L} \cap \mathcal{H}')$ and closes case (b1). For case (b2), we assume that $s'' < s$. Here, we write $s = s''t$ with $t \in \Sigma^*$ and refer to the l.u.c. to obtain $s'tv \in \mathcal{H}$ for suitably chosen $v \in \Sigma^\omega$. Note that $s'[\equiv_{\mathcal{L}}]s''$ implies $s't[\equiv_{\mathcal{L}}]s''t$ and, thus, $s''t = s \in \operatorname{pre}\mathcal{L}$ implies $s't \in \operatorname{pre}\mathcal{L}$. Referring to non-conflictingness of $\mathcal{L}$ and $\mathcal{H}$, this ensures existence of $u \in \Sigma^\omega$, such that $s'tu \in \mathcal{L} \cap \mathcal{H}$. We conclude $su = s''tu \in \tilde{\mathcal{H}}$ by the l.u.c. and $su = s''tu \in \mathcal{L}$ by $s'[\equiv_{\mathcal{L}}]s''$. This closes case (b2) with $su \in \mathcal{L} \cap \tilde{\mathcal{H}}$. □

Proposition 11. Given a plant $\mathcal{L} \subseteq \Sigma^\omega$ with uncontrollable events Σ_{uc}, some not necessarily admissible controller $\mathcal{H} \subseteq \Sigma^\omega$, and two $\mathcal{L}$-equivalent strings $s', s'' \in \Sigma^*$, $s'[\equiv_{\mathcal{L}}]s''$, consider $\tilde{\mathcal{H}}$ as defined in Eq. (10). Then

$$\begin{aligned}&s' \in (\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\mathcal{H}) \quad\text{and}\quad s' \notin \operatorname{pre}(\mathcal{L} \cap \mathcal{H}) \quad (11)\\ \Rightarrow\quad &s'' \in (\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\tilde{\mathcal{H}}) \quad\text{and}\quad s'' \notin \operatorname{pre}(\mathcal{L} \cap \tilde{\mathcal{H}}) \quad (12)\end{aligned}$$

Proof. The first clause in Eq. (12) is an immediate consequence of $s'[\equiv_{\mathcal{L}}]s''$ and Eq. (10), respectively. For a proof by contradiction of the second clause, we now assume that $s'' \in \operatorname{pre}(\mathcal{L} \cap \tilde{\mathcal{H}})$ and choose $w \in \Sigma^\omega$ such that $s''w \in \mathcal{L} \cap \tilde{\mathcal{H}}$. By $s'[\equiv_{\mathcal{L}}]s''$, this implies $s'w \in \mathcal{L}$, and, by Eq. (10), $s'w \in \mathcal{H}$. This contradicts the second clause in Eq. (11). □

The above propositions are utilised in the below theorem to establish, that the test $s \in \operatorname{pre}\mathcal{K}_s$ from condition [C2] evaluates uniformly over sets of strings that are $\mathcal{L}$-equivalent and $\mathcal{L}'$-equivalent.

Theorem 12. Given a plant $\mathcal{L} \subseteq \Sigma^\omega$ with uncontrollable events Σ_{uc}, and an abstraction $\mathcal{L}' \subseteq \Sigma^\omega$ with uncontrollable events Σ'_{uc} subject to conditions [A1] and [A2], consider strings $s', s'' \in \operatorname{pre}\mathcal{L}$ such that $s'[\equiv_{\mathcal{L}}]s''$ and $s'[\equiv_{\mathcal{L}'}]s''$. Then $s' \in \operatorname{pre}\mathcal{K}_{s'}$ if and only if $s'' \in \operatorname{pre}\mathcal{K}_{s''}$, where we refer to Eqs. (6) and (7).

Proof. It suffices to prove the "if"-part, since "only if" will follow from uniform substitution. Thus, we assume that $s' \in \operatorname{pre}\mathcal{K}_{s'}$. Since the prefix operator distributes over arbitrary union, this implies $s' \in \operatorname{pre}\mathcal{K}$ for some $\mathcal{K} \subseteq \mathcal{E}_{s'}$ that satisfies [K0] and [K1] for the abstraction $\mathcal{L}'$ with uncontrollable events Σ'_{uc}. By Proposition 2, there must exist a controller $\mathcal{H}'$ that is admissible to $\mathcal{L}'$ with $\mathcal{L}' \cap \mathcal{H}' = \mathcal{K} \subseteq \mathcal{E}_{s'}$. Referring

to $s' \in \operatorname{pre}\mathcal{K}$ and Eq. (6), we conclude $s'w \notin \mathcal{L}$ for all $w \in \Sigma^\omega$ with $s'w \in \mathcal{H}'$, i.e., $s' \in (\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\mathcal{H}')$ and $s' \notin \operatorname{pre}(\mathcal{L} \cap \mathcal{H}')$ as in Eq. (11). We construct a controller candidate $\tilde{\mathcal{H}}$ from $\mathcal{H}'$ as is Eq. (10). By Proposition 10 and referring to $s'[\equiv_{\mathcal{L}'}]s''$, the candidate $\tilde{\mathcal{H}}$ is admissible to $\mathcal{L}'$, and, by Proposition 2, $\tilde{\mathcal{K}} := \mathcal{L}' \cap \tilde{\mathcal{H}}$ satisfies [K0] and [K1] for the abstraction. Note also that $s'' \in \operatorname{pre}\mathcal{L}'$ and $s'' \in \operatorname{pre}\tilde{\mathcal{H}}$, by $s'[\equiv_{\mathcal{L}'}]s''$ and $s' \in \operatorname{pre}\mathcal{H}'$, respectively. With admissibility of $\tilde{\mathcal{H}}$ this implies $s'' \in \operatorname{pre}\tilde{\mathcal{K}}$. Referring to Proposition 11 with $s'[\equiv_{\mathcal{L}}]s''$, we obtain $s'' \in (\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\tilde{\mathcal{H}})$ and $s'' \notin \operatorname{pre}(\mathcal{L} \cap \tilde{\mathcal{H}})$. In particular, $s''w \notin \mathcal{L}$ for any $w \in \Sigma^\omega$ with $s'' \in \tilde{\mathcal{H}}$ and therefore $\tilde{\mathcal{K}} = \mathcal{L}' \cap \tilde{\mathcal{H}} \subseteq \mathcal{E}_{s''}$. Hence, $\tilde{\mathcal{K}}$ is a component in the union construction of $\mathcal{K}_{s''}$ and we conclude with $s'' \in \operatorname{pre}\tilde{\mathcal{K}} \subseteq \operatorname{pre}\mathcal{K}_{s''}$. □

Provided that both languages $\mathcal{L}$ and $\mathcal{L}'$ are given as deterministic finite automata realisations, we can again use a product composition to obtain a sufficiently rich state set such that the test $s \in \operatorname{pre}\mathcal{K}_s$ evaluates uniformly and, hence, can be applied per state. The test itself requires the computation of $\operatorname{pre}\mathcal{K}_s$, which is also known as the *controllability prefix*. The latter has been studied in [15, 16], including a computational procedure for deterministic automata with either Büchi acceptance condition or Rabin acceptance condition.

VII. CONCLUSION

This paper provides two characterisations of abstraction consistency in the context of supervisory control of sequential behaviours. First, we turned attention to plants with topologically closed behaviours and developed the sufficient and necessary condition [C1] that ensures that any controller designed for the abstraction will also faithfully operate the actual plant and do so without risking deadlocks. Second, we considered plants that may not be topologically closed and derived an according characterisation of consistency [C2]. Here, any abstraction based controller design is additionally guaranteed not to run the closed loop into a livelock. Regarding both conditions, we gave an outline for a software implementation that is applicable for the situation of regular behaviours, e.g., realised by deterministic Rabin automata.

REFERENCES

[1] C. Baier and T. Moor. A hierarchical and modular control architecture for sequential behaviours. *Discrete Event Dynamic Systems*, 25(1–2):95–124, 2015.

[2] L. Feng and W. M. Wonham. Supervisory control architecture for discrete-event systems. *IEEE Transactions on Automatic Control*, 53(6):1449–1461, 2008.

[3] R. Kumar, V. Garg, and S. I. Marcus. On supervisory control of sequential behaviors. *IEEE Transactions on Automatic Control*, 37(12):1978–1985, 1992.

[4] R. J. Leduc, M. Lawford, and W. M. Wonham. Hierarchical interface-based supervisory control-part ii: parallel case. *IEEE Transactions on Automatic Control*, 50(9):1336–1348, 2005.

[5] T. Moor. Natural projections for the synthesis of non-conflicting supervisory controllers. *Workshop on Discrete Event Systems (WODES)*, 2014.

[6] T. Moor and J. Raisch. Supervisory control of hybrid systems within a behavioural framework. *Systems and Control Letters*, 38:157–166, 1999.

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

[8] G. Pola, A. Girard, and P. Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[9] G. Pola and P. Tabuada. Symbolic models for nonlinear control systems: Alternating approximate bisimulations. *SIAM Journal on Control and Optimization*, 48(2):719–733, 2009.

[10] J. Raisch and S. O'Young. Discrete approximation and supervisory control of continuous systems. *IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems*, 43(4):569—-573, 1998.

[11] P. Ramadge. Some tractable supervisory control problems for discrete-event systems modeled by büchi automata. *IEEE Trans. Aut. Contr.*, 34 (1):10–19, 1989.

[12] 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.

[13] G. Reissig, A. Weber, and M. Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, 62, 2017.

[14] P. Tabuada. Verification and control of hybrid systems, a symbolic approach. *Springer Verlag*, 2009.

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

[16] J. G. Thistle and W.M. Wonham. Supervision of infinite behavior of discrete-event systems. *SIAM Journal on Control and Optimization*, 32 (4):1098–1113, 1994.

[17] K. C. Wong and W. M. Wonham. Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems*, 6(3):241–273, 1996.

APPENDIX

Proof of Proposition 2. Consider a controller $\mathcal{H}$ that is admissible to $\mathcal{L}$ and let $\mathcal{K} := \mathcal{L} \cap \mathcal{H}$. For relative closedness [K0], observe that $(\operatorname{clo}\mathcal{K}) \cap \mathcal{L} = (\operatorname{clo}(\mathcal{L} \cap \mathcal{H})) \cap \mathcal{L} \subseteq (\operatorname{clo}\mathcal{H}) \cap \mathcal{L} = \mathcal{H} \cap \mathcal{L} = \mathcal{K}$, where the converse inclusion is trivialy true. For controllability [K1], observe $((\operatorname{pre}\mathcal{K})\Sigma_{\mathrm{uc}}) \cap (\operatorname{pre}\mathcal{L}) = ((\operatorname{pre}(\mathcal{L} \cap \mathcal{H}))\Sigma_{\mathrm{uc}}) \cap (\operatorname{pre}\mathcal{L}) \subseteq ((\operatorname{pre}\mathcal{H})\Sigma_{\mathrm{uc}}) \cap (\operatorname{pre}\mathcal{L}) \subseteq (\operatorname{pre}\mathcal{H}) \cap (\operatorname{pre}\mathcal{L}) = \operatorname{pre}(\mathcal{H} \cap \mathcal{L}) = \operatorname{pre}\mathcal{K}$. This completes the first part of the proof.

For the second part, assume that $\mathcal{K}$ satisfy [K0] and [K1] and consider the candidate controller $\mathcal{H} = \lim((\operatorname{pre}\mathcal{K})\Sigma_{\mathrm{uc}}^*)$. Note that, by construction, $\mathcal{H}$ is the limit of a prefix-closed and complete language. In particular, $\mathcal{H}$ is topologically closed [H0] and

$$\operatorname{pre}\mathcal{H} = \operatorname{pre}\lim((\operatorname{pre}\mathcal{K})\Sigma_{\mathrm{uc}}^*) = (\operatorname{pre}\mathcal{K})\Sigma_{\mathrm{uc}}^* . \tag{13}$$

Hence $(\operatorname{pre}\mathcal{H})\Sigma_{\mathrm{uc}} \subseteq \operatorname{pre}\mathcal{H}$, i.e., we also have [H1]. We now turn to an inductive proof of the following hypothesis

$$((\operatorname{pre}\mathcal{K})\Sigma_{\mathrm{uc}}^i) \cap \ (\operatorname{pre}\mathcal{L}) \subseteq \operatorname{pre}\mathcal{K} , \tag{14}$$

which by [K0] holds for $i = 0$. Pick an i such that (14) holds, concatenate both sides with Σ_{uc} and intersect with $\operatorname{pre}\mathcal{L}$, to obtain

$$((\operatorname{pre}\mathcal{K})\Sigma_{\mathrm{uc}}^{i+1}) \cap ((\operatorname{pre}\mathcal{L})\Sigma_{\mathrm{uc}}) \cap (\operatorname{pre}\mathcal{L}) \subseteq ((\operatorname{pre}\mathcal{K})\Sigma_{\mathrm{uc}}) \cap (\operatorname{pre}\mathcal{L}) . \tag{15}$$

For the left-hand-side, note that $((\operatorname{pre}\mathcal{L})\Sigma_{\mathrm{uc}}) \cap (\operatorname{pre}\mathcal{L}) = (\Sigma^*\Sigma_{\mathrm{uc}}) \cap (\operatorname{pre}\mathcal{L})$ and that $(\operatorname{pre}\mathcal{K})\Sigma_{\mathrm{uc}}^{i+1} \subseteq \Sigma^*\Sigma_{\mathrm{uc}}$. For the right-hand-side, we refer to [K1]. Overall, we obtain

$$((\operatorname{pre}\mathcal{K})\Sigma_{\mathrm{uc}}^{i+1}) \cap (\operatorname{pre}\mathcal{L}) \subseteq \operatorname{pre}\mathcal{K} . \tag{16}$$

This completes the proof of Eq. (14) by induction. Taking unions over all $i \geq 0$, we obtain

$$((\operatorname{pre}\mathcal{K})\Sigma_{\mathrm{uc}}^*) \cap \ (\operatorname{pre}\mathcal{L}) \subseteq \operatorname{pre}\mathcal{K}. \tag{17}$$

The converse inclusion is obvious, hence, we have equality. Together with Eq. (13) this implies that

$$(\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\mathcal{H}) = \operatorname{pre}\mathcal{K} , \tag{18}$$

and, moreover, $\mathcal{K} = (\operatorname{clo}\mathcal{K}) \cap \mathcal{L} = \lim((\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\mathcal{H})) \cap \mathcal{L}$. For prefix-closed languages the limit distributes over intersection, and we continue with $\lim((\operatorname{pre}\mathcal{L}) \cap (\operatorname{pre}\mathcal{H})) \cap \mathcal{L} = (\operatorname{clo}\mathcal{L}) \cap (\operatorname{clo}\mathcal{H}) \cap \mathcal{L} = \mathcal{L} \cap \mathcal{H}$, i.e., $\mathcal{K} = \mathcal{L} \cap \mathcal{H}$, with non-conflictingness [H2] as an immediate consequence. □