Natural Projections for the Synthesis of Non-Conflicting Supervisory Controllers

Thomas Moor *

* *Lehrstuhl für Regelungstechnik*
Friedrich-Alexander Universität Erlangen-Nürnberg, Germany

Abstract: A common strategy in the design of discrete-event systems is to apply synthesis algorithms not to the actual plant model but to an abstraction that is realised on a significantly smaller state set. Depending on the control objectives, certain conditions are imposed on the plant and on the abstraction, in order to end up with an appropriate controller. A well known result from the literature is that abstractions obtained by a so called *natural observer* can be used for the purpose of non-blocking supervisory control. Despite additional favourable properties of natural observers regarding state count and composed plants, as a condition for non-blocking supervisory control it is restrictive, i.e., sufficient but not necessary. This contrasts the sufficient and necessary condition developed in this paper.

Keywords: Discrete-event systems, supervisory control, abstraction-based synthesis, liveness properties.

1. INTRODUCTION

When the plant model provides more detail than required for the controller design problem at hand, one may resort to an appropriate plant abstraction instead. A crucial question in such an *abstraction-based controller design* is whether the resulting controller enforces relevant control objectives not only for the abstraction but also for the original plant model.

More specifically, we consider the situation where the plant model is given as a formal language and the *natural projection* to strings of *high-level events* is considered as a candidate for an abstraction; see also (Feng and Wonham, 2008, 2010). This setting applies to the design of hierarchical control architectures when a group of plant components, each subject to low-level control (Ramadge and Wonham, 1987, 1989), are composed and the subsequent task is the synthesis of a supervisor that addresses cooperative behaviour, specified w.r.t. high-level events. For computational procedures, including the choice of a suitable high-level alphabet, see e.g. (Schmidt et al., 2008; Feng and Wonham, 2010). In the present paper, we rephrase the question, whether the abstraction-based design solves the original problem as a requirement imposed on the high-level alphabet and we develop an implementable test to verify this requirement.

Our study relates to (Wong and Wonham, 1996), where within a general framework the notion of an *observer* is defined and proven to be a sufficient condition for the purpose of non-blocking hierarchical controller synthesis. Variations of the *natural observer* property that explicitly take into account controllability are presented in (Feng and Wonham, 2008) and address minimal restrictive hierarchical supervision (Schmidt and Breindl, 2011). In (Malik et al., 2007), it is shown that the observer property is not only sufficient but also necessary to obtain a conflict equivalent abstraction used for *compositional non-blocking verification*. The present paper is a further development of the reachability analysis presented in (Moor et al., 2013). In contrast to the earlier results, the novel condition obtained in the present paper is not only sufficient but also necessary for *non-conflicting controller synthesis*, i.e., we characterise precisely those projections, for which an abstraction-based controller design is guaranteed to exhibit a non-conflicting closed-loop behaviour.

The paper is organised as follows. Preliminaries and notational conventions are given in Section 2 and prepare for the technical problem statement in Section 3. To obtain a characterisation of a non-conflicting closed-loop configuration, Section 4 relates individual conflicts to the minimal restrictive solution of a particular controller synthesis problem. Consequences for the situation of regular languages are drawn in Section 5 to provide the basis for a software implementation. Finally, Section 6 interprets the results in the context of the reachability analysis proposed in (Moor et al., 2013).

2. PRELIMINARIES AND NOTATION

Let Σ be a *finite alphabet*, i.e., a finite set of symbols $\sigma \in \Sigma$. The *Kleene-closure* Σ^* is the set of finite strings $s = \sigma_1\sigma_2\cdots\sigma_n$, $n \in \mathbb{N}$, $\sigma_i \in \Sigma$, 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 $r \leq s$.

A *formal language* (or short a *language*) over Σ is a subset $L \subseteq \Sigma^*$. Given a language $L \subseteq \Sigma^*$, the equivalence relation $[\equiv_L]$ on Σ^* is defined by $s' \,[\equiv_L]\, s''$ if and only if $(\forall\, t \in \Sigma^*)[\, s't \in L \leftrightarrow s''t \in L\,]$. The language L is *regular* if $[\equiv_L]$ has only finitely many equivalence classes.

The *prefix* of a language $L \subseteq \Sigma^*$ is defined by $\mathrm{pre}\, L := \{r \in \Sigma^* \mid \exists\, s \in L : r \leq s\}$. A language L is *prefix-closed* (or short *closed*) if $L = \mathrm{pre}\, L$. A language K is *relatively closed w.r.t. L* if $K = (\mathrm{pre}\, K) \cap L$. The languages L and K are *non-conflicting* if $\mathrm{pre}\,(L \cap K) = (\mathrm{pre}\, L) \cap (\mathrm{pre}\, K)$. The prefix operator distributes over arbitrary unions of languages.

For the *observable events* $\Sigma_o \subseteq \Sigma$, the *natural projection* $\mathrm{p_o}\colon \Sigma^* \to \Sigma_o^*$ is defined iteratively: (1) let $\mathrm{p_o}\epsilon := \epsilon$; (2) for $s \in \Sigma^*$, $\sigma \in \Sigma$, let $\mathrm{p_o}(s\sigma) := (\mathrm{p_o}s)\sigma$ if $\sigma \in \Sigma_o$, or, if $\sigma \notin \Sigma_o$, let $\mathrm{p_o}(s\sigma) := \mathrm{p_o}s$. The set-valued inverse $\mathrm{p_o^{-1}}$ of $\mathrm{p_o}$ is defined

by $\mathrm{p}_\mathrm{o}^{-1}(r) := \{s \in \Sigma^* \mid \mathrm{p}_\mathrm{o}(s) = r\}$ for $r \in \Sigma_\mathrm{o}^*$. When applied to languages, the projection distributes over unions, and the inverse projection distributes over unions and intersections. The prefix operator commutes with projection and inverse projection.

The projection $\mathrm{p}_\mathrm{o}\colon \Sigma^* \to \Sigma_\mathrm{o}^*$ is a *natural observer* for a language $L \subseteq \Sigma^*$, if for all $s \in \mathrm{pre}\, L$ and all $u \in \Sigma_\mathrm{o}^*$ with $(\mathrm{p}_\mathrm{o} s)u \in \mathrm{p}_\mathrm{o} L$ there exists $t \in \Sigma^*$ such that $st \in L$ and $\mathrm{p}_\mathrm{o} t = u$; see e.g. Feng and Wonham (2010).

Given two languages L, $K \subseteq \Sigma^*$, and a set of *uncontrollable events* $\Sigma_\mathrm{uc} \subseteq \Sigma$, we say K is *controllable w.r.t. L*, if $(\mathrm{pre}\, K)\Sigma_\mathrm{uc} \cap (\mathrm{pre}\, L) \subseteq \mathrm{pre}\, K$. Note that, in contrast to e.g. (Ramadge and Wonham, 1987) but in compliance with e.g. (Cassandras and Lafortune, 2008), this variant of controllability does not insist in $K \subseteq L$. Controllability, closedness and relative closedness are each retained under arbitrary union.

Unless otherwise noted, the alphabets Σ, Σ_c, Σ_uc, Σ_o and Σ_uo refer to the *common partitioning* $\Sigma = \Sigma_\mathrm{c} \dot\cup \Sigma_\mathrm{uc} = \Sigma_\mathrm{o} \dot\cup \Sigma_\mathrm{uo}$ in controllable, uncontrollable, observable and unobservable events, respectively.

3. PROBLEM STATEMENT

For the purpose of this paper, let the *plant* and the *controller* be represented by formal languages $L \subseteq \Sigma^*$ and $H \subseteq \Sigma^*$, respectively, to obtain the *closed-loop behaviour* $K \subseteq \Sigma^*$ by intersection, i.e., $K = L \cap H$. The following definition imposes conditions on the controller for a well-posed closed-loop configuration.

Definition 1. Given a *plant* $L \subseteq \Sigma^*$, $\Sigma = \Sigma_\mathrm{c} \dot\cup \Sigma_\mathrm{uc}$, a *controller* $H \subseteq \Sigma^*$ is *admissible w.r.t. L*, if

(i) H is prefix-closed;
(ii) H is controllable w.r.t. L; and,
(iii) L and H are non-conflicting. □

It is readily verified that a closed-loop behaviour $K \subseteq L$ can be achieved by an admissible controller H if and only if K is controllable w.r.t. L and relatively closed w.r.t. L. This corresponds to *non-blocking supervision* as originally proposed by Ramadge and Wonham (1987). There, control is exercised by a causal feedback map $V\colon \mathrm{pre}\, L \to \Gamma$, which maps the respective past string $s \in \mathrm{pre}\, L$ to a control pattern $\gamma = V(s)$, $\Sigma_\mathrm{uc} \subseteq \gamma \subseteq \Sigma$, to indicate the set of enabled successor events after the occurrence of s. In this paper, the controller H is interpreted as a representation of the feedback map V, and we omit explicit references to V in the subsequent development.

When a *language inclusion specification* $E \subseteq L$ is given, controller design amounts to the computation of the supremal achievable closed-loop behaviour $K^\uparrow \subseteq E$ in order to extract a corresponding controller $H := \mathrm{pre}\, K^\uparrow$; see e.g. (Wonham and Ramadge, 1987) for a computational procedure. Now consider the case, where the controller can only observe events from a restricted alphabet $\Sigma_\mathrm{o} \subseteq \Sigma$. This paper takes the perspective of *hierarchical control*, see e.g. (Wong and Wonham, 1996), where one motivation in the deliberate restriction of observable events is to gain computational benefits. In this setting, one may assume that any aspects of the specification that relates to unobservable events has been dealt with by a low-level controller and that the specification at hand exclusively refers to Σ_o, i.e., $E = \mathrm{p}_\mathrm{o}^{-1}\mathrm{p}_\mathrm{o} E$. It is then proposed to synthesise an admissible controller $H_\mathrm{o} \subseteq \Sigma_\mathrm{o}^*$ for the projected plant $L_\mathrm{o} := \mathrm{p}_\mathrm{o} L \subseteq \Sigma_\mathrm{o}^*$ to satisfy the projected specification $E_\mathrm{o} := \mathrm{p}_\mathrm{o} E$. In this approach, L_o is interpreted as an *abstraction* of the plant L, and, in turn, $H := \mathrm{p}_\mathrm{o}^{-1} H_\mathrm{o}$ as an *implementation* of the *high-level controller* H_o to operate on the actual plant L. By construction, we obtain

$$L \cap H \subseteq \mathrm{p}_\mathrm{o}^{-1}(L_\mathrm{o} \cap H_\mathrm{o}), \qquad L_\mathrm{o} \cap H_\mathrm{o} = \mathrm{p}_\mathrm{o}(L \cap H),$$

where the latter equality is referred to as *hierarchical consistency*; see also (Zhong and Wonham, 1990). In particular, the actual closed-loop behaviour $K = L \cap H$ satisfies the language inclusion specification:

$$K = L \cap H \subseteq \mathrm{p}_\mathrm{o}^{-1}(L_\mathrm{o} \cap H_\mathrm{o}) \subseteq \mathrm{p}_\mathrm{o}^{-1} E_\mathrm{o} = E\,.$$

It must be noted, that in the worst case the number of states required to realise L_o is even larger when compared to L; see (Wong, 1998). However, for relevant applications a substantial reduction of the required state set can be observed. In such a prospective situation, there remains the question whether admissibility of the high-level controller H_o implies admissibility of the implementation $H := \mathrm{p}_\mathrm{o}^{-1} H_\mathrm{o}$. This question is readily rephrased as a formal requirement imposed on the abstraction.

Definition 2. Given a plant $L \subseteq \Sigma^*$ with the common alphabet partitioning, the plant abstraction $L_\mathrm{o} := \mathrm{p}_\mathrm{o} L$ is *consistent for the purpose of controller design* (or short *consistent*), if admissibility is retained under implementation; i.e., if for all $H_\mathrm{o} \subseteq \Sigma_\mathrm{o}^*$, $H := \mathrm{p}_\mathrm{o}^{-1} H_\mathrm{o}$, the following implication holds:

$$\begin{aligned} & H_\mathrm{o} \text{ is admissible w.r.t. } L_\mathrm{o} \\ \Longrightarrow\quad & H \text{ is admissible w.r.t. } L\,. \end{aligned}$$ □

Regarding the individual properties closedness, controllability and non-conflictingness, we recall well-known facts from the literature.

Proposition 3. Given a plant $L \subseteq \Sigma^*$ with the common alphabet partitioning, consider the abstraction $L_\mathrm{o} := \mathrm{p}_\mathrm{o} L$, a controller candidate $H_\mathrm{o} \subseteq \Sigma_\mathrm{o}^*$ and its implementation $H := \mathrm{p}_\mathrm{o}^{-1} H_\mathrm{o}$. Then each of the following three implications holds true individually:

$$\begin{aligned} & H_\mathrm{o} \text{ is prefix-closed} \\ \Longrightarrow\quad & H \text{ is prefix-closed}\,; \end{aligned}$$

$$\begin{aligned} & H_\mathrm{o} \text{ is controllable w.r.t. } L_\mathrm{o} \\ \Longrightarrow\quad & H \text{ is controllable w.r.t. } L\,; \end{aligned}$$

and, provided that p_o is a natural observer for L,

$$\begin{aligned} & L_\mathrm{o} \text{ and } H_\mathrm{o} \text{ are non-conflicting} \\ \Longrightarrow\quad & L \text{ and } H \text{ are non-conflicting}\,. \end{aligned}$$

Proof. For the first implication, recall that the prefix operator $\mathrm{pre}\,(\cdot)$ and the projection $\mathrm{p}_\mathrm{o}(\cdot)$ commute. The second and the third implication are consequences of the more general results given in (Zhong and Wonham, 1990), Theorem 4.1, and in (Wong and Wonham, 1996), Theorem 6, respectively. For a direct proof addressing the specific situation at hand, see also (Moor et al., 2013). □

In particular, the above proposition identifies the natural observer property as a *sufficient condition* for the consistency of an abstraction.

Theorem 4. If for a plant $L \subseteq \Sigma^*$ with the common alphabet partitioning the projection $\mathrm{p}_\mathrm{o}\colon \Sigma^* \to \Sigma_\mathrm{o}^*$ is a natural observer, then the abstraction $L_\mathrm{o} := \mathrm{p}_\mathrm{o} L$ is consistent for the purpose of controller design. □

However, observe from Proposition 3 that the natural observer property implies a non-conflicting closed loop regardless whether the controller H_o is prefix-closed and/or satisfies the controllability requirement. Thus, the converse implication of the above theorem is suspected not to hold; i.e., a projection may fail to constitute a natural observer but still yield a consistent abstraction. This situation is illustrated by the following example, Fig. 1.

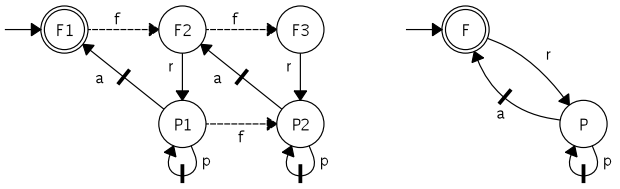

Fig. 1. L and L_o, resp., with $\Sigma_o = \{r, p, a\}$, $\Sigma_c = \{a, p\}$

The plant L has been extracted from a real-world application, where the event f represents the feed of a workpiece to a processing machine with built-in buffer of capacity two. Provided that a workpiece is present, the factory management may issue a request event r, and awaits an acknowledge a to forward the workpiece to a subsequent processing station. The design task at hand is to synthesise a processing controller that, on request r, first applies a recipe represented by a particular number of processing events p and then enables the acknowledgement event a. By intuition, the feed event f is not relevant for the controller design and one may propose the projection $L_o := p_o L$ with $\Sigma_o = \Sigma - \{f\}$ as a suitable abstraction.

To observe that the projection p_o fails to be a natural observer, consider the string $s = ffr$, which advances the plant to state P2 and the abstraction to state P. The abstraction suggests that the event a leads to a marked state. In contrast, the actual plant requires two a events for this purpose. Therefore, p_o is not a natural observer for L. Moreover, since f is the only unobservable event, no extension of Σ_o yields a natural observer that reduces the state count.

Note that the example is readily adapted to any fixed buffer capacity, generating arbitrarily large state counts for the actual plant while not affecting the abstraction. Thus, the reduction of the state count when using the abstraction for a controller design can be substantial. In the remainder of the present paper, a *sufficient and necessary* condition to characterise consistency is developed. In particular, the condition is satisfied for the above example and thereby justifies a controller design based on the proposed abstraction.

4. A CHARACTERISATION OF CONFLICTS

Consider a high-level controller $H_o \subseteq \Sigma_o^*$, admissible w.r.t. the plant abstraction $L_o := p_o L$, and its implementation $H := p_o^{-1} H_o$. By Proposition 3, H is prefix-closed and controllable w.r.t. L and we are left to discuss whether or not the closed-loop configuration is non-conflicting. Here, we use the terminology of a conflict at a particular string $s \in \operatorname{pre} L$ w.r.t. a target language $M \subseteq \operatorname{pre} L$, i.e., we say that the closed loop *conflicts at s w.r.t. M*, whenever $s \in (\operatorname{pre} L) \cap H$ and

$$(\forall\, t \in \Sigma^*)[\, st \in H \;\rightarrow\; st \notin M\,]\,.$$

To this end, the specific case of $M := L$ motivates our study: L and H are non-conflicting if and only if no string $s \in \operatorname{pre} L$ conflicts w.r.t. the target L. More general targets $M \subseteq \operatorname{pre} L$ will become relevant for the reachability analysis in Section 6.

Conflicts can be characterised be a language inclusion specification that prevents the execution of any extension that enters the target. Given $s \in \operatorname{pre} L$ and a target $M \subseteq \operatorname{pre} L$, denote all extensions of s that enter M by

$$M_s := M \cap (s\Sigma^*)\,, \tag{1}$$

and let

$$E_{os} := L_o - (p_o M_s)\Sigma_o^*\,; \tag{2}$$

see the below figure for an example continued from the previous section. If the high-level controller H_o complies with the specification E_{os}, it must disable all extensions of s that enter M, and, provided that s is in the local close-loop behaviour, this causes a conflict at s. The converse implication also holds.

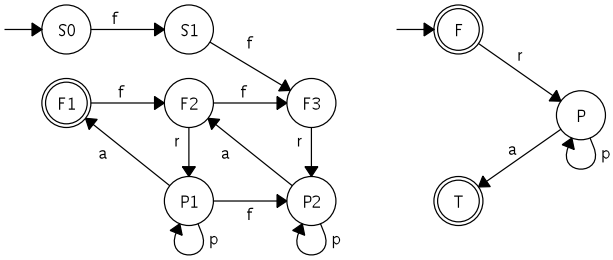

Fig. 2. M_s and E_{os}, resp., with $M = L$ and $s = ffr$

Lemma 5. Given a plant $L \subseteq \Sigma^*$ with the common alphabet partitioning and a target $M \subseteq \operatorname{pre} L$, let $H_o \subseteq \Sigma_o^*$ be admissible w.r.t. the abstraction $L_o := p_o L$ and consider the implementation $H := p_o^{-1} H_o$. Then, for any $s \in \operatorname{pre} L$, the following are equivalent:

(i) $L_o \cap H_o \subseteq E_{os}$ and $p_o s \in H_o$;
(ii) L and H conflict at s w.r.t. M.

Proof. We first establish "¬(ii) → ¬(i)". By "¬(ii)", we can choose $t \in \Sigma^*$ with $st \in M \cap H$. This implies $p_o(st) \in (\operatorname{pre} L_o) \cap H_o$, and, referring to admissibility of H_o w.r.t. L_o, we can choose $u \in \Sigma_o^*$ such that $p_o(st)u \in L_o \cap H_o$. However, $st \in M \cap s\Sigma^*$, and, hence, $p_o(st)u \in (p_o M_s)\Sigma_o^*$. Therefore $p_o(st)u \notin E_{os}$. We turn to the converse implication "(ii) → (i)". The conflict at s, by definition, implies $s \in (\operatorname{pre} L) \cap H$ and, thus, $p_o s \in p_o H = H_o$. Now, pick an arbitrary $r \in L_o \cap H_o$, and, for a proof by contradiction, assume that $r \notin E_{os}$. This implies $r \in (p_o M_s)\Sigma_o^*$, and we can choose $v \in M \cap (s\Sigma^*)$ and $w \in \Sigma^*$ such that $p_o(vw) = r$. Rewrite v as $v = st \in M$ and observe $stw \in p_o^{-1} r \subseteq p_o^{-1} H_o = H$. This implies $st \in H \cap M$ and therefore contradicts with (ii). We conclude $r \in E_{os}$ and, hence, $L_o \cap H_o \subseteq E_{os}$. □

The characterisation of a conflict by a language inclusion specification has the particular benefit that the existence of a controller that leads to the conflict can be verified by inspecting the supremal high-level closed-loop behaviour

$$K_{os}^{\uparrow} := \sup\{\, K_o \subseteq E_{os} \mid K_o \text{ is controllable and relatively closed w.r.t. } L_o \,\} \tag{3}$$

that satisfies E_{os}.

Lemma 6. Consider a plant $L \subseteq \Sigma^*$ with the common alphabet partitioning, a string $s \in \operatorname{pre} L$, a target $M \subseteq \operatorname{pre} L$ and the abstraction $L_o := p_o L$. Then there exists a high-level controller

H_o, admissible w.r.t. L_o, such that L and $H := p_o^{-1} H_o$ conflict at s w.r.t. M, if and only if $p_o s \in \operatorname{pre} K_{os}^{\uparrow}$.

Proof. First, assume that $p_o s \in \operatorname{pre} K_{os}^{\uparrow}$. Then $H_o := \operatorname{pre} K_{os}^{\uparrow}$ is admissible w.r.t. L_o with $L_o \cap H_o \subseteq E_{os}$ and $p_o s \in H_o$. By Lemma 5, L and H conflict at s w.r.t. M. For the converse implication, consider an arbitrary admissible high-level controller H_o with implementation $H := p_o^{-1} H_o$ and assume that L and H conflict at s w.r.t. M. Referring to Lemma 5, we have $K_o := L_o \cap H_o \subseteq E_{os}$. Admissibility of H_o implies that K_o is controllable and relatively closed w.r.t. L_o. Thus, $K_o \subseteq K_{os}^{\uparrow}$. Since L and H conflict at s, we must have $s \in (\operatorname{pre} L) \cap H$, and, hence, $p_o s \in p_o((\operatorname{pre} L) \cap H) \subseteq (\operatorname{pre} L_o) \cap H_o = \operatorname{pre} K_o$. Thus, we obtain $p_o s \in \operatorname{pre} K_o \subseteq \operatorname{pre} K_{os}^{\uparrow}$. □

For the example and with $s = \mathrm{ffr}$, it turns out that $K_{os}^{\uparrow} = \emptyset$. Thus, no abstraction-based design exhibits a conflict at $s = \mathrm{ffr}$. By quantification over all $s \in \operatorname{pre} L$ and with target $M := L$ we obtain a characterisation of a non-conflicting closed-loop configuration.

Theorem 7. Given a plant $L \subseteq \Sigma^*$ with the common alphabet partitioning, consider the target $M := L$ and the abstraction $L_o := p_o L$. Then L_o is consistent for the purpose of controller design, if and only if $p_o s \notin \operatorname{pre} K_{os}^{\uparrow}$ for all $s \in \operatorname{pre} L$.

Proof. First, assume that L_o is consistent for the purpose of controller design and pick an arbitrary $s \in \operatorname{pre} L$. By consistency, any implementation $H = p_o^{-1} H_o$ of an admissible high-level controller H_o is admissible, and, in particular, forms a non-conflicting closed loop with the plant L. Thus, there exists no admissible high-level controller H_o, such that the implementation H and the plant L conflict at s. By Lemma 6, this implies $p_o s \notin \operatorname{pre} K_{os}^{\uparrow}$. For the converse implication, assume that $p_o s \notin \operatorname{pre} K_{os}^{\uparrow}$ for all $s \in \operatorname{pre} L$, pick an arbitrary high-level controller H_o, admissible w.r.t. L_o, and consider the implementation $H = p_o^{-1} H_o$. From Proposition 3, we have that H is prefix closed and controllable w.r.t. L. To establish that L and H are non-conflicting, pick an arbitrary $s \in (\operatorname{pre} L) \cap H$. By Lemma 6, L and H do not conflict at s. Thus, there exists $t \in \Sigma^*$ such that $st \in L \cap H$. Since s was chosen arbitrarily, this implies that L and H are non-conflicting. In particular, H is admissible w.r.t. L. Since H_o was chosen arbitrarily, this concludes the proof of consistency. □

5. CONSEQUENCES FOR REGULAR LANGUAGES

The characterisation of consistency by Theorem 7 replaces the universal quantification over all admissible high-level controllers from Definition 2 by the quantification over all strings from the local plant behaviour. To address a possible software implementation for the situation of regular languages, we will show that it suffices to test one representative for each class from a specifically chosen equivalence relation.

The following technical lemma shows that the control exercised for one string can be mapped to another one, provided that both strings are equivalent w.r.t. the plant behaviour.

Lemma 8. Given a plant $L \subseteq \Sigma^*$, $\Sigma = \Sigma_c \dot\cup \Sigma_{uc}$, consider a string $s' \in \operatorname{pre} L$ and a controller $H' \subseteq \Sigma^*$, admissible w.r.t. L, with $s' \in H'$. For s'', $s'' \,[\equiv_L]\, s'$, let

$$H'' = \{s \mid s \in \Sigma^*,\ s'' \notin \operatorname{pre} s\} \cup \{s''t \mid s't \in H'\}.$$

Then H'' is admissible w.r.t. L and $s'' \in (\operatorname{pre} L) \cap H''$.

Proof. Thinking of a closed language as an infinite tree with root ϵ, the left component of H'' consists of all strings in Σ^* that do not pass s'', whilst the right component consists of s'' and the sub-tree of H' corresponding to the nodes reachable from s'. The claims $s'' \in \operatorname{pre} L$ and $s'' \in H''$ follow from $s'' \,[\equiv_L]\, s'$ and the definition of H'', respectively. Ad closedness. Pick an arbitrary $s \in H''$ and consider a prefix $v \leq s$. If $s'' \leq v$, then s is chosen from the right component of H'' and we obtain $v \in H''$ from closedness of H'. If $s'' \not\leq v$, we have $s'' \notin \operatorname{pre} v$ and v is within the left component of H''. Ad controllability. Pick arbitrary $s \in H''$ and $\sigma \in \Sigma_{uc}$ such that $s\sigma \in \operatorname{pre} L$. If $s'' \leq s$, then s is chosen from the right component of H'', and we may write $s = s''t$ with $t \in \Sigma^*$ such that $s't \in H'$. Since $s'' \,[\equiv_L]\, s'$, we also obtain $s't\sigma \in \operatorname{pre} L$. Thus, controllability of H' implies $s't\sigma \in H'$ and therefore $s\sigma = s''t\sigma \in H''$. If, on the other hand, $s'' \not\leq s$, we have either $s'' \not\leq s\sigma$ or $s\sigma = s''$. In the first case, $s\sigma$ is within the left component of H''. In the second case, $s\sigma$ is within the right component of H''. Ad non-conflictingness. Pick an arbitrary $s \in (\operatorname{pre} L) \cap H''$. If $s'' \leq s$, then s is chosen from the right component of H'', and we may write $s = s''t$ for $s't \in H'$. Since $s'' \,[\equiv_L]\, s'$, we also obtain $s't \in \operatorname{pre} L$. By non-conflictingness of L and H', we can choose v such that $s'tv \in L \cap H'$. This implies $s''tv \in L \cap H''$, where we again refer to $s'' \,[\equiv_L]\, s'$ and the right component of H''. If, on the other hand, $s'' \not\leq s$, we choose an arbitrary $t \in \Sigma^*$ such that $st \in L$. Here, we distinguish two cases. First, assume that $s'' \not\leq st$, to obtain $st \in H''$ from the left component of H''. For the second case, we have $s'' \leq st$ and can write $svw = st$ with $v,\ w \in \Sigma^*$ such that $sv = s''$. In particular, we have $sv \in (\operatorname{pre} L) \cap H''$ and argue as for the situation of $s'' \leq s$. □

Mapping the control exercised at a particular string also maps respective conflicts.

Lemma 9. Given a plant $L \subseteq \Sigma^*$ with the common alphabet partitioning, consider a high-level controller H_o', admissible w.r.t. $L_o := p_o L$, such that L and $H' := p_o^{-1} H_o'$ conflict at $s' \in \operatorname{pre} L$ w.r.t. M, for $M \subseteq \operatorname{pre} L$. Let $s'' \in \Sigma^*$ with $s'' \,[\equiv_L]\, s'$, $p_o s'' \,[\equiv_{L_o}]\, p_o s'$, and $s'' \,[\equiv_M]\, s'$. Then there exists a high-level controller H_o'' such that L and $H'' := p_o^{-1} H_o''$ conflict at s'' w.r.t. M.

Proof. Let $r' := p_o s'$, $r'' := p_o s''$ and observe that $r' \in (\operatorname{pre} L_o) \cap p_o H' = (\operatorname{pre} L_o) \cap H_o'$. We apply Lemma 8 to H_o' to obtain

$$H_o'' = \{r \mid r \in \Sigma^*,\ r'' \notin \operatorname{pre} r\} \cup \{r''u \mid r'u \in H_o'\}$$

as a controller admissible w.r.t. L_o with $r'' \in (\operatorname{pre} L_o) \cap H_o''$. Denote the implementation $H'' := p_o^{-1} H_o''$, to observe $s'' \in (\operatorname{pre} L) \cap H''$. For a proof by contradiction, assume that L and H'' do not conflict at s'', i.e., we can choose $t \in \Sigma^*$ such that $s''t \in M \cap H''$. By $s'' \,[\equiv_M]\, s'$, we obtain $s't \in M$. Moreover, $r'' p_o t \in H_o''$, and, thus, $r' p_o t \in H_o'$. This implies $s't \in H'$, and we obtain $s't \in L \cap H'$. This contradicts with L and H' to conflict at s'. Therefore, the above choice of t can not be made and L and H'' must conflict at s''. □

By Lemmata 6 and 9, the condition $p_o s \in \operatorname{pre} K_{os}^{\uparrow}$ is either satisfied or dissatisfied uniformly for all strings that are equivalent with s.

Lemma 10. Given a plant $L \subseteq \Sigma^*$ with the common alphabet partitioning, and a target $M \subseteq \operatorname{pre} L$, consider two strings s', $s' \in \Sigma^*$ such that $s'' \,[\equiv_L]\, s'$, $s'' \,[\equiv_{p_o^{-1} L_o}]\, s'$ and $s'' \,[\equiv_M]\, s'$. Then $p_o s' \in \operatorname{pre} K_{os'}^{\uparrow}$ if and only if $p_o s'' \in \operatorname{pre} K_{os''}^{\uparrow}$

Proof. Let $r' := p_o s'$, $r'' := p_o s''$. Referring to $s'' \,[\equiv_{p_o^{-1} L_o}]\, s'$, it is readily verified that $r'' \,[\equiv_{L_o}]\, r'$. Now assume that $p_o s' \in$

pre $K^{\uparrow}_{\mathrm{o}s'}$. By Lemma 6 this implies the existence of an high-level controller H'_{o}, admissible w.r.t. L_{o} such that the plant L and the implementation $H' := \mathrm{p}_{\mathrm{o}}^{-1} H'_{\mathrm{o}}$ conflict at s'. Then, by Lemma 9, there exists a high-level controller H''_{o} such that L and $H'' := \mathrm{p}_{\mathrm{o}}^{-1} H''_{\mathrm{o}}$ conflict at s''. Again by Lemma 6, we obtain that $\mathrm{p}_{\mathrm{o}} s'' \in \mathrm{pre}\, K^{\uparrow}_{\mathrm{o}s''}$. The converse implication follows likewise. □

As a consequence of Lemma 10, the condition $\mathrm{p}_{\mathrm{o}} s \in \mathrm{pre}\, K^{\uparrow}_{\mathrm{o}s}$ in the characterisation of consistency in Theorem 7 only needs to be evaluated for one representative per equivalence class for an equivalence at least as fine as $[\equiv_L]$ and $[\equiv_{\mathrm{p}_{\mathrm{o}}^{-1} L_{\mathrm{o}}}]$. Provided that L is regular, such an equivalence can be represented by the product of two automata, one realising L and one realising $\mathrm{p}_{\mathrm{o}}^{-1} L_{\mathrm{o}}$. Here, each state corresponds to a set of strings that are equivalent w.r.t. both, $[\equiv_L]$ and $[\equiv_{\mathrm{p}_{\mathrm{o}}^{-1} L_{\mathrm{o}}}]$.

A computational procedure to test whether L_{o} is consistent can then be implemented as an iteration over all states of the product automaton, where at each state $\mathrm{pre}\, K^{\uparrow}_{\mathrm{o}s}$ is evaluated to test for $\mathrm{p}_{\mathrm{o}} s \in \mathrm{pre}\, K^{\uparrow}_{\mathrm{o}s}$. The overall test is passed when the condition is satisfied at all states. For the example from Section 4, the iteration can be performed on the plant state set, where consistency is successfully verified.

For abstractions that fail the test at particular states, one may restrict the actual plant L by a *low-level controller* H admissible w.r.t. L to render the critical states unreachable and to re-evaluate the condition in Theorem 7 with $K = L \cap H$ in the role of the actual plant. This approach effectively sacrifies specific capabilities of the plant in order to maintain a prescribed projection for the high-level controller design.

6. REACHABILITY ANALYSIS

We interpret the characterisation of consistency by Theorem 7 in the context of the reachability analysis proposed in (Moor et al., 2013). The given reference presents sufficient conditions to identify strings from the local plant behaviour that, under supervision by any abstraction based controller, exhibit at least one extension that enters a given target language. Technically, the discussion in (Moor et al., 2013) is stated in terms of so called *universal star-reachability operators*:

Definition 11. Given a plant $L \subseteq \Sigma^*$ with the common alphabet partitioning, denote $L_{\mathrm{o}} := \mathrm{p}_{\mathrm{o}} L$ the plant abstraction. An operator Ω with $\Omega(M) \subseteq \mathrm{pre}\, L$ for $M \subseteq \mathrm{pre}\, L$, is a *universal star-reachability operator*, if, for all $M \subseteq \mathrm{pre}\, L$ and all controllers $H_{\mathrm{o}} \subseteq \Sigma_{\mathrm{o}}^*$ admissible w.r.t. L_{o}, it holds that

$$(\forall\, s \in \Omega(M) \cap \mathrm{p}_{\mathrm{o}}^{-1} H_{\mathrm{o}})(\exists\, t \in \Sigma^*)[\, st \in M \cap \mathrm{p}_{\mathrm{o}}^{-1} H_{\mathrm{o}}\,].$$

□

Note that, (Moor et al., 2013) refers to a slightly stronger notion of admissibility. However, two of the proposed operators are also relevant for the present paper and we recall their respective definitions. Given a plant $L \subseteq \Sigma^*$ and the common alphabet partitioning, the operators Ω_{A} and Ω_{B} are defined by

$$\Omega_{\mathrm{A}}(M) := \{\, s \in \mathrm{pre}\, L \mid \exists\, \sigma \in \Sigma_{\mathrm{uo}} : s\sigma \in M \,\}, \tag{4}$$

$$\Omega_{\mathrm{B}}(M) := \{\, s \in \mathrm{pre}\, L \mid \exists\, \sigma \in \Sigma_{\mathrm{uc}} : s\sigma \in M \,\}, \tag{5}$$

for $M \subseteq \mathrm{pre}\, L$. Since no implementation of an admissible controller can disable unobservable or uncontrollable events, both operators are indeed universal star-reachability operators; the proof of Proposition IV.3, (Moor et al., 2013), applies literally to the situation in the present paper.

However, the operator Ω_{C} from (Moor et al., 2013) does not apply to the present paper, and we define the following alternative variant for Ω_{D} for $M \subseteq \mathrm{pre}\, L$:

$$\Omega_{\mathrm{F}}(M) := \{ s \in \mathrm{pre}\, L \mid (\exists\, \sigma \in \Sigma : s\sigma \in M) \text{ and } (\mathrm{p}_{\mathrm{o}} s \notin \mathrm{pre}\, K^{\uparrow}_{\mathrm{o}s} \text{ as def. by Eqs. (1)–(3)}) \}. \tag{6}$$

As a consequence of Lemma 6, Ω_{F} is a universal star-reachability operator.

Proposition 12. Given a plant $L \subseteq \Sigma^*$ with the common alphabet partitioning, denote $L_{\mathrm{o}} := \mathrm{p}_{\mathrm{o}} L$ the plant abstraction. Then Ω_{F} is a universal star-reachability operator.

Proof. Choose $M \subseteq \mathrm{pre}\, L$ and a controller $H_{\mathrm{o}} \subseteq \Sigma_{\mathrm{o}}^*$ that is admissible w.r.t. L_{o}, both arbitrarily. For $s \in \Omega_{\mathrm{F}}(M)$ we have by definition $s \in \mathrm{pre}\, L$ and $\mathrm{p}_{\mathrm{o}} s \notin \mathrm{pre}\, K^{\uparrow}_{\mathrm{o}s}$. The latter clause implies by Lemma 6, that L and $H := \mathrm{p}_{\mathrm{o}}^{-1} H_{\mathrm{o}}$ do not conflict at s. Assuming in addition $s \in H$, there must exists $t \in \Sigma^*$ such that $st \in L \cap H$. Thus, Ω_{F} satisfies the requirements of Definition 11. □

It follows immediately from Definition 11, that arbitrary unions of universal reachability operators are again universal reachability operators. Moreover, the identity is a universal reachability operator. Finally, adapting (Moor et al., 2013), Proposition IV.4, to the situation of the present paper, iterations of universal reachability operators are again universal reachability operators. Thus, we can freely combine Ω_{A}, Ω_{B} and Ω_{F} in nested iterations to obtain a less restrictive universal reachability operator. Compared with Ω_{A} and Ω_{B}, the evaluation of Ω_{F} is computationally more expensive. Therefore, it is proposed to alternate fixpoint iterations of Ω_{A} and Ω_{B} in an inner loop with an extension of the target M by applying Ω_{F} in an outer loop.

Algorithm 1. Verification of consistency

```
function IsConsistent(L, Σc, Σuc, Σo, Σuo)
    L_o ← p_o L
    M ← L
    repeat
        repeat
            M_recent ← M
            M ← M ∪ Ω_A(M)
            M ← M ∪ Ω_B(M)
        until M == M_recent
        M_recent ← M
        M ← M ∪ Ω_F(M)
    until M == M_recent
    return M == pre L
end function
```

Theorem 13. Given a regular plant $L \subseteq \Sigma^*$ with the common alphabet partitioning, both repeat-until loops in the function IsConsistent$(L, \Sigma_{\mathrm{c}}, \Sigma_{\mathrm{uc}}, \Sigma_{\mathrm{o}}, \Sigma_{\mathrm{uo}})$ terminate after finitely many iterations. Moreover, the return value is *true* if and only if the abstraction $L_{\mathrm{o}} := \mathrm{p}_{\mathrm{o}}^{-1} L$ is consistent for the purpose of controller design.

Proof. Observe that, during the iteration, M is monotonously increasing and bounded by $\mathrm{pre}\, L$. Thus, $L \subseteq M \subseteq \mathrm{pre}\, L$ is a loop invariant. Now assume that at some stage we have

$$(\forall\, s', s'' \in \Sigma^*)[\, s'' [\equiv_L] s',\ s'' [\equiv_{\mathrm{p}_{\mathrm{o}}^{-1} L_{\mathrm{o}}}] s' \ \rightarrow\ s'' [\equiv_M] s'\,]. \tag{7}$$

Observe by Lemma 10 of this paper and by Proposition V.1 from (Moor et al., 2013) that Eq. (7) is retained when applying either reachability operator. Since Eq. (7) is true upon initialisation with $M = L$, Eq (7) is a loop invariant, and there is

an upper bound on the number of states required to realise M. Thus, during the iteration, M can only take a finite number of different values. Monotonicity then implies that a fixpoint is reached after a finite number of iterations. For correctness of the algorithm, recall from the discussion above, that the two nested loops evaluate a universal star-reachability operator with result in the variable M at line 13. If the condition $M == \operatorname{pre} L$ is satisfied, consistency is implied by Definition 11. Now assume that the condition $M == \operatorname{pre} L$ is not satisfied. Since $M \subseteq \operatorname{pre} L$, we can then choose $s \in \operatorname{pre} L$, $s \notin M$. By $s \in \operatorname{pre} L$ and $L \subseteq M$, s can be extended to a string in M. Thus, we may assume without loss of generality that our choice of s also satisfies $s\Sigma \cap M \neq \emptyset$. The exit condition of the outer loop in line 12 implies $M = \Omega_F(M) \cup M$, and, thus, $s \notin \Omega_F(M)$. By $s\Sigma \cap M \neq \emptyset$, we obtain $\mathrm{p_o} s \in \operatorname{pre} K^{\uparrow}_{\mathrm{o}s}$. Referring to Lemma 6 , there must exist an addmissible high-level controller H_o with implementation $H := \mathrm{p_o^{-1}} H_\mathrm{o}$ such that the closed loop has a conflict w.r.t. M at s. Since we have $L \subseteq M$ this also constitutes a conflict w.r.t. L and, hence, the abstraction is not consistent. □

We have applied Algorihtm 1 to the processing machine example; see Fig. 1. The below automata Fig. 3–6 mark the iterate variable M on non-trivial updates during the iteration.

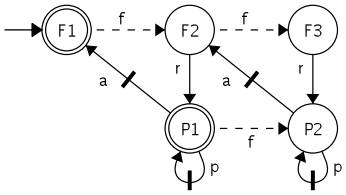

Fig. 3. First evaluation of Ω_F (line 11)

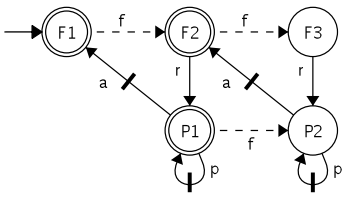

Fig. 4. Second inner loop, evaluation of Ω_B (line 8)

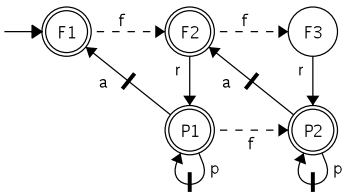

Fig. 5. Second evaluation of Ω_F (line 11)

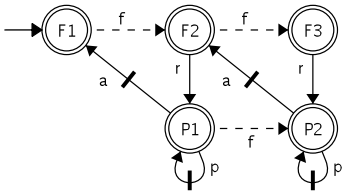

Fig. 6. Third inner loop, evaluation of Ω_B (line 8)

The first update to a different value occurs after the evaluation of Ω_F in line 11. After the last update, Fig. 6, we have $M = \operatorname{pre} L$ to satisfy the termination condition by the next iteration. In particular, the function returns with value *true* to indicate that the abstraction is consistent. Recall that the example does not possess the natural observer property and that by increasing the buffer capacity of the processing machine, the reduction in the state count by the proposed projection becomes significant.

CONCLUSION

We have developed a sufficient and necessary condition to guarantee that abstraction-based controller design yields a non-conflicting closed-loop configuration. In the development of our characterisation of consistency we eliminate the universal quantification over all admissible high-level controllers and thereby obtain a test that can be practically evaluated for the situation of regular plant behaviours. Necessity of our condition indicates that the result is not more restrictive than known sufficient conditions, including the well-studied observer properties. On the downside, our condition does not share the additional benefits known for natural observers, such as the guarantee for a reduced state count and an efficient verification for composed plants. From a practical perspective, one may interpret our result as a last resort for situations where natural observers are not applicable. The relevance of such situations is demonstrated by an example.

REFERENCES

Cassandras, C.G. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer, second edition.

Feng, L. and Wonham, W.M. (2010). On the computation of natural observers in discrete-event systems. *Discrete Event Dynamic Systems*, 20, 63–102.

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

Malik, R., Flordal, H., and Pena, P. (2007). Conflicts and projections. *1st IFAC Workshop on Dependable Control of Discrete Systems (DCDS)*, 63–68.

Moor, T., Baier, C., and Wittmann, T. (2013). Consistent abstractions for the purpose of supervisory control. *52nd IEEE Conference on Decision and Control*, 7291–7296.

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

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

Schmidt, K. and Breindl, C. (2011). Maximally permissive hierarchical control of decentralized discrete event systems. *IEEE Transactions on Automatic Control*, 56(4), 723–737.

Schmidt, K., Moor, T., and Perk, S. (2008). Nonblocking hierarchical control of decentralized discrete event systems. *IEEE Transactions on Automatic Control*, 53(10), 2252–2265.

Wong, K.C. (1998). On the complexity of projections of discrete-event systems. In *In IEE Workshop on Discrete Event Systems*, 201–208.

Wong, K.C. and Wonham, W.M. (1996). Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems*, 6, 241–306.

Wonham, W.M. and Ramadge, P.J. (1987). On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25, 637–659.

Zhong, H. and Wonham, W.M. (1990). On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control*, 35, 1125–1134.