Abstraction Based Supervisory Control for Non-Regular $*$-Languages

Lukas Triska * **Thomas Moor** *

* *Lehrstuhl für Regelungstechnik, Friedrich-Alexander Universität Erlangen-Nürnberg, Germany (e-mail: lrt@fau.de)*

Abstract: The problem of supervisory control, in its most basic form, is well understood for regular parameters, i.e., with regular languages for plant and upper-bound language-inclusion specification. However, when dropping the requirement of regularity, the situation becomes much more involved: even decidability becomes an issue. In this paper, we propose a simplistic but yet fairly general approach in that non-regular parameters are replaced by regular substitutes. In a language based setting we provide (a) conditions such that any solution found for the substitute parameters carries over to the original problem and (b) a refinement scheme which can be applied when the substitute parameters render the control problem at hand unsolvable. For practical purposes, we further elaborate our approach for plants represented as unbounded Petri nets and (c) identify a condition under which the alternation of trial synthesis and refinement is guaranteed to terminate with success.

Keywords: supervisory control, abstraction based controller design, controllability prefix, abstraction refinement, unbounded Petri nets.

INTRODUCTION

Supervisory control theory, as originally introduced by Ramadge and Wonham (1987, 1989), is a framework for the control of discrete-event systems. In its most basic form, the plant is represented by a *deterministic finite automaton* and an upper-bound specification is given by a *regular $*$-language*. Essentially, the control problem can be parametrised by two regular $*$-languages L and E over a common alphabet Σ to represent plant and specification, respectively. Achievable closed-loop behaviours K that satisfy the specification are algebraically characterised in terms of L and E. Specifically, $K \subseteq E$ must be *controllable* and *relatively closed*, both w.r.t. L. Based on this characterisation, there are well understood procedures to decide whether the problem has a solution, and, if so, to effectively compute such a solution; see (Wonham and Ramadge, 1987). Inspecting the cited literature, neither the problem statement nor the characterisation of the achievable closed-loop behaviours relate to the regularity of the parameters L and E. Hence, the problem is well posed even if either of the two parameters fails to be regular and the characterisation of all solutions remains valid. However, the computational procedures proposed by Wonham and Ramadge (1987) specifically address finite automata realisations and require adaptation when other types of realisations are to be considered.

Extending the practical scope of supervisory control theory, we are interested in computational procedures that address more general classes of $*$-languages not representable by finite automata. In this regard, Giua and DiCesare (1995, 1994) consider the languages L and E realised by deterministic Petri nets with different acceptance conditions, referred to as *L-type* and *G-type*, or *marked behaviour* and *weak behaviour*, respectively. Including the case of unbounded nets, i.e. nets with a countably infinite reachability graph, L and E are not necessarily regular. Controllability of the synchronous composition of a specification with the plant turns out decidable for both acceptance conditions. However, the trimming procedure proposed for controller synthesis may fail for unbounded nets with G-type acceptance condition; see (Giua, 2013; Holloway et al., 1997) for an overview and more references on supervisory control for Petri net models. Another branch of generalisation to non-regular languages addresses *deterministic context free $*$-languages* which can be realised by *deterministic pushdown automata*. Masopust (2012) and Sreenivas (1993) show, that the question whether a language K is controllable w.r.t. L, becomes undecidable when both parameters are deterministic context free $*$-languages. Schmuck et al. (2016) constructively show how to solve the control problem when only the specification E is a deterministic context free $*$-language but L is regular. The cited literature makes evident that regarding practical solutions and even regarding decidability, supervisory control becomes substantially more involved when dropping regularity.

In the present paper, we take a conceptionally more simplistic approach. Rather than to seek for procedures that solve the control problem directly, we propose to substitute not necessarily regular parameters L and E by regular abstractions L' and E' such that any solution found for the substitute parameters also solves the original problem. If no solution exists for L' and E', the abstractions need to be refined. Thus, we end up with a possibly non-terminating alternation of trial synthesis and abstraction refinement. However, in the case of termination it is guaranteed that a valid solution has been found. This approach is well established for the control of hybrid systems; see e.g. Raisch and O'Young (1998); Stursberg (2006); Yang et al. (2020). Obviously, we can not be successful, if specific aspects of the original problem parameters intrinsically rely on a non-regular representation. However, in practical applications

⋆ This work was supported by the German Research Foundation (DFG) under grant MO 1697/4-1.

this may not be the case. Consider e.g. a manufacturing system which at an early stage of planning may include components with infinite capacity to buffer or process workpieces. Once we put in place a specification for the intended behaviour, this should implicitly express that only finite capacities are required — infinite- capacities can not be technologically realised anyway. Hence, we may expect regular abstractions of the problem parameters to exist and appreciate a systematic procedure to reveal such abstractions. Our main technical contributions to this end are the identification of conditions that ensure solutions found for the substitute parameters to carry over to the original problem and a fairly general scheme of abstraction refinement, all stated purely in terms of $*$-languages. For practical applications, the proposed abstraction refinement scheme needs to be further elaborated in terms of some finitary representation of the non-regular parameters. In this regard, we focus attention to the plant being given as an unbounded Petri net with L-type acceptance condition and a regular specification.

The paper is organised as follows. After providing notational conventions in Section 1, we recall the basic problem of supervisory control in Section 2 and elaborate conditions of substitute problem parameters L' and E' such that solutions found for the abstraction carry over to the original problem. Refinement of abstractions are discussed in a purely language based setting in Section 3. We further elaborate our results in Sections 4 and 5, where we assume the plant to be represented as an unbounded Petri net and the specification to be regular.

1. PRELIMINARIES AND NOTATION

We largely follow the same notational conventions as Cassandras and Lafortune (2008); Wonham and Cai (2019), and, regarding Petri nets, Giua (2013).

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 $r \leq s$; if in addition $r \neq s$, we say r is a *strict prefix* of s and write $r < s$.

A $*$-*language* (or short a *language*) over Σ is a subset $L \subseteq \Sigma^*$. The *prefix* of a language $L \subseteq \Sigma^*$ is defined by $\mathrm{pfx}\, L := \{r \in \Sigma^* \mid \exists\, s \in L : r \leq s\}$. The prefix operator is also referred to as the *prefix-closure*, and, a language L is *closed* if $L = \mathrm{pfx}\, L$. A language K is *relatively closed w.r.t.* L if $K = (\mathrm{pfx}\, K) \cap L$. The prefix operator distributes over arbitrary unions of languages. However, for the intersection of two languages L and M, we have $\mathrm{pfx}\,(L \cap M) \subseteq (\mathrm{pfx}\, L) \cap (\mathrm{pfx}\, M)$. If equality holds, L and M are said to be *non-conflicting*. Given a language $L \subseteq \Sigma^*$, the *Nerode equivalence* $[\equiv_L]$ is an equivalence relation on Σ^* defined by $s'\,[\equiv_L]\,s''$ if and only if $(\forall\, t \in \Sigma^*)[\, s't \in L \leftrightarrow s''t \in L\,]$. Given two languages $K \subseteq L \subseteq \Sigma^*$, and a set of *uncontrollable events* $\Sigma_{\mathrm{uc}} \subseteq \Sigma$, we say K is *controllable w.r.t.* L, if $(\mathrm{pfx}\, K)\Sigma_{\mathrm{uc}} \cap (\mathrm{pfx}\, L) \subseteq \mathrm{pfx}\, K$.

An *automaton* is a tuple $A = (Q, \Sigma, \delta, q_{\mathrm{o}}, Q_{\mathrm{m}})$, with *state set* Q, *initial state* $q_{\mathrm{o}} \in Q$, *final states* $Q_{\mathrm{m}} \subseteq Q$, and the *transition relation* $\delta \subseteq Q \times \Sigma \times Q$. Throughout this paper we consider the transition relation to be deterministic, i.e, (q, σ, p'), $(q, \sigma, p'') \in \delta$ implies $p' = p''$. Thus, whenever convenient, we associate the transition relation with the partial *transition function* $\delta : Q \times \Sigma \to Q$ and write $\delta(p, \sigma) = p$ for $(q, \sigma, p) \in \delta$. The automaton is *full* if for all $q \in Q$ and $\sigma \in \Sigma$ there exists $p \in Q$ with $(q, \sigma, p) \in \delta$. We also refer to the common extension to the domain $Q \times \Sigma^*$; i.e., for $q \in Q$, we have $\delta(q, \epsilon) = q$ and, for $s \in \Sigma^*$ and $\sigma \in \Sigma$, we define $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma))$ if and only if $\delta(q, s)$ and $\delta(\delta(q, s), \sigma))$ are defined; else $\delta(q, s\sigma)$ is undefined. We write $\delta(q, s)!$ to indicate that δ is defined for the specified arguments $q \in Q$ and $s \in \Sigma^*$. A state $q \in Q$ is *reachable* (or *coreachable*) if there exists $s \in \Sigma^*$ such that $q = \delta(q_{\mathrm{o}}, s)$ (or $\delta(q, s) \in Q_{\mathrm{m}}$, resp.). With the automaton $A = (Q, \Sigma, \delta, q_{\mathrm{o}}, Q_{\mathrm{m}})$, we denote the *generated language* $\mathrm{L}(A) := \{\, s \in \Sigma^* \mid \delta(q_{\mathrm{o}}, s)!\,\}$ and the *accepted language* $\mathrm{L}_{\mathrm{m}}(A) := \{\, s \in \Sigma^* \mid \delta(q_{\mathrm{o}}, s) \in Q_{\mathrm{m}}\,\}$. A language $L \subseteq \Sigma^*$ is said to be *regular*, if it is accepted by some automaton with finitely many states. Regularity of a language is equivalent with the Nerode equivalence to induce only a finite number of equivalence classes. Given two automata A and B over the same alphabet Σ, we denote the product $A \times B$; see (Cassandras and Lafortune, 2008) for a formal definition. Then $A \times B$ accepts the intersection $\mathrm{L}_{\mathrm{m}}(A) \cap \mathrm{L}_{\mathrm{m}}(B)$. Specifically, if A and B are full automata, we can find alternative final states for $A \times B$ to accept $\mathrm{L}_{\mathrm{m}}(A)$, $\mathrm{L}_{\mathrm{m}}(B)$ and prefixes thereof.

A *Petri net* is a tuple $\mathcal{G} = (P, T, R, m_{\mathrm{o}}, F)$, where P is a finite set of *places*, T is a finite set of *transitions*, $R \subseteq (P \times T) \cup (T \times P)$ is the *incidence relation*, m_{o} is the *initial marking*, and F is a finite set of *final markings*. A *marking* $m : P \to \mathbb{N}_0$ is an allocation of finitely many tokens to each place $p \in P$. The set of all markings is denoted $\mathcal{M}$. Specifically, we have $m_{\mathrm{o}} \in \mathcal{M}$ and $F \subseteq \mathcal{M}$. Given a marking $m \in \mathcal{M}$, a transition $t \in T$ *can fire* or *is enabled* if all predecessor places in R have at least one token. This is denoted $m \xrightarrow{t}$. When a transition $m \xrightarrow{t}$ fires, it takes one token from each predecessor place and releases one token to each successor place. This is denoted $m \xrightarrow{t} m'$, with m' the marking after the transition has fired. Note that firings a transition may increase or decrease the overall number of tokens. If there is a maximum number of tokens over all markings that can be attained by firing finitely many transitions the net is called *bounded*. Else, it is called *unbounded*.

There are a variety of formal languages associated with a given Petri net $\mathcal{G}$. For the present paper, we refer to the *transition relation* $\mathcal{R} := \{\, (m, t, m') \in \mathcal{M} \times T \times \mathcal{M} \mid m \xrightarrow{t} m' \,\}$ and thereby effectively obtain a deterministic automaton representation over the infinite state set $\mathcal{M}$. Specifically, we reinterpret $\mathcal{R}$ as a partial transition function $\mathcal{R} : \mathcal{M} \times T \to \mathcal{M}$ with the same common extensions as in the context of deterministic automata. The *generated language* and the *accepted language* are then defined $\mathrm{L}(\mathcal{G}) := \{\, s \in T^* \mid \mathcal{R}(m_{\mathrm{o}}, s)!\,\}$ and $\mathrm{L}_{\mathrm{m}}(\mathcal{G}) := \{\, s \in T^* \mid \mathcal{R}(m_{\mathrm{o}}, s) \in F \neq \emptyset\,\}$ respectively, i.e., the set of feasible firing sequences and the subset of the latter that attains a final marking.

Remark 1. Referring to the notation in (Giua, 2013), our setting corresponds to the identity labelling $\ell(t) = t$ for all $t \in T =: \Sigma$, input and output functions I and O with range $\{0, 1\}$, and L-type acceptance condition. However, for our discussion in Sections 4 and 5, the representation of $\mathrm{L}(\mathcal{G})$ and $\mathrm{L}_{\mathrm{m}}(\mathcal{G})$ by the deterministic transition relation $\mathcal{R}$ are the only vital aspect. Specifically, our results readily extend to ϵ-free deterministic labelled nets with general input and output functions and with either L-type or G-type acceptance condition. □

2. ABSTRACTION BASED SUPERVISORY CONTROL

Following Ramadge and Wonham (1987, 1989), we consider the following closed-loop configuration for the control of discrete-event systems.

Definition 2. Given an alphabet Σ partitioned in controllable and uncontrollable events, $\Sigma = \Sigma_c \dot\cup \Sigma_{uc}$, consider a plant with *accepted behaviour* $L \subseteq \Sigma^*$ and with *local behaviour* $\mathrm{pfx}\, L$. With $\Gamma := \{\gamma \subseteq \Sigma \,|\, \Sigma_{uc} \subseteq \gamma\}$ the set of *control patterns*, a *supervisor* is a map $f : \mathrm{pfx}\, L \to \Gamma$ that maps event sequences $s \in \mathrm{pfx}\, L$ to a control pattern $f(s) \in \Gamma$. The *local closed-loop behaviour* and the *accepted closed-loop behaviour* are then defined by

$$L_f := \{ s \in \mathrm{pfx}\, L \,|\, \forall\, t \in \Sigma^*,\, \sigma \in \Sigma\,.\, t\sigma \in \mathrm{pfx}\, s \to \sigma \in f(t) \}, \quad (1)$$

and

$$K_f := L_f \cap L, \quad (2)$$

respectively. The supervisor is *non-blocking* if L_f and L are non-conflicting; i.e., if for all $s \in L_f$ there exists $t \in \Sigma^*$ such that $st \in K_f$. □

When compared with the original literature, our formal setting is slightly different in that we avoid the explicit reference to an automaton representation. However, this difference is purely cosmetic and the characterisation of achievable closed-loop behaviours under non-blocking supervisory control immediately carries over to our language-based setting.

Lemma 3. Consider a plant $L \subseteq \Sigma^*$ and a non-empty closed-loop candidate $K \subseteq \Sigma^*$, $\emptyset \neq K \subseteq L$. Then there exists a non-blocking supervisor $f : \mathrm{pfx}\, L \to \Gamma$ with accepted closed-loop behaviour K_f such that $K_f = K$ if and only if (a) K is relatively closed w.r.t. L and (b) K is controllable w.r.t. L.

Proof. See Theorem 6.1 in (Ramadge and Wonham, 1987) □

Given a closed-loop candidate $K \neq \emptyset$ that is both, relatively closed and controllable, the cited literature proves the existence of a suitable supervisor f by construction. Hence, the problem of supervisory controller synthesis is effectively equivalent to the synthesis of a feasible closed-loop behaviour that satisfies any additional control objectives. For the purpose of this paper, we address upper-bound language-inclusion specifications.

Definition 4. A *supervisory control problem with safety specification* (or short *control problem*) over Σ is a pair (L, E) with the plant $L \subseteq \Sigma^*$ and the specification $E \subseteq \Sigma^*$. Let

$$\mathcal{CF}(L, E) := \{ K \subseteq E \,|\, K \text{ is relatively closed w.r.t. } L \text{ and controllable w.r.t. } L \}. \quad (3)$$

Non-empty languages $K \in \mathcal{CF}(L, E)$ are referred to as *solutions* to (L, E). □

We recall that the two properties relative closedness and controllability are retained under arbitrary union; see Proposition 7.1 in (Ramadge and Wonham, 1987). Hence, the supremum

$$K^\uparrow := \sup \mathcal{CF}(L, E) := \cup\{ K \,|\, K \in \mathcal{CF}(L, E) \} \quad (4)$$

itself is relatively closed and controllable. Consequently, the control problem (L, E) has a solution if and only if $K^\uparrow$ is non-empty. To this end, Wonham and Ramadge (1987) consider regular parameters L and E and provide algorithms for the computation of finite automata realisations of $K^\uparrow$ and f.

In the case the plant L fails to be regular, one may resort to a regular abstraction L', i.e., an alternative less detailed model for the plant which is regular. Likewise, we may substitute E by E' for a more favourable representation. There are two core questions with this approach.

[Q1] If we find a solution for (L', E'), does this solution carry over to (L, E)?

[Q2] If (L', E') has no solution, how can we refine the plant abstraction L' or shape the substitute E' of the specification in order to increase our chances for success?

We begin with [Q1]. Since the supervisor restricts the behaviour of the plant and since our control objective is an upper bound, $L \subseteq L'$ and $E' \subseteq E$ are a natural prerequisite for an affirmative answer. Indeed, we have the following proposition.

Proposition 5. Given two control problems (L, E) and (L', E') over Σ with $L \subseteq L'$ and $E' \subseteq E$, consider any $K' \in \mathcal{CF}(L', E')$. Then K' is controllable w.r.t. L. If additionally $K' \subseteq L$, then K' is relatively closed w.r.t. L. In particular, under the assumption $E' \subseteq L$, we have that $\mathcal{CF}(L', E') \subseteq \mathcal{CF}(L, E)$.

Proof. Let $K' \in \mathcal{CF}(L', E')$. Then K' is controllable w.r.t. L', i.e., $((\mathrm{pfx}\, K')\Sigma_{uc}) \cap (\mathrm{pfx}\, L') \subseteq \mathrm{pfx}\, K'$. By monotonicity of the prefix operator we conclude $((\mathrm{pfx}\, K')\Sigma_{uc}) \cap (\mathrm{pfx}\, L) \subseteq ((\mathrm{pfx}\, K')\Sigma_{uc}) \cap (\mathrm{pfx}\, L') \subseteq \mathrm{pfx}\, K'$, hence K' is controllable w.r.t. L. Moreover, K' is relatively closed w.r.t. L', i.e., $(\mathrm{pfx}\, K') \cap L' = K'$. Thus, $(\mathrm{pfx}\, K') \cap L \subseteq (\mathrm{pfx}\, K') \cap L' = K'$. Referring to the prerequisite $K' \subseteq L$, we obtain the converse inclusion by extensiveness of the prefix operator, i.e., $K' = K' \cap L \subseteq (\mathrm{pfx}\, K') \cap L$. Thus, we have $K' = (\mathrm{pfx}\, K') \cap L$ and, hence, K' is relatively closed w.r.t. L. Since $E' \subseteq L$ implies $K' \subseteq L$ for all $K' \in \mathcal{CF}(L', E')$, we conclude $\mathcal{CF}(L', E') \subseteq \mathcal{CF}(L, E)$. □

Under the hypothesis of the above proposition, any solution to the substitute problem (L', E') also solves the original problem (L, E), and this answers [Q1]. Note that the additional assumption $E' \subseteq L$ is not restrictive since we have $\mathcal{CF}(L, E) = \mathcal{CF}(L, E \cap L)$ and may without loss of generality assume that $E \subseteq L$. Starting with the original control problem (L, E), we hence propose to find a regular superlanguage L' of L and a regular sublanguage E' of E. Our substitute control problem (L', E') can then be addressed by well understood methods.

Regardless the original problem, $L' = \Sigma^*$ and $E' = \emptyset$ trivially qualify as substitutes, but $(\Sigma^*, \emptyset)$ has no solution. Intuitively, our chances depend on how closely the substitute parameters relate to the original parameters. In practice, we begin with an initial regular-substitute problem and run a trial synthesis. If this fails, we refine our substitute parameters and again test for solutions. The two steps are then alternated until either a solution is found, and, hence, by Proposition 5 we have solved the original problem, or computational resources are exhausted.

The availability of effective procedures to obtain and/or to refine the regular-substitute parameters naturally depends on the actual representation of the original non-regular parameters; see e.g. Eisman and Ravikumar (2005) and Cordy and Salomaa (2007) for a general discussion on the approximation of non-regular languages from a computability perspective. For our specific use case of supervisory control we provide more detail in the following sections, and, doing so, we address question [Q2] and propose refinement strategies [H1]–[H3].

3. STRATEGIC ABSTRACTION REFINEMENT

Becoming more specific on how regular abstractions can be obtained, we consider the countably-infinite union composition

$$L := \cup\{ L_i \,|\, i \in \mathbb{N} \}, \quad (5)$$

where each component $L_i \subseteq \Sigma^*$, $i \in \mathbb{N}$, is regular. Note that any possibly non-regular $*$-language can be represented in this form. Clearly, with any finite selection $I \subset \mathbb{N}$ the finite union

$$N_I := \cup\{ L_i \,|\, i \in I \}, \quad (6)$$

is a regular sublanguage of L. Moreover, if we enlarge the selection by $J \subset \mathbb{N}$, $I \subseteq J$, we have

$$N_{\mathrm{I}} \subseteq N_{\mathrm{J}} \subseteq L\,, \tag{7}$$

i.e., the lower bound will not degenerate and we may optimistically expect it to improve.

Turning to regular superlanguages of L, a nearby choice is

$$L' := \{\, s \in \Sigma^* \mid \exists\, t \in L :\ t\,[\equiv_N]\, s \,\} \tag{8}$$

for any regular language $N \subseteq \Sigma^*$. Clearly, we have $L \subseteq L'$ and it is readily shown that the equivalence $[\equiv_N]$ is at least as fine as $[\equiv_{L'}]$, hence L' is regular. However, for the subsequent discussion it turns out useful to explicitly base the construction of a superlanguage L'_{I} on a given sublanguage $N_{\mathrm{I}} \subseteq L$, $I \subset \mathbb{N}$. Therefore, consider all sequences that exit $\operatorname{pfx} N_{\mathrm{I}}$ via $\operatorname{pfx} L$, i.e.,

$$M_{\mathrm{I}} := \{\, s\sigma \mid s \in \operatorname{pfx} N_{\mathrm{I}}\,,\ \sigma \in \Sigma,\ s\sigma \notin \operatorname{pfx} N_{\mathrm{I}}\,,\ s\sigma \in \operatorname{pfx} L \,\}\,. \tag{9}$$

If we see N_{I} as a model of L, the sequences $s\sigma \in M_{\mathrm{I}}$ are those, that exit the scope of N_{I} while still compliant with L. We hence expect

$$L'_{\mathrm{I}} := N_{\mathrm{I}} \ \cup\ (M_{\mathrm{I}}\Sigma^*)\,. \tag{10}$$

to be a superlanguage of L, with N_{I} the "good part" where our abstraction is accurate and $M_{\mathrm{I}}\Sigma^*$ the "bad part", where the abstraction degenerates. Indeed, we have the following proposition.

Proposition 6. Given a plant $L = \cup\{\, L_i \mid i \in \mathbb{N} \,\}$ with $\emptyset \neq L_i \subseteq \Sigma^*$ for all $i \in \mathbb{N}$, a finite selection $I \subset \mathbb{N}$, $I \neq \emptyset$, and N_{I} and M_{I} as defined by Eqs. (6&9). Then L'_{I} defined by Eq. (10) is a superlanguage of L. Applying the same construction for an enlarged selection $J \subset \mathbb{N}$, $I \subseteq J$, we have $L \subseteq L'_{\mathrm{J}} \subseteq L'_{\mathrm{I}}$. The claims remain valid when substituting $\operatorname{pfx} L$ in Eq. (9) by $\operatorname{pfx} V$ for any superlanguage V, $L \subseteq V \subseteq \Sigma^*$.

Proof. To see that $L \subseteq L'_{\mathrm{I}}$ pick any $t \in L$. We distinguish two cases. If $t \in N_{\mathrm{I}}$, we immediately obtain $t \in N_{\mathrm{I}} \subseteq L'_{\mathrm{I}}$. For the remaining case $t \notin N_{\mathrm{I}}$, we observe that by hypothesis $N_{\mathrm{I}} \neq \emptyset$ and there hence uniquely exists the longest strict prefix $s < t$ with $s \in \operatorname{pfx} N_{\mathrm{I}}$. Thus, we can decompose t by $t = s\sigma r$, $r \in \Sigma^*$, $\sigma \in \Sigma$, such that $s \in \operatorname{pfx} N_{\mathrm{I}}$ and $s\sigma \notin \operatorname{pfx} N_{\mathrm{I}}$. Since $t \in L$ we also have that $s\sigma \in \operatorname{pfx} L \subseteq \operatorname{pfx} V$. We therefore obtain $s\sigma \in M_{\mathrm{I}}$ and, hence $t \in M_{\mathrm{I}}\Sigma^* \subseteq L'_{\mathrm{I}}$. This concludes the second case and establishes $L \subseteq L'_{\mathrm{I}}$. Regarding monotonicity w.r.t. the selection, pick an arbitrary $t \in L'_{\mathrm{J}}$. Here, we distinguish three cases. If $t \in N_{\mathrm{I}}$, we immediately obtain $t \in N_{\mathrm{I}} \subseteq L'_{\mathrm{I}}$. If $t \notin N_{\mathrm{I}}$ but $t \in N_{\mathrm{J}}$, we decompose t as above by $t = s\sigma r$, t, $r \in \Sigma^*$, $\sigma \in \Sigma$, with $s < t$ the longest strict prefix in $\operatorname{pfx} N_{\mathrm{I}}$. Since $t \in N_{\mathrm{J}} \subseteq L$, this implies $s\sigma \in \operatorname{pfx} L \subseteq \operatorname{pfx} V$, and hence $t \in M_{\mathrm{I}}\Sigma^* \subseteq L'_{\mathrm{I}}$. For the remaining third case, we have $t \in M_{\mathrm{J}}\Sigma^*$ but $t \notin N_{\mathrm{J}}$. Here, we consider two decompositions. First, choose $s \in \operatorname{pfx} N_{\mathrm{J}}$ and $\sigma \in \Sigma$ such that $s\sigma \leq t$ with $s\sigma \in \operatorname{pfx} V$ and $s\sigma \notin \operatorname{pfx} N_{\mathrm{J}}$. By $N_{\mathrm{I}} \subseteq N_{\mathrm{J}}$ we can further decompose s by with $r \in \Sigma^*$ and $\rho \in \Sigma$ with $r\rho \leq s\sigma$ such that $r \in \operatorname{pfx} N_{\mathrm{I}}$ and $r\rho \notin \operatorname{pfx} N_{\mathrm{I}}$. Observe by $r\rho \leq s\sigma \in \operatorname{pfx} V$ that $r\rho \in \operatorname{pfx} V$. Hence, $r\rho \in M_{\mathrm{I}}$ and this implies $t \in M_{\mathrm{I}}\Sigma^* \subseteq L'_{\mathrm{I}}$. The three cases being exhaustive, and since $t \in L'_{\mathrm{J}}$ was chosen arbitrarily, we conclude that $L'_{\mathrm{J}} \subseteq L'_{\mathrm{I}}$. □

The test for $s\sigma \in \operatorname{pfx} L$ in Eq. (9), however, in general can not be implemented by a finite automaton and, hence, we need to impose some additional requirements on the sublanguages L_i, $i \in \mathbb{N}$, in order to ensure regularity of M_{I}. Since N_{I} is regular, and since we are testing for $s \in \operatorname{pfx} L$ and $s\sigma \notin \operatorname{pfx} L$ anyway, a nearby requirement for this purpose is

$$\forall\, s',\, s'' \in \operatorname{pfx} N_{\mathrm{I}} :\ s'\,[\equiv_{N_I}]\, s'' \ \rightarrow\ s'\,[\equiv_L]\, s''\,, \tag{11}$$

i.e., restricted to the domain $\operatorname{pfx} N_{\mathrm{I}}$ the equivalence $[\equiv_{N_I}]$ is at least as fine as $[\equiv_L]$, while on the entire domain Σ^* the equivalence $[\equiv_L]$ is expected to be finer than $[\equiv_{N_I}]$.

An alternative way to ensure regularity on M_{I} is to relax the test for $s\sigma \in \operatorname{pfx} L$ in Eq. (9), i.e., to allow for false positives and thereby obtain a regular representation. Here, we propose the following variation in the definition of M_{I}:

$$M_{\mathrm{I}} = \{\, s\sigma \mid s \in \operatorname{pfx} N_{\mathrm{I}}\,,\ \sigma \in \Sigma,\ s\sigma \notin \operatorname{pfx} N_{\mathrm{I}}\,, \quad \exists\, t :\ t\,[\equiv_{N_I}]\, s\sigma \text{ and } t \in \operatorname{pfx} L \,\}\,. \tag{12}$$

Technically, this relaxation amounts the substitution of L by a superlanguage V, $L \subseteq V$, and is, hence, in the scope of Proposition 6.

We are now in the position to discuss question [Q2]. Here, we assume that both, the plant and the specification, are given as a countable-infinite union composition of regular components, i.e., $L = \cup\{\, L_i \mid i \in \mathbb{N} \,\}$ as above and likewise $E = \cup\{\, E_i \mid i \in \mathbb{N} \,\}$, with all L_i, $E_i \subseteq \Sigma^*$ regular. For the regular superlanguage of L we use L'_{I} defined in Eq. (10), where $I \subset \mathbb{N}$ is a finite selection. For the regular sublanguage of E we use E'_{I} in analogy to Eq. (6), i.e., $E'_{\mathrm{I}} := \cup\{\, E_i \mid i \in I \,\}$. In the case that we have no practical test to check for $E \subseteq L$, we additionally intersect with N_{I} to ensure that $E'_{\mathrm{I}} \subseteq L$ as required by Proposition 5. Note that using the same selection I for plant and specification is not restrictive since the components can be formally rearranged.

We say that the selection I is *successful* if the control problem $(L'_{\mathrm{I}},\, E'_{\mathrm{I}})$ exhibits a solution. By Proposition 5, any solution carries over to the original problem $(L,\, E)$ and, having a successful selection I we are done. If I is not successful, we propose to enlarge I by $J \subset \mathbb{N}$, $I \subset J$. Clearly, iterating the two steps of trial synthesis and refinement, we can only become eventually successful if a successful selection exists at all. This natural prerequisite is not only necessary, but also sufficient.

Proposition 7. Given a control problem $(L,\, E)$ over Σ with representations $L = \cup\{\, L_i \mid i \in \mathbb{N} \,\}$ and $E = \cup\{\, E_i \mid i \in \mathbb{N} \,\}$ consider a refinement scheme $(I_k)_{k\in\mathbb{N}}$, $I_k \subset I_{k+1} \subset \mathbb{N}$, such that $\cup\{\, I_k \mid k \in \mathbb{N} \,\} = \mathbb{N}$. If there exists a finite successful selection $J \subset \mathbb{N}$ then there also exists $k \in \mathbb{N}$ such that I_k is successful.

Proof. Since J is finite, there exists $k \in \mathbb{N}$ such that $J \subseteq I_k =: I$. We immediately have $E'_{\mathrm{J}} \subseteq E'_{\mathrm{I}}$ and refer to Proposition 6 for $L'_{\mathrm{I}} \subseteq L'_{\mathrm{J}}$. By hypothesis, $(L'_{\mathrm{J}},\, E'_{\mathrm{J}})$ exhibits a solution, and, by Proposition 5 this solution also solves $(L'_{\mathrm{I}},\, E'_{\mathrm{I}})$. Hence, $I_k = I$ is a successful selection. □

A refinement scheme $(I_k)_{k\in\mathbb{N}}$ which qualifies for the above proposition is $I_k := \{\, i \in \mathbb{N} \mid i \leq k \,\}$, i.e., including components one by one in a prescribed order.

Example 8. Consider a processing machine which can take workpieces and release processed workpieces, one at a time, represented by the events a and b, respectively. Thus, $\Sigma = \{\, a,\, b \,\}$ is our alphabet. Taking a conceptual perspective, we do not want to become specific about the internal capacity of the machine at this stage and, hence, model the machine by the language

$$L := \{\, s \in \Sigma^* \mid \#_a(s) = \#_b(s),\ \forall\, t \leq s :\ \#_b(t) \leq \#_a(t) \,\}\,, \tag{13}$$

where $\#\_(s)$ denotes the count of the specified event in s. In particular, L is non-regular; for an infinite automaton realisation see Figure 1. Limiting the capacity to any finite number $i \in \mathbb{N}$ we obtain $L_i := \{\, s \in L \mid \forall\, t \leq s :\ \#_a(t) - \#_b(t) \leq i \,\}$. In particular, L_i is regular; see Figure 1 for a realisation of L_2. Since any

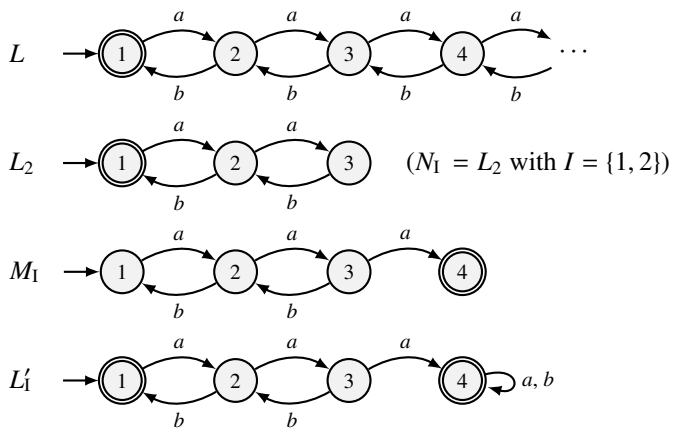

Fig. 1. Automata realisations for Example 8.

$s \in L$ has finitely many prefixes $t \leq s$, we can take $i \in \mathbb{N}$ as the maximum of $\#_a(t) - \#_b(t)$ to observe $s \in L_i$, and, hence $L = \cup\{ L_i \mid i \in \mathbb{N} \}$. To obtain an abstraction L'_{I} as proposed in this section, consider the selection $I = \{1, 2\}$. Since for our specific case we have that $L_i \subseteq L_j$ for any $i \leq j$, we observe $N_{\mathrm{I}} = L_2$. Realisations of M_{I} and L'_{I} again are shown in Figure 1. Intuitively, any specification that one could enforce for the actual plant which only utilises a capacity of no more than 2 workpieces can also be enforced for L'_{I}, and, hence, can be synthesised based on finite automata representations. Likewise, for any regular specification that utilises some finite capacity, there exists a successful selection $J \subset \mathbb{N}$ for which (L'_{J}, E) exhibits a solution. Although this seems obvious for our simple example, it will become less so for more involved scenarios. □

In the above example, increasing the capacity of the processing machine apparently is the only sensible strategy for abstraction refinement. For more complex situations, we present a number of heuristics on how specifically to extend a not successful selection $I \subset \mathbb{N}$.

[H1] Consider $j \notin I$ such that $(\mathrm{pfx}\, L_j) \cap M_{\mathrm{I}}$ is non-empty. If we extend our selection by $J = I \cup \{j\}$ for such a component, this will replace the degenerated sequences $s\sigma\Sigma^* \subseteq L'_{\mathrm{I}}$ for all $s\sigma \in (\mathrm{pfx}\, L_j) \cap M_{\mathrm{I}}$ by $s\sigma\Sigma^* \cap \mathrm{pfx}\, L_j$ in the refined abstraction $L'_{\mathrm{J}} \subseteq L'_{\mathrm{I}}$. Such a refinement will push the boundary between the "good part" and the "bad part" of the abstraction by a minimal amount and is, hence, excepted to be of moderate computational cost.

[H2] Consider $j \notin I$ with $(\mathrm{pfx}\, L_j) \cap M_{\mathrm{I}} \neq \emptyset$ as in [H1] and, additionally, $(\mathrm{pfx}\, L_j) \cap M_{\mathrm{I}} \cap (\mathrm{pfx}\, E) = \emptyset$. Then, the future evolution of the plant after any sequence $s\sigma \in (\mathrm{pfx}\, L_j) \cap M_{\mathrm{I}}$ violates the specification and, hence, a supervisor needs to prevent $s\sigma$ in the first place. Therefore we do not expect any benefits from a refinement regarding the future of such sequences $s\sigma$, and should avoid such additional components L_j for a refinement.

By the above heuristic [H2], the future of sequences $s\sigma \in M_{\mathrm{I}}$ but $s\sigma \notin \mathrm{pfx}\, E$ shall not be addressed in a refined abstraction, because they are "doomed" anyway. Likewise, we do not need to address the future sequences, which we can already control with a given abstraction L'_{I}. Such sequences are referred to as *winning configurations* and define the so called *controllability prefix*. The latter concept stems from the control of ω-languages (Thistle and Wonham, 1994) and has been adapted to $*$-languages by Moor and Schmidt (2017); Moor et al. (2020) as follows:

Definition 9. Given a plant $L \subseteq \Sigma^*$, a specification E, $\emptyset \neq E \subseteq L$, and a string $s \in \Sigma^*$, we use the convenient notation

$$L_s := L \cap (s\Sigma^*), \quad E_s := E \cap (s\Sigma^*). \tag{14}$$

The *controllability prefix* $\mathrm{cfx}\,(L, E) \subseteq \Sigma^*$ is then defined as the set of all strings $s \in \Sigma^*$ such that $\sup \mathcal{CF}(L_s, E_s) \neq \emptyset$. □

Specifically, for $s \in \mathrm{cfx}\,(L'_{\mathrm{I}}, E'_{\mathrm{I}})$ the future behaviour after the occurrence of s does not need to be addressed in a refinement of the abstraction. This leads to the following additional heuristic.

[H3] Consider $j \notin I$ with $(\mathrm{pfx}\, L_j) \cap M_{\mathrm{I}} \neq \emptyset$ as in [H1] and, additionally, assume that for all $s\sigma \in (\mathrm{pfx}\, L_j) \cap M_{\mathrm{I}}$ we have $s \in \mathrm{cfx}\,(L'_{\mathrm{I}}, E'_{\mathrm{I}})$. Then s is a winning configuration in the control problem $(L'_{\mathrm{I}}, E'_{\mathrm{I}})$ and we do not expect any benefits from using L_j for the refinement of the abstraction.

4. ABSTRACTIONS FOR UNBOUNDED PETRI NETS

We further elaborate our results in the context of unbounded Petri nets where the state set of the corresponding reachability graph becomes infinite. Similar to the idea showcased in Proposition 6, we aim to design a regular abstraction of the accepted language $\mathrm{L_m}(\mathcal{G})$ of a Petri net $\mathcal{G}$ with an accurate and a degenerate part. As such we start with the trivial bounds $\emptyset \subseteq \mathrm{L_m}(\mathcal{G}) \subseteq \Sigma^*$ and iteratively construct refinements thereof. To this end, consider Procedure 1 stated in the notation given in Section 1; specifically, $\mathcal{R} \subseteq \mathcal{M} \times T \times \mathcal{M}$ denotes the transition relation associated with the Petri net $\mathcal{G}$.

Procedure 1 Output A_i, B_i in every iteration

Require: Petri net $\mathcal{G} = (P, T, R, m_{\mathrm{o}}, F)$
1: $i \leftarrow 1$
2: $Q_1 := P_1 := \{m_{\mathrm{o}}\}$, $\delta_1 := \emptyset$
3: **loop**
4: **print** $A_i = (Q_i, T, m_{\mathrm{o}}, \delta_i, (Q_i \cap F) \setminus P_i)$
5: $O_i := \{ m \in Q_i \mid m \text{ is not coreachable in } A_i \}$
6: $\rho_i := \{ (m, t, m) \mid m \in O_i, t \in T, \forall m' : (m, t, m') \notin \delta_i \}$
7: **print** $B_i = (Q_i, T, m_{\mathrm{o}}, \delta_i \cup \rho_i, (Q_i \cap F) \cup O_i))$
8: $P_{i+1} := \{ m' \in \mathcal{M} \setminus Q_i \mid \exists m \in P_i, t \in T : (m, t, m') \in \mathcal{R} \}$
9: $Q_{i+1} := Q_i \dot\cup P_{i+1}$
10: $\delta_{i+1} := \delta_i \dot\cup \{ (m, t, m') \in \mathcal{R} \mid m \in P_i \}$
11: $i \leftarrow i + 1$
12: **end loop**

Commenting the above procedure, we first focus attention on the finite sets Q_i, $P_i \subseteq \mathcal{M}$, $i \in \mathbb{N}$; i.e., their initialisation, line 2, and their propagation, lines 8 and 9. It this regard, the procedure matches the common iteration that computes the markings attainable form the initial marking m_{o}; i.e., Q_i are the markings that can be attained by no more than i transitions and we have $Q_i \subseteq Q_{i+1}$ for all $i \in \mathbb{N}$. In our formulation, $P_i \subseteq Q_i$ are the markings newly discovered in the i-th iteration, i.e., the set of those markings, that can be attained by i transitions but not by less. Turning to δ_i, line 10, we note that this is the finite subset of $\mathcal{R}$ of those transitions inspected so far to establish Q_i. In particular, δ_i is deterministic and we have $\delta_i \subseteq \delta_{i+1} \subseteq \mathcal{R}$ for all $i \in \mathbb{N}$. Moreover, as a partial functions, δ_i and $\mathcal{R}$ are identical on the restricted domain $(Q_i \setminus P_i) \times \Sigma$. Once per iteration the procedure outputs two automata A_i and B_i with state set Q_i, transition relation δ_i or the variation ρ_i thereof, and with

strategic final states such that $\mathrm{L_m}(A_i)$ and $\mathrm{L_m}(B_i)$ are regular sublanguages or superlanguages, respectively, of $\mathrm{L_m}(\mathcal{G})$.

Remark 10. Inspecting Procedure 1, it exclusively relies on a possibly infinite automaton to realise $L = \mathrm{L_m}(\mathcal{G})$ and the effective evaluation of one-step successors $\mathcal{R}(m, t)$ for $m \in \mathcal{M}$ and $t \in T$. Specifically, $\{\, \mathcal{R}(m, t) \,|\, t \in T \,\}$ is required to be finite. Our approach should be applicable to any other class of models that satisfy likewise requirements. □

The following proposition justifies Procedure 1.

Proposition 11. Let $\mathcal{G}$ be the Petri net used as input for Procedure 1. We then have that

$$\mathrm{L_m}(A_i) \subseteq \mathrm{L_m}(A_{i+1}) \subseteq \mathrm{L_m}(\mathcal{G}) \subseteq \mathrm{L_m}(B_{i+1}) \subseteq \mathrm{L_m}(B_i) \qquad (15)$$

for all $i \in \mathbb{N}$. Moreover, $\mathrm{L_m}(\mathcal{G}) = \cup\{\, \mathrm{L_m}(A_i) \,|\, i \in \mathbb{N} \,\}$.

Proof. We first consider the languages $\mathrm{L_m}(A_i)$, $i \in \mathbb{N}$. Pick any $s \in \mathrm{L_m}(A_i)$, i.e.,

$$\delta_i(m_o,\, s) \in (\mathrm{Q}_i \cap F) \setminus P_i\,. \qquad (16)$$

By $\delta_i \subseteq \mathcal{R}$ this implies $\mathcal{R}(m_o,\, s) \in F$ and, hence, $s \in \mathrm{L_m}(\mathcal{G})$. We conclude $\mathrm{L_m}(A_i) \subseteq \mathrm{L_m}(\mathcal{G})$ for all $i \in \mathbb{N}$. Referring to $P_{i+1} \cap Q_i = \emptyset$ and $Q_i \subseteq Q_{i+1}$ we make the preliminary observation $(\mathrm{Q}_i \cap F) \setminus P_i \subseteq (\mathrm{Q}_i \cap F) \setminus P_{i+1} \subseteq (\mathrm{Q}_{i+1} \cap F) \setminus P_{i+1}$ Picking again any $s \in \mathrm{L_m}(A_i)$, Eq. (16) by $\delta_i \subseteq \delta_{i+1}$ and by our preliminary observation implies $\delta_{i+1}(m_o,\, s) \in (\mathrm{Q}_{i+1} \cap F) \setminus P_{i+1}$ and, hence, $s \in \mathrm{L_m}(A_{i+1})$. We conclude $\mathrm{L_m}(A_i) \subseteq \mathrm{L_m}(A_{i+1})$ for all $i \in \mathbb{N}$. So far, we have established the first two inclusions in Eq. (15). This obviously implies $\mathrm{L_m}(\mathcal{G}) \supseteq \cup\{\, \mathrm{L_m}(A_i) \,|\, i \in \mathbb{N} \,\}$. For the converse, pick any $s \in \mathrm{L_m}(\mathcal{G})$. With l the length of the string s, there exists a sequence of l transitions from m_o to a final marking $m_l \in F$. Clearly, each intermediate state can be reached by at most l transitions and is, hence, reachable in any A_i with $i \geq l$, i.e., $\delta_i(m_\mathrm{o},\, s) = m_l \in Q_i$ for $i \geq l$. Since we could have $m_l \in P_l$, and since P_l is removed from the final states of A_l, we instead consider A_{l+1} to observe $m_l = \delta_{l+1}(m_\mathrm{o},\, s) \in (\mathrm{Q}_{l+1} \cap F) \setminus P_{l+1}$. Hence, $s \in \mathrm{L_m}(A_{l+1})$, and this establishes $\mathrm{L_m}(\mathcal{G}) \subseteq \cup\{\, \mathrm{L_m}(A_i) \,|\, i \in \mathbb{N} \,\}$.

We now address the languages $\mathrm{L_m}(B_i)$, and we do so by referring to Proposition 6 and the construction via N_I and M_I in Eqs. (6&9), where we relax the test for $s\sigma \in \operatorname{pfx} L$ by $s\sigma \in \mathrm{L}(\mathcal{G}) =: V$ in the definition of M_I. Let $L_i := \mathrm{L_m}(A_i)$ for $i \in \mathbb{N}$ and consider the selection $I = \{1, 2, \ldots i\}$ for a specific $i \in \mathbb{N}$. By monotonicity we have $N_\mathrm{I} = L_i$. For a realisation of M_I, we need to test for sequences $s\sigma$, $s \in \Sigma^*$, $\sigma \in \Sigma$, with $s \in \operatorname{pfx} N_\mathrm{I}$, $s\sigma \notin \operatorname{pfx} N_\mathrm{I}$ and $s\sigma \in \mathrm{L}(\mathcal{G})$. Since N_I is realised by A_i, $s \in \operatorname{pfx} N_\mathrm{I}$ is equivalent to $q := \delta_i(m_\mathrm{o},\, s)$ being coreachable in A_i, i.e., $q \in Q_i \setminus O_i$. By the final states $(Q_i \cap F) \setminus P_i$ of A_i, line 4, and the absence of outgoing transitions from P_i, line 10, this implies $q \in Q_i \setminus P_i$. Therefore, $s\sigma \in \mathrm{L}(\mathcal{G})$ is characterised by $q' := \delta_i(m_\mathrm{o},\, s\sigma) \in Q_i$. Given that $\delta_i(m_\mathrm{o},\, s\sigma)$ is defined, $s\sigma \notin \operatorname{pfx} N_\mathrm{I}$ is equivalent to $q' \in O_i$. Thus, using O_i as final states, we realise M_I over the state set Q_i with the transition relation δ_i. Introducing self-loops for the extended transition relation ρ_i, line 6, then realises $M_\mathrm{I}\Sigma^*$. Finally, by expanding the final states by $Q_i \cap F$ for automaton B_i, line 7, we obtain $\mathrm{L_m}(B_i) = N_\mathrm{I} \cup M_\mathrm{I}\Sigma^* = L'_\mathrm{I}$. Then, the remaining two inclusions in Eq (15) are an immediate consequence of Proposition 6. □

Identifying $N_\mathrm{I} = L_i = \mathrm{L}(A_i)$, $I = \{1, 2, \ldots, i\}$, $i \in \mathbb{N}$, and referring to the discussion in Section 3, the proposed procedure adheres to Heuristic [H1] in that $A_{i+1} \neq A_i$ implies the existence of $s \in \Sigma^*$ and $\sigma \in \Sigma$ such that $s \in \mathrm{L}(A_i)$, $s\sigma \notin \mathrm{L}(A_i)$ and $s\sigma \in \mathrm{L}(A_{i+1}) \subseteq \mathrm{L}(\mathcal{G})$.

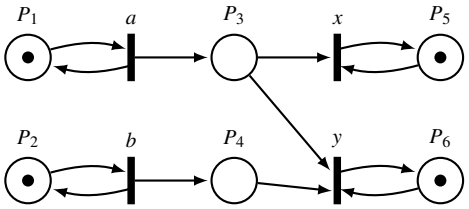

Fig. 2. Petri net $\mathcal{G}$ for Example 12.

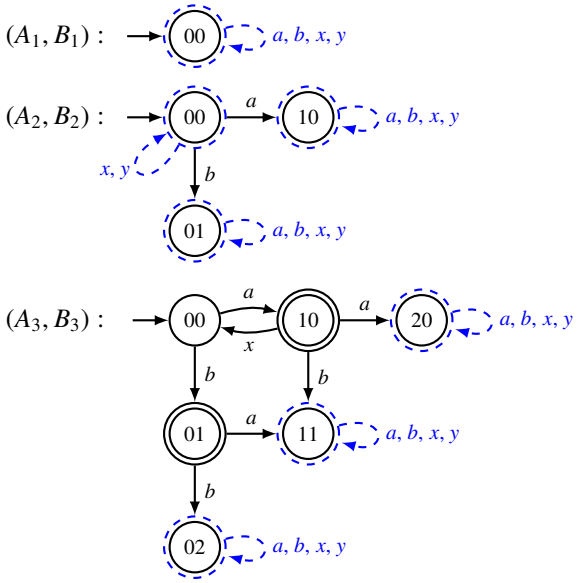

Fig. 3. Initial three outputs of Procedure 1 for Example 12.

Example 12. Consider a machine that can produce, store, and further process two types of workpieces. Production is represented by the events a and b and processing with x and y. Conceptually we want to leave the storage buffer unlimited and aim to derive a suitable capacity from subsequent controller synthesis. As a first step, however, we focus on the construction of regular abstractions for the plant. See Figure 2 for a Petri net realisation. In this context a token in P_3 or P_4 represents a workpiece of type 1 or 2 respectively. For illustration purposes only, the set of final markings is assumed $F = \{[111011]^\top, [110111]^\top\}$. The first three outputs of Procedure 1 are displayed in Figure 3. Transitions and final markings belonging solely to B_i are represented by dashed blue lines. Since only the number of tokens in P_3 and P_4 can change, the state labels in the figure refer to the token allocation of those two places. □

5. SYNTHESIS FOR UNBOUNDED PETRI NETS

At this point we are in the position to combine the results from Propositions 5 and 11 for the synthesis of a supervisor. Here, we specifically consider a plant represented by a possibly unbounded Petri net $\mathcal{G}$ and a regular specification $E \subseteq T^*$. The proposed alternation of abstraction refinement in the spirit of Procedure 1 with trial synthesis is given as Procedure 2.

Recalling that in Procedure 1 refinement amounts to an inspection of one-step successors of *all* states in P_i discovered in the most recent iteration, specific refinement strategies e.g. Heuristics [H2] and [H3] are implemented by suspending the inspection of one-step successors for certain states $S_i \subseteq P_i$; see lines 11 and 14. Note that a state $m \in S_i \subseteq P_i$ for which refinement is suspended in the i-th iteration may need to be

refined in a later iteration, and we therefore merge S_i to the newly discovered states P_{i+1}; see line 12.

Procedure 2 Output final $A_i, B_i, K^\uparrow$

Require: Petri net $\mathcal{G} = (P, T, R, m_o, F)$, $T = T_c \dot\cup T_{uc}$, $E \subseteq T^*$

1: $i \leftarrow 1$
2: $Q_1 := P_1 := \{m_o\}$, $\delta_1 := \emptyset$
3: **loop**
4: $A_i = (Q_i, T, m_o, \delta_i, (Q_i \cap F) \setminus P_i)$
5: $O_i := \{ m \in Q_i \,|\, m \text{ is not coreachable in } A_i \}$
6: $\rho_i := \{ (m, t, m) \,|\, m \in O_i, t \in T, \forall m' : (m, t, m') \notin \delta_i \}$
7: $B_i = (Q_i, T, m_o, \delta_i \cup \rho_i, (Q_i \cap F) \cup O_i))$
8: **if** $K^\uparrow := \sup \mathcal{CF}(\mathrm{L_m}(B_i), \mathrm{L_m}(A_i) \cap E) \neq \emptyset$ **then**
9: **return** $A_i, B_i, K^\uparrow$
10: **end if**
11: choose $S_i \subseteq P_i$ as refinement strategy, e.g. Eqs. (19&20)
12: $P_{i+1} := \{m' \in \mathcal{M} \setminus Q_i \,|\, \exists m \in P_i \setminus S_i,\ t \in T : (m, t, m') \in \mathcal{R}\} \cup S_i$
13: $Q_{i+1} := Q_i \dot\cup P_{i+1}$
14: $\delta_{i+1} := \delta_i \dot\cup \{ (m, t, m') \in \mathcal{R} \,|\, m \in P_i \setminus S_i \}$
15: $i \leftarrow i + 1$
16: **end loop**

Proposition 13. Let the Petri net $\mathcal{G}$ and the specification E be the inputs for Procedure 2. If the procedure terminates, we have that

$$\emptyset \neq K^\uparrow \in \mathcal{CF}(\mathrm{L_m}(\mathcal{G}), E)\,, \tag{17}$$

regardless any specific refinement strategy $S_i \subseteq P_i$, line 11.

Proof. Note that the addition of suspended states S_i in Procedure 2 when compared to Procedure 1 merely slows abstraction refinement in specific directions. Specifically, this does not affect the proof of Proposition 11 regarding the inclusions $\mathrm{L_m}(A_i) \subseteq \mathrm{L_m}(\mathcal{G}) \subseteq \mathrm{L_m}(B_i)$. Referring to line 9, invoking trial synthesis with the specification $\mathrm{L_m}(A_i) \cap E$, we have $K^\uparrow \subseteq \mathrm{L_m}(\mathcal{G})$. Thus, the prerequisites of Proposition 5 are satisfied and Eq. (17) is an immediate consequence of that proposition. □

Proposition 14. Let the Petri net $\mathcal{G}$ and the specification E be the inputs for Procedure 2. If there exists a regular solution $K \in \mathcal{CF}(\mathrm{L_m}(\mathcal{G}), E)$, $K \neq \emptyset$, such that the attainable markings

$$\mathcal{M}_{\mathrm{pfx}\,K} := \{ m \in \mathcal{M} \,|\, \exists\, s \in \mathrm{pfx}\,K \colon \mathcal{R}(m_o, s) = m \} \tag{18}$$

in closed-loop configuration are bounded, then the Procedure 2 without refinement strategy, i.e., $S_i = \emptyset$, terminates.

Proof. Denote H a finite automaton with state set X and final states X_K that accepts K and, without loss of generality, assume the H is full, i.e., $\mathrm{L_m}(H) = K$, $\mathrm{L}(H) = T^*$. Moreover denote $G = (\mathcal{M}, T, \mathcal{R}, m_o, F)$ the infinite automaton realisation associated with $\mathcal{G}$, i.e., $\mathrm{L_m}(G) = \mathrm{L_m}(\mathcal{G})$, $\mathrm{L}(G) = \mathrm{L}(\mathcal{G})$. Consider the product composition $H \times G$ over the state set $Z = X \times \mathcal{M}$ and denote the final states Z_K. Since $K \subseteq \mathrm{L_m}(\mathcal{G})$, $H \times G$ accepts K, and, referring to the alternative set of final states $Z_G := X \times F$, the otherwise identical automaton accepts $\mathrm{L_m}(\mathcal{G})$. Likewise, we obtain by $Z_{\mathrm{pfx}\,K} := \{ (x, m) \in Z) \,|\, x \text{ is reachable and coreachable} \}$ a set of final states to accept $\mathrm{pfx}\,K$. Since K is relatively closed w.r.t. $\mathrm{L_m}(G)$ we have $Z_{\mathrm{pfx}\,K} \cap Z_G = Z_K$, and, in particular, $Z_K \subseteq Z_G$. By hypothesis, $\mathcal{M}_{\mathrm{pfx}\,K}$ is bounded, and, hence there are only finitely many markings $m \in Q_{\mathrm{pfx}\,K} := \{ m \in \mathcal{M} \,|\, \exists\, x \colon (x, m) \in Z_{\mathrm{pfx}\,K} \}$. For each individual marking $m \in \mathcal{M}$, there is a shortest sequence $s_m \in T^*$ such that $\mathcal{R}(m_o, s) = m$ and we denote $l \in \mathbb{N}$ the length of the longest such sequence s_m over all $m \in Q_{\mathrm{pfx}\,K}$. Now consider automata A_i and B_i for some iteration $i \geq l + 1$. Referring to lines 12–14 of Procedure 2 with $S_i = \emptyset$, we note that $Q_{\mathrm{pfx}\,K} \subseteq Q_i / P_i$ and that δ_i restricted to the domain $Q_{\mathrm{pfx}\,K} \times T$ as a partial function is identical to $\mathcal{R}$ restricted to the same domain. This implies that $K \subseteq \mathrm{L_m}(A_i)$ and that $H \times B_i$ accepts K. As above, we may consider the alternative set of final states $Z_B := X \times Q_B$, $Q_B := (Q_i \cap F) \cup O_i$, such that the otherwise identical automaton accepts $\mathrm{L_m}(B_i)$. Since K is controllable w.r.t. $\mathrm{L_m}(G)$, any transition (z, σ, z') of $H \times B_i$ with $z \in Z_{\mathrm{pfx}\,K}$, $z' \notin Z_{\mathrm{pfx}\,K}$ must be controllable, i.e., $\sigma \in T_c$. This implies controllability of K w.r.t. $\mathrm{L_m}(B_i)$. To observe relative closedness of K w.r.t. $\mathrm{L_m}(B_i)$, we note that $Z_{\mathrm{pfx}\,K}$ is coreachable in $H \times B_i$, and, hence, $Q_{\mathrm{pfx}\,K}$ are coreachable in B_i. For the final states of Q_B of B_i this implies $q \in F$ for all x such that $(x, q) \in Z_{\mathrm{pfx}\,K} \cap Z_B$. Therefore, we conclude $Z_{\mathrm{pfx}\,K} \cap Z_B \subseteq Z_{\mathrm{pfx}\,K} \cap Z_G = Z_K$ and, hence, $(\mathrm{pfx}\,K) \cap \mathrm{L_m}(B_i) \subseteq (\mathrm{pfx}\,K) \cap \mathrm{L_m}(\mathcal{G})$. By $\mathrm{L_m}(\mathcal{G}) \subseteq \mathrm{L_m}(B_i)$ we obtain equality. Since K is relatively closed w.r.t. $\mathrm{L_m}(\mathcal{G})$, we indeed have $(\mathrm{pfx}\,K) \cap \mathrm{L_m}(B_i) = K$. To this end, we have established $K \in \mathcal{CF}(\mathrm{L_m}(B_i), E)$. For $K \in \mathcal{CF}(\mathrm{L_m}(B_i), E \cap \mathrm{L_m}(A_i))$ recall $K \subseteq \mathrm{L_m}(A_i)$ from above. Since K is a solution of $(\mathrm{L_m}(\mathcal{G})), E)$, it is nonempty and since $\emptyset \neq K \subseteq \sup \mathcal{CF}(\mathrm{L_m}(B_i), E \cap \mathrm{L_m}(A_i))$, Procedure 2 terminates after the i-th iteration at the latest. □

Regarding Heuristic [H2], we want to suspend the refinement for states $m \in P_i$ that when attained in the current abstraction A_i are known to invalidate the upper-bound specification; i.e.,

$$S_i^{[\mathbf{H2}]} := \{ m \in P_i \,|\, \forall\, s \in T^* : \delta_i(m_o, s) = m \;\rightarrow\; s \notin \mathrm{pfx}\,(E \cap \mathrm{L_m}(A_i)) \}\,. \tag{19}$$

Algorithmically, states in $S_i^{[\mathbf{H2}]}$ can be identified by (1.) considering a full-automaton realisation H of E with state set X that accepts E and generates Σ^* (2.) building the product $H \times A_i$ with state set $X \times Q_i$ (3.) observing $m \in S_i^{[\mathbf{H2}]}$ if and only if $m \in P_i$ and for all x' such that $(x', m) \in X \times Q_i$ is reachable in $H \times A_i$, (x', m) is not coreachable in $H \times A_i$. Likewise, Heuristic [H3] can be represented as

$$S_i^{[\mathbf{H3}]} := \{ m \in P_i \,|\, \forall\, s \in T^* : \delta_i(m_o, s) = m \;\rightarrow\; s \in \mathrm{cfx}\,(\mathrm{L_m}(B_i)\,, E \cap \mathrm{L_m}(A_i)) \}\,. \tag{20}$$

The algorithm for the computation of the controllability prefix provided by Moor et al. (2020) returns its result as an alternative set of final states for the product of realisations of specification and plant. For the specific case at hand, we observe that we can find alternative final states for the product $H \times B_i$ to accept the languages $\mathrm{L_m}(B_i)$ and $E \cap \mathrm{L_m}(A_i)$. Thus, the controllability prefix can also be represented as an alternative set of final states for $H \times B_i$. Then, $m \in S_i^{[\mathbf{H3}]}$ if and only if $m \in P_i$ and for all x' such that $(x', m) \in X \times Q_i$ is reachable in $H \times B_i$, (x', m) is a final state in $H \times B_i$ when accepting the controllability prefix.

We illustrate Procedure 2 by the following example.

Example 15. Recall the Petri net $\mathcal{G} = (P, T, R, m_o, F)$ given in Figure 2. We now seek to solve a control problem with $\mathrm{L_m}(\mathcal{G})$ by using Procedure 2. The set of uncontrollable events is assumed $T_{uc} = \{y\}$ and the set of final markings is $F = \{[110011]^\top\}$, i.e. no workpieces must remain in the buffer places. Furthermore, we want to enforce a specification $E \subseteq T^*$, such that transitions x and y alternate and fire at least one time each. The firing of transitions a and b is not restricted. See Figure 4 for a realisation of an automaton C with $\mathrm{L_m}(C) = E$. Procedure 2 terminates for the given parameters in the fourth iteration and delivers a solution to the task at hand. The results are depicted in Figure 4 with $\mathrm{L_m}(D) = K^\uparrow$. Akin to earlier, transitions and final markings belonging solely to B_4, but not to A_4, are represented by dashed blue lines. □

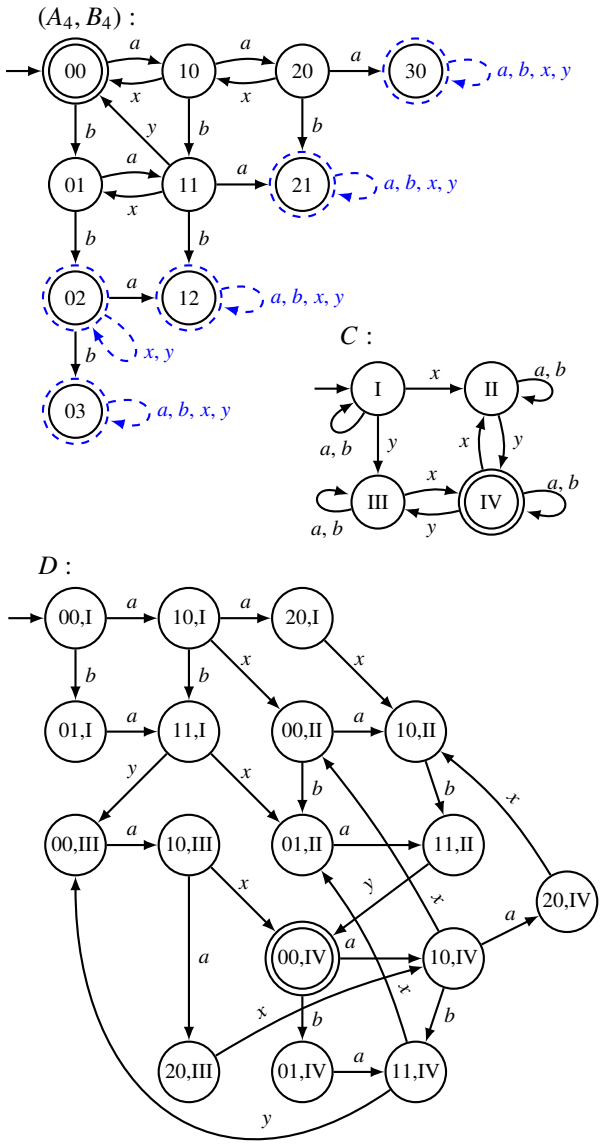

Fig. 4. Automata realisations for Example 15 with $E = \mathrm{L_m}(C)$ and $K^{\uparrow} = \mathrm{L_m}(D)$.

CONCLUSION

In this paper, we address the basic problem of supervisory control for non-regular parameters, and we consider unbounded Petri nets as a prototypical example. We propose to substitute the non-regular parameters with regular abstractions in a way that any solution for the substitute problem carries over to the original problem; Proposition 5. If no solution is found, a refinement of the abstraction is applied; Propositions 6 and 11. Iterating these two steps, we obtain a semi-algorithm, i.e., an algorithm that may not terminate, but in the case of termination provides a valid solution to the original problem; Proposition 13. Although not more can be expected given the known results e.g. on the decidability of controllability for deterministic context free parameters, we do envisage benefits for applications in which the non-regularity of a parameter is not essential for the solution of the control problem. In this regard, our iteration is guaranteed to terminate with success provided that there exists a supervisor which enforces a regular closed-loop behaviour with uniformly bounded markings; Proposition 14. The bound, however, does not need to be known in advance.

REFERENCES

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

Cordy, B. and Salomaa, K. (2007). On the existence of regular approximations. *Theoretical Computer Science*, 387(2), 125–135.

Eisman, G. and Ravikumar, B. (2005). Approximate recognition of non-regular languages by finite automata. *Twenty-Eighth Australasian Conference on Computer Science*, 38, 219–228.

Giua, A. (2013). *Supervisory Control of Petri Nets with Language Specifications*, 235–255. Springer.

Giua, A. and DiCesare, F. (1994). Blocking and controllability of petri nets in supervisory control. *IEEE Transactions on Automatic Control*, 39(4), 818–823.

Giua, A. and DiCesare, F. (1995). Decidability and closure properties of weak petri net languages in supervisory control. *IEEE Transactions on Automatic Control*, 40(5), 906–910.

Holloway, L., Krogh, B., and Giua, A. (1997). A survey of petri net methods for controlled discrete event systems. *Discrete Event Dynamic Systems*, 7, 151–190.

Masopust, T. (2012). A note on controllability of deterministic context-free systems. *Automatica*, 48(8), 1934–1937.

Moor, T. and Schmidt, K.W. (2017). The controllability prefix for supervisory control under partial observation. *20th IFAC WC, invited track DCDS*, 14206–14211.

Moor, T., Schmidt, K.W., and Schmuck, A.K. (2020). An efficient algorithm for the computation of the controllability prefix of *-languages. *21th IFAC WC*, 2122–2129.

Raisch, J. and O'Young, S. (1998). Discrete approximation and supervisory control of continuous systems. *IEEE Transactions on Automatic Control*, 43(4), 569—-573.

Ramadge, P.J. and Wonham, W.M. (1987). Supervisory control of a class of discrete event processes. *cp woSIAM 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.

Schmuck, A.K., Schneider, S., Raisch, J., and Nestmann, U. (2016). Supervisory control synthesis for deterministic context free specification languages. *Discrete Event Dynamic Systems*, 26, 5—32.

Sreenivas, R. (1993). On a weaker notion of controllability of a language K with respect to a language L. *IEEE Transactions on Automatic Control*, 38(9), 1446–1447.

Stursberg, O. (2006). Supervisory control of hybrid systems based on model abstraction and guided search. *Nonlinear Analysis: Theory, Methods & Applications*, 65(6), 1168–1187.

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

Wonham, W.M. and Cai, K. (2019). *Supervisory control of discrete-event systems*. Springer.

Wonham, W.M. and Ramadge, P.J. (1987). On the supremal controllable sublanguage of a given language. *SIAM Journal on Control and Optimization*, 25(3), 637–659.

Yang, J.M., Moor, T., and Raisch, J. (2020). Refinements of behavioural abstractions for the supervisory control of hybrid systems. *Discrete Event Dynamic Systems*, 30, 533–560.