Robust stutter bisimulation for abstraction and controller synthesis with disturbance

Automatica 160 (2024) 111394

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Brief paper

Robust stutter bisimulation for abstraction and controller synthesis with disturbance☆

Jonas Krook [a,c,*], Robi Malik [b], Sahar Mohajerani [d], Martin Fabian [a]

[a] Department of Electrical Engineering, Chalmers University of Technology, Göteborg, Sweden
[b] Department of Software Engineering, University of Waikato, Hamilton, New Zealand
[c] Zenseact, Göteborg, Sweden
[d] Mathworks, Boston, USA

ARTICLE INFO

Article history:
Received 31 May 2022
Received in revised form 5 October 2023
Accepted 11 October 2023
Available online 22 November 2023

Keywords:
Controller synthesis
Cyber-physical systems
Disturbances
Linear temporal logic
Abstraction

ABSTRACT

This paper proposes a method to synthesise controllers for cyber-physical systems subjected to disturbances, such that the controlled system satisfies specifications given as linear temporal logic formulas. To solve this problem, a finite-state abstraction of the original system is first constructed, and then a controller is synthesised for the abstraction. Due to the disturbances and uncertainty in the environment, future states cannot be predicted exactly, and the abstraction must take this into account. For this purpose, the *robust stutter bisimulation* relation is introduced, which preserves the existence of controllers for any given linear temporal logic formula that excludes the *next* operator. States are related by the robust stutter bisimulation relation if the same target sets can be guaranteed to be reached or avoided under control of some controller, thus ensuring that disturbances have similar effect on paths that start in related states. It is shown that there exists a controller enforcing a linear temporal logic formula for the original system if and only if a controller exists for the abstracted system. The approach is illustrated by a robot navigation example.

1. Introduction

A *cyber-physical system* (CPS) is an integration of a dynamical physical system and computers affecting its behaviour (Lee, 2015). This can be a continuous time dynamical system subject to process disturbances, under control of a computer with a fixed sampling time. CPSs are often safety critical, thus they must meet correctness guarantees. One way to achieve such guarantees is to use *formal synthesis* to construct the control logic automatically (Belta et al., 2017).

Synthesis computes, from a model of the CPS and a *formal specification*, the allowed control actions such that the behaviour of the controlled system satisfies the specification. The specification is a formalisation of the requirements, and can be expressed in, for instance, *Linear Temporal Logic* (LTL, Baier and Katoen 2008). LTL extends propositional logic with temporal operators so that requirements on future behaviour can be specified.

Standard algorithms (Belta et al., 2017; Kloetzer & Belta, 2008; Ramadge, 1989) for controller synthesis with LTL specifications require finite transition systems, whereas many CPSs are described by continuous models with infinite state spaces. Such models can be turned into transition systems by *discretisation*, but in general neither the state space nor the transition relation of the resulting *concrete* transition system are finite (Tabuada, 2009). Finite-state *abstraction*, though, groups states in a finite *quotient* state space. One such abstraction method is *bisimulation* (Milner, 1989). It preserves all LTL properties (Baier & Katoen, 2008), and is guaranteed to produce finite quotient spaces for certain types of systems (Alur et al., 2000).

If the bisimulation quotient of a system is infinite, a finite quotient space might be obtained by using a coarser abstraction. One approach to obtain coarser quotients is *approximate bisimulation* where bisimulation is relaxed to allow a bounded difference between the behaviours of the concrete and abstract system (Girard & Pappas, 2007). Coarser quotients also result from (approximate) *simulation* (Belta et al., 2017; Reissig et al., 2016; Tabuada, 2006; Zamani et al., 2011), which relaxes bisimulation by retaining only some controlled behaviours of the concrete system. *Dual-simulation* (Wagenmaker & Ozay, 2016) produces a coarser abstraction than bisimulation by using overlapping subsets. Another

☆ This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Michel Reniers under the direction of Editor Christos G. Cassandras.

* Corresponding author at: Department of Electrical Engineering, Chalmers University of Technology, Göteborg, Sweden.
E-mail addresses: krookj@chalmers.se (J. Krook), robi@waikato.ac.nz (R. Malik), smohajer@mathworks.com (S. Mohajerani), fabian@chalmers.se (M. Fabian).

https://doi.org/10.1016/j.automatica.2023.111394

coarse abstraction is obtained by *divergent stutter bisimulation* which allows *stutter steps* within the abstract states (Baier & Katoen, 2008). Mohajerani et al. (2021) show that divergent stutter bisimulation yields abstractions for which a controller can be synthesised if and only if a controller can be synthesised for the concrete system, given that the synthesis is performed with LTL specifications without the next operator ($\text{LTL}_{\setminus\circ}$). Without this operator specifications cannot refer to specific time intervals, but this is not always necessary for formalising the requirements. Unfortunately, the work by Mohajerani et al. (2021) applies to *deterministic* transition systems and do not work for systems subject to disturbances.

Disturbances, that stem from the system dynamics or fidelity loss in the discretisation, create *non-determinism*. Abstraction methods for controller synthesis for non-deterministic transition systems have been considered by Liu and Ozay (2016), using $\text{LTL}_{\setminus\circ}$. Also, Nilsson et al. (2012) consider abstractions of discrete-time systems subject to disturbances where N-step reachability is the basis for abstraction, and in the work by Nilsson et al. (2017) progress groups are introduced to allow coarser abstractions while preserving long-horizon reachability and progress properties. However, for these approaches, controllers that exist in the concrete system might not have a corresponding controller in the abstract system. Another abstraction approach is (approximate) *alternating bisimulation* (Alur et al., 1998; Pola & Tabuada, 2009), which is an extension of (approximate) bisimulation applicable for synthesis with LTL specifications on non-deterministic systems.

This paper addresses the problem of designing a controller for a given non-deterministic concrete transition system and an $\text{LTL}_{\setminus\circ}$ specification using an abstraction method that guarantees that a controller can be synthesised for the concrete system if and only if a controller can be synthesised for the abstract system. The abstractions are based on the *robust stutter bisimulation* relation introduced in this paper, which extends divergent stutter bisimulation to non-deterministic systems, and relaxes alternating bisimulation to $\text{LTL}_{\setminus\circ}$ specifications. As such, this work can be seen as a combination of the works by Mohajerani et al. (2021) and Pola and Tabuada (2009). It is shown how an abstract transition system can be constructed based on robust stutter bisimulation, and how this abstraction is used to synthesise and implement a controller for the concrete system.

2. Preliminaries

2.1. Transition systems

A *transition system* is a tuple $G = \langle S, \Sigma, \delta, S^\circ, \text{AP}, L\rangle$ with S a set of *states*, Σ a set of *transition labels*, $\delta \subseteq S \times \Sigma \times S$ a *transition relation*, $S^\circ \subseteq S$ a set of *initial states*, AP a set of *atomic propositions*, and $L : S \to 2^{\text{AP}}$ a *state labelling function*. Denote the set of all transition systems by $\mathcal{T}$.

Let S^* and S^ω denote the sets of all finite and infinite sequences of S, respectively, and $S^\infty = S^* \cup S^\omega$. The set of non-empty finite sequences is denoted by $S^+ = S^* \setminus \{\varepsilon\}$, where ε is the empty sequence. Two sequences $\rho \in S^*$ and $\pi \in S^\infty$ can be concatenated to form a new sequence $\rho\pi \in S^\infty$. A finite sequence $\rho \in S^*$ is a *prefix* of $\pi \in S^\infty$, written $\rho \sqsubseteq \pi$, if there exists a sequence $\pi' \in S^\infty$ such that $\rho\pi' = \pi$, and ρ is a *proper prefix* of π, written $\rho \sqsubset \pi$, if $\rho \sqsubseteq \pi$ and $\rho \neq \pi$. A finite sequence of states $\rho = s_0 \cdots s_n \in S^*$ is a *finite path fragment* of G if for all $0 \leq i < n$ there exists $\sigma_i \in \Sigma$ such that $(s_i, \sigma_i, s_{i+1}) \in \delta$. The set of all finite path fragments of G is denoted $\text{Frags}^*(G)$. An infinite sequence of states $\pi \in S^\omega$ is an *infinite path fragment* of G if for all finite prefixes $\rho \sqsubset \pi$ it holds that $\rho \in \text{Frags}^*(G)$. The set of all infinite path fragments of G is denoted $\text{Frags}^\omega(G)$. The set of all path fragments of G is $\text{Frags}^\infty(G) = \text{Frags}^*(G) \cup \text{Frags}^\omega(G)$. A *path* of G is an infinite path fragment $\pi = s_0 s_1 \cdots \in \text{Frags}^\omega(G)$ with $s_0 \in S^\circ$. The set of all paths of G is denoted $\text{Paths}^\omega(G)$. For $G \in \mathcal{T}$ the state labelling function L can be extended to sequences $\pi = s_0 s_1 \cdots \in \text{Frags}^\infty(G)$, as $L(\pi) = L(s_0)L(s_1)\cdots \in (2^{\text{AP}})^\infty$.

For $G_1, G_2 \in \mathcal{T}$ with $S_1 \cap S_2 = \emptyset$, their *union* is $G_1 \cup G_2 = \langle S_1 \cup S_2, \Sigma_1 \cup \Sigma_2, \delta_1 \cup \delta_2, S_1^\circ \cup S_2^\circ, \text{AP}_1 \cup \text{AP}_2, L\rangle$ with $L(s) = L_1(s)$ for $s \in S_1$, and $L(s) = L_2(s)$ for $s \in S_2$.

$G \in \mathcal{T}$ is *deadlock-free* if for each state $s \in S$ there exists a label $\sigma \in \Sigma$ and a state $t \in S$ such that $(s, \sigma, t) \in \delta$.

2.2. Relations

Given a set X, a relation $\mathcal{R} \subseteq X \times X$ is an *equivalence relation* on X if it is reflexive, symmetric, and transitive. The *equivalence class* of $x \in X$ is $[x]_\mathcal{R} = \{x' \in X \mid (x, x') \in \mathcal{R}\}$. The set of all equivalence classes modulo $\mathcal{R}$, the *quotient space* of X under $\mathcal{R}$, is $X/\mathcal{R} = \{[x]_\mathcal{R} \mid x \in X\}$. Partitioning into equivalence classes is one way to abstract the state set of a transition system; in the abstract system each equivalence class is one state. A relation $\mathcal{R}_1$ is a *refinement* of a relation $\mathcal{R}_2$ if $\mathcal{R}_1 \subseteq \mathcal{R}_2$, and then $\mathcal{R}_2$ is said to be *coarser* than $\mathcal{R}_1$.

Let $\mathcal{R} \subseteq X \times X$ be an equivalence relation. A set $T \subseteq X$ is a *superblock* of $\mathcal{R}$, if for all $x_1 \in T$ and all $x_2 \in X$ such that $(x_1, x_2) \in \mathcal{R}$, it holds that $x_2 \in T$. The set of all superblocks of $\mathcal{R}$ is denoted $\text{SB}(\mathcal{R})$. Superblocks are sets of elements that are closed under the equivalence relation $\mathcal{R}$. Alternatively, they can be characterised as unions of zero or more equivalence classes; thus, the empty set is also a superblock.

2.3. Linear temporal logic

A formula of *Linear Temporal Logic without Next* ($\text{LTL}_{\setminus\circ}$) (Baier & Katoen, 2008) is a logical formula over atomic propositions from a set AP, the propositional logic operators, and the binary operator $\mathcal{U}$. Its syntax is defined by $\varphi = \top \mid p \mid \neg\psi \mid \psi \wedge \theta \mid \psi \,\mathcal{U}\, \theta$, where $p \in \text{AP}$, and ψ and θ are $\text{LTL}_{\setminus\circ}$ formulas. For $G \in \mathcal{T}$, whether an infinite path fragment $\pi = s_0 s_1 \cdots \in \text{Frags}^\omega(G)$ satisfies the $\text{LTL}_{\setminus\circ}$ formula φ, written $\pi \vDash \varphi$, is defined inductively on the structure of φ:

- $\pi \vDash \top$ always holds;
- $\pi \vDash p$ iff $p \in L(s_0)$;
- $\pi \vDash \neg\psi$ iff $\pi \models \psi$ does not hold;
- $\pi \vDash \psi \wedge \theta$ iff $\pi \vDash \psi$ and $\pi \vDash \theta$;
- $\pi \vDash \psi \,\mathcal{U}\, \theta$ iff there is $m \geq 0$ such that $s_m s_{m+1} \cdots \vDash \theta$ and for all $0 \leq i < m$ it holds that $s_i s_{i+1} \cdots \vDash \psi$.

An $\text{LTL}_{\setminus\circ}$ formula φ holds at state s, written $\langle G, s\rangle \vDash \varphi$, if all infinite path fragments starting at s satisfy φ, i.e, for all infinite path fragments $\pi \in \text{Frags}^\omega(G)$ with $s \sqsubset \pi$, $\pi \models \varphi$. The transition system G *satisfies* φ, written $G \models \varphi$, if $\langle G, s^\circ\rangle \models \varphi$ for all initial states $s^\circ \in S^\circ$.

Additional temporal operators are defined based on the existing ones as $\Diamond\psi \equiv \top \,\mathcal{U}\, \psi$, $\Box\psi \equiv \neg\Diamond\neg\psi$, and $\psi \,\mathcal{W}\, \theta \equiv \Box\psi \vee (\psi \,\mathcal{U}\, \theta)$. The operators $\mathcal{U}$ and $\mathcal{W}$ are read as *(strong) until* and *weak until*. If $\psi \,\mathcal{U}\, \theta$ is satisfied on a path fragment, then θ eventually holds for some state sequence in the path fragment, and ψ holds in all state sequences before that. $\psi \,\mathcal{W}\, \theta$ is similar , but is also satisfied on path fragments where ψ holds in all states.

$\text{LTL}_{\setminus\circ}$ can be generalised over state sets $X \subseteq S$. For $\pi = s_0 s_1 \cdots \in \text{Frags}^\infty(G)$, let $\pi \vDash X$ iff $s_0 \in X$. Such $\text{LTL}_{\setminus\circ}$ formulas are called *generalised* $\text{LTL}_{\setminus\circ}$ formulas.

Let $G \in \mathcal{T}$. A *stutter step formula* for G is a generalised $\text{LTL}_{\setminus\circ}$ formula of the form $P \;\mathcal{V}\; T$, where $P, T \subseteq S$ with $P \cap T = \emptyset$ and $\mathcal{V} \in \{\mathcal{U}, \mathcal{W}\}$. Let $\mathcal{R} \subseteq S \times S$ be an equivalence relation. If

$P, T \in \mathrm{SB}(\mathcal{R})$, then the formula $P \ \mathcal{V} \ T$ is called an *$\mathcal{R}$-superblock step formula* from P.

Stutter step formulas $P \ \mathcal{V} \ T$ describe the immediate future when a system is in some state set P. The set T contains states that can be entered next from P. A stutter step formula of the form $P \ \mathcal{U} \ T$ means that a path visiting P will eventually reach T, whereas $P \ \mathcal{W} \ T$ means that it is possible to stay in P indefinitely or enter T. Superblock step formulas, where the source and target sets both are superblocks, describe the transition relation of an abstract system whose states are equivalence classes. $\mathcal{R}$-superblock step formulas make it possible to abstract away so-called *$\mathcal{R}$-stutter steps*, transitions within an equivalence class. Whether a path fragment satisfies an $\mathcal{R}$-superblock step formula is independent of the number of $\mathcal{R}$-stutter steps. If $\mathcal{R}$ preserves state labels, then it can also be shown that the satisfaction of $\mathrm{LTL}_{\setminus\circ}$ formulas is independent of such steps.

For $\pi = s_0 s_1 \cdots \in \mathrm{Frags}^\infty(G)$, its *stutter free* sequence $\mathrm{sf}(\pi) \in S^\infty$ is obtained from π by removing all elements s_{i+1} such that $s_{i+1} = s_i$. Two path fragments $\pi_1, \pi_2 \in \mathrm{Frags}^\infty(G)$ are *stutter equivalent* if $\mathrm{sf}(L(\pi_1)) = \mathrm{sf}(L(\pi_2))$, and if both are either finite or infinite. The *trace of equivalence classes* of π with respect to an equivalence relation $\mathcal{R}$ is $[\pi]_\mathcal{R} = [s_0]_\mathcal{R}[s_1]_\mathcal{R} \cdots$. Then π_1, π_2 are *$\mathcal{R}$-stutter equivalent* if $\mathrm{sf}([\pi_1]_\mathcal{R}) = \mathrm{sf}([\pi_2]_\mathcal{R})$.

Theorem 1 (*Baier and Katoen 2008*)**.** *Let $G \in \mathcal{T}$, let φ be an $\mathrm{LTL}_{\setminus\circ}$ formula, and let $\pi_1, \pi_2 \in \mathrm{Frags}^\omega(G)$ be two stutter equivalent path fragments. Then $\pi_1 \vDash \varphi$ iff $\pi_2 \vDash \varphi$.*

2.4. Controllers

Transition systems can be controlled by dynamically restricting the set of allowed transitions. A controller does this by deciding, based on the history of visited states, the set of allowed transition labels. Thus, a *(general) controller* for $G \in \mathcal{T}$ is a function $C : S^+ \to 2^\Sigma$.

As a controller controls a transition system G, the resultant behaviour is described by another transition system, called the *controlled* system, the states of which are sequences of states of the original transition system G.

Let $G \in \mathcal{T}$, and $C : S^+ \to 2^\Sigma$ be a controller for G. The *controlled system* is $C/G = \langle S^+, \Sigma, \delta_C, S^\circ, \mathrm{AP}, L_C \rangle$, with

$$\delta_C = \left\{ (s_0 \cdots s_n, \sigma, s_0 \cdots s_n s_{n+1}) \in S^+ \times \Sigma \times S^+ \mid (s_n, \sigma, s_{n+1}) \in \delta \text{ and } \sigma \in C(s_0 \cdots s_n) \right\}.$$

and $L_C(s_0 \cdots s_n) = L(s_n)$.

The initial states of the controlled system C/G are the initial states of the original system G, interpreted as sequences of length one. After observing a state sequence $s_0 \cdots s_n$, the state of G is s_n and the state of C/G is $s_0 \cdots s_n$. The transitions from this state in C/G are those possible in G from s_n and allowed by C from $s_0 \cdots s_n$. C is said to be *deadlock-free* if C/G is deadlock-free. Let $\mathcal{C}$ denote all deadlock-free controllers.

C *enforces* a (generalised) $\mathrm{LTL}_{\setminus\circ}$-formula ψ from state s of G if $\langle C/G, s \rangle \vDash \psi$, and C enforces ψ on G if $C/G \vDash \psi$. That is, an $\mathrm{LTL}_{\setminus\circ}$ formula is enforced from a state if every controlled path fragment starting from that state satisfies the formula. An $\mathrm{LTL}_{\setminus\circ}$ formula is enforced on a transition system if it is enforced from every initial state.

The path fragments of a controlled system C/G are given by $\mathrm{Frags}^\infty(C/G)$, which by definition are sequences of states of C/G and thus sequences of sequences of states of G. Such path fragments $(s_0)(s_0 s_1) \cdots (s_0 s_1 \cdots s_n) \cdots$ of the controlled system C/G can be replaced by path fragments $s_0 s_1 \cdots s_n \cdots$ of G. Let $C : S^+ \to 2^\Sigma$ be a controller for G. A finite path fragment $\rho = s_0 s_1 s_2 \cdots s_n \in \mathrm{Frags}^*(G)$ is *permitted* by C if for all $0 \leq i < n$ there exists $\sigma_i \in C(s_0 \cdots s_i)$ such that $(s_i, \sigma_i, s_{i+1}) \in \delta$. The set of all finite path fragments in G permitted by C is denoted by $\mathrm{Frags}^*(C, G)$. This can be extended to permitted infinite path fragments, $\mathrm{Frags}^\omega(C, G)$, and permitted path fragments, $\mathrm{Frags}^\infty(C, G)$. The set of permitted paths, $\mathrm{Paths}^\omega(C, G)$, is defined likewise.

2.5. Positional controllers

A *positional controller* for $G \in \mathcal{T}$ is a function $\overline{C} : S \to 2^\Sigma$. As a positional controller only makes decisions based on the current state, its controlled system can be alternatively defined using the original transition system state space. The *controlled system* of G under the control of $\overline{C}$ is $\overline{C}/G = C/G$ where $C : S^+ \to 2^\Sigma$ with $C(\rho u) = \overline{C}(u)$ for all $\rho \in S^*$ and $u \in S$. Let $\bar{\mathcal{C}}$ denote all deadlock-free positional controllers.

Let ψ be a generalised $\mathrm{LTL}_{\setminus\circ}$-formula for $G \in \mathcal{T}$. The set of states where ψ can be enforced by control is:

$$\mathcal{EC}_G(\psi) = \left\{ s \in S \mid \exists C \in \mathcal{C} \text{ s.t. } \langle C/G, s \rangle \vDash \psi \right\};$$
$$\overline{\mathcal{EC}}_G(\psi) = \left\{ s \in S \mid \exists \overline{C} \in \bar{\mathcal{C}} \text{ s.t. } \langle \overline{C}/G, s \rangle \vDash \psi \right\}.$$

It is clear that $\overline{\mathcal{EC}}_G(\psi) \subseteq \mathcal{EC}_G(\psi)$ for every generalised $\mathrm{LTL}_{\setminus\circ}$ formula ψ. For stutter step formulas, i.e., generalised $\mathrm{LTL}_{\setminus\circ}$ formulas of the form $P \ \mathcal{U} \ T$ and $P \ \mathcal{W} \ T$, also the converse holds.

Proposition 2. *Let $G \in \mathcal{T}$ be deadlock-free, and let ψ be a stutter step formula for G. Then $\mathcal{EC}_G(\psi) = \overline{\mathcal{EC}}_G(\psi)$, and there exists a $\overline{C} \in \bar{\mathcal{C}}$ such that, for all states $s \in \overline{\mathcal{EC}}_G(\psi)$ it holds that $\langle \overline{C}/G, s \rangle \vDash \psi$.*

Proof. Let $\psi \equiv P \ \mathcal{V} \ T$ be a stutter step formula. Clearly, $\overline{\mathcal{EC}}_G(\psi) \subseteq \mathcal{EC}_G(\psi)$, as a positional controller is a special case of a general controller. It remains to show $\mathcal{EC}_G(\psi) \subseteq \overline{\mathcal{EC}}_G(\psi)$. Assume that $s_0 \in \mathcal{EC}_G(\psi)$. Either it holds that $s_0 \in T$ or $s_0 \in P$. If $s_0 \in T$, clearly $s_0 \in \overline{\mathcal{EC}}_G(\psi)$. The proof continues on a case-by-case basis for $s_0 \in P$.

If $\psi \equiv P \ \mathcal{W} \ T$ it must be shown that there exists a controller $\overline{C} \in \bar{\mathcal{C}}$ such that all infinite path fragments $\pi \in \mathrm{Frags}^\omega(\overline{C}, G)$ has the property that either there exists a path fragment $\tau = \tau' t \sqsubseteq \pi$ with $s_0 \sqsubseteq \tau' \in P^*$ and $t \in T$, or there exists a path fragment $\tau \sqsubseteq \pi$ with $s_0 \sqsubseteq \tau \in P^\omega$. Since there is a controller $C \in \mathcal{C}$ that enforces ψ from s_0, there is always a choice σ from any state $s_1 \in P \cap \mathcal{EC}_G(\psi)$ such that $C(s_1, \sigma) \subseteq P \cup T$. $\overline{C}$ can therefore always enforce the next possible states to be in $P \cup T$, which means all controlled path fragments have the desired property.

The other case is when $\psi \equiv P \ \mathcal{U} \ T$. For a contradiction, assume that there does not exist a controller $\overline{C} \in \bar{\mathcal{C}}$ that enforces ψ from s_0. Then there must exist an $s_1 \in P \cap C(s_0)$ such that there does not exist a controller $\overline{C}' \in \bar{\mathcal{C}}$ that enforces ψ from s_1. It holds that $\langle C/G, s_0 s_1 \rangle \vDash P \ \mathcal{U} \ T$, which means that $s_1 \in \mathcal{EC}_G(\psi)$. By induction, it is possible to construct an infinite path fragment $\pi \in \mathrm{Frags}^\omega(C, G)$ with $\pi = s_0 s_1 \cdots \in P^\omega$. This contradicts $\langle C/G, s_0 \rangle \vDash P \ \mathcal{U} \ T$. □

3. Controller synthesis by abstraction

This paper concerns synthesis of a controller enforcing an $\mathrm{LTL}_{\setminus\circ}$ formula φ on a *concrete* transition system G, as illustrated by the dashed arrow in Fig. 1. Typically, G models a discrete-time dynamical system derived from difference equations with infinite state, control, and disturbance spaces, thus G's state space is infinite.

Synthesis Problem. Given $G \in \mathcal{T}$ and an $\mathrm{LTL}_{\setminus\circ}$ formula φ, find $C \in \mathcal{C}$ that enforces φ on G.

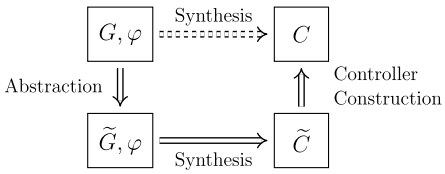

Fig. 1. Illustration of controller synthesis by abstraction. The dashed arrow represents synthesis directly on the concrete system, the solid arrows represent synthesis by abstraction.

To apply methods for finite transition systems to infinite state spaces, an *abstraction*-based approach is used, shown by the solid arrows in Fig. 1. First, the concrete transition system G is transformed into a finite-state abstraction $\tilde{G}$. Then is synthesised an *abstract controller* $\tilde{C}$ that enforces φ on $\tilde{G}$. Third, a *concrete controller* C, which enforces φ on G, is constructed from $\tilde{C}$. As synthesis of controllers for $\text{LTL}_{\setminus\circ}$ specifications is known (Kloetzer & Belta, 2008; Ramadge, 1989), this paper only concerns the first and third steps, i.e., constructing the abstraction $\tilde{G}$ and the concrete controller C.

Abstraction Problem. Given a concrete $G \in \mathcal{T}$, and an $\text{LTL}_{\setminus\circ}$ formula φ, construct an abstract $\tilde{G} \in \mathcal{T}$ such that, there exists $\tilde{C} \in \mathcal{C}$ enforcing φ on $\tilde{G}$ iff there exists $C \in \mathcal{C}$ enforcing φ on G.

To solve the Abstraction Problem, the abstract transition system $\tilde{G}$ must be constructed so that it is equivalent with respect to synthesis to the concrete system G. This ensures *soundness* and *completeness* of the approach, i.e., any controller obtained from $\tilde{G}$ is related to a controller for G, and if there exists a controller for G, then it can be found by synthesis based on $\tilde{G}$.

Though the existence of a controller $\tilde{C}$ for $\tilde{G}$ carries over to the existence of a controller C for G, it remains to find a method to construct C from $\tilde{C}$.

Controller Construction Problem. Given a concrete $G \in \mathcal{T}$, an abstraction $\tilde{G}$ of G, an $\text{LTL}_{\setminus\circ}$ formula φ, and $\tilde{C} \in \mathcal{C}$ that enforces φ on $\tilde{G}$, construct $C \in \mathcal{C}$ that enforces φ on G.

4. Robust stutter bisimulation

This section defines the *robust stutter bisimulation* (RSBS) relation, which identifies two transition systems as equivalent if the same $\text{LTL}_{\setminus\circ}$ formulas can be enforced on both systems. This is the crucial criterion to identify solutions to the Abstraction Problem, ensuring that any transition system $\tilde{G}$ that is robust stutter bisimilar to the concrete system G can be used as its abstraction.

The goal is to classify two states of a transition system as robust stutter bisimilar when the same $\text{LTL}_{\setminus\circ}$ formulas can be enforced from both. Due to the absence of the "next" operator, this condition can be simplified by considering only stutter step formulas, or, more precisely, superblock step formulas. Given an equivalence class P, a superblock step formula $P \,\mathcal{U}\, T$ or $P \,\mathcal{W}\, T$ expresses that the system can transition to other equivalence classes with or without the guarantee that this transition occurs eventually. The condition can be further simplified to superblock step formulas that can be enforced by positional controllers because of Proposition 2. Controllers for arbitrary $\text{LTL}_{\setminus\circ}$ formulas can be constructed by combining positional controllers enforcing stutter step formulas from different states.

Definition 1. An equivalence relation $\mathcal{R} \subseteq S \times S$ is a *robust stutter bisimulation* (RSBS) on $G \in \mathcal{T}$ if for all $(s_1, s_2) \in \mathcal{R}$ the following conditions hold:

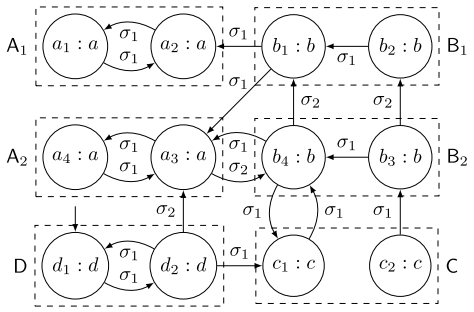

Fig. 2. A transition system G with an RSBS $\mathcal{R}$.

(i) $L(s_1) = L(s_2)$
(ii) for every $\mathcal{R}$-superblock step formula ψ from $[s_1]_{\mathcal{R}}$ such that $s_1 \in \overline{\mathcal{EC}}_G(\psi)$ it holds that $s_2 \in \overline{\mathcal{EC}}_G(\psi)$.

Two states $s_1, s_2 \in S$ are *robust stutter bisimilar*, denoted $s_1 \approx s_2$, if there exists an RSBS relation $\mathcal{R}$ on G with $(s_1, s_2) \in \mathcal{R}$.

So, a relation $\mathcal{R}$ is an RSBS if (i) it preserves the labels of states, and (ii) the same superblock step formulas can be enforced from equivalent states. Though (ii) is not inherently symmetric, symmetry is ensured as $\mathcal{R}$ is an equivalence relation. $\mathcal{R}$ is required to be an equivalence relation so that RSBS can be defined in terms of superblock step formulas. Since the identity relation is an RSBS, it is clear that there exist RSBS subject to the additional requirement of being an equivalence relation. It is shown in Proposition D.2 by Krook et al. (2022) that a coarsest RSBS exists and is an equivalence relation. By Proposition 2 it is enough to define enforceability through $\overline{\mathcal{EC}}_G(\psi)$, i.e., using positional controllers. Definition 1 can be seen as a generalisation of *divergent stutter bisimulation* (Baier & Katoen, 2008) to transition systems with disturbances.

Example 1. Consider the transition system $G = \langle S, \Sigma, \delta, S^\circ, \text{AP}, L\rangle$ in Fig. 2 and the equivalence relation $\mathcal{R}$ that divides the state space into six equivalence classes $S/\mathcal{R} = \{\mathsf{A}_1, \mathsf{A}_2, \mathsf{B}_1, \mathsf{B}_2, \mathsf{C}, \mathsf{D}\}$ where $\mathsf{A}_1 = \{a_1, a_2\}$, $\mathsf{A}_2 = \{a_3, a_4\}$, $\mathsf{B}_1 = \{b_1, b_2\}$, $\mathsf{B}_2 = \{b_3, b_4\}$, $\mathsf{C} = \{c_1, c_2\}$, and $\mathsf{D} = \{d_1, d_2\}$. Clearly, $\mathcal{R}$ fulfils condition (i) in Definition 1 as all states in each class are labelled the same from $\text{AP} = \{a, b, c, d\}$. By enumerating all deadlock-free positional controllers and all $\mathcal{R}$-superblock step formulas it can also be verified that condition (ii) is fulfilled by $\mathcal{R}$. □

To identify two transition systems as equivalent, the definition of an RSBS on a single transition system is lifted to a relation between transition systems by considering the union transition system.

Definition 2. Let $G_1, G_2 \in \mathcal{T}$ with $S_1 \cap S_2 = \emptyset$. G_1 and G_2 are robust stutter bisimilar, $G_1 \approx G_2$, if there exists a RSBS $\mathcal{R}$ on $G_1 \cup G_2$, and it holds that

$$\forall s_1 \in {S_1}^\circ.\, \exists s_2 \in {S_2}^\circ.\, (s_1, s_2) \in \mathcal{R}\ , \quad \text{and}$$
$$\forall s_2 \in {S_2}^\circ.\, \exists s_1 \in {S_1}^\circ.\, (s_1, s_2) \in \mathcal{R}\ .$$

5. Quotient transition system

Given a RSBS $\mathcal{R}$ on the states of $G \in \mathcal{T}$, the Abstraction Problem still requires to construct an equivalent and hopefully smaller transition system. This can be done by constructing a *quotient* transition system whose states are the equivalence classes induced by $\mathcal{R}$. Standard quotient constructions (Baier & Katoen, 2008) fail to ensure RSBS between the original and quotient systems since they do not distinguish the different stutter step formulas. Thus, the following definition.

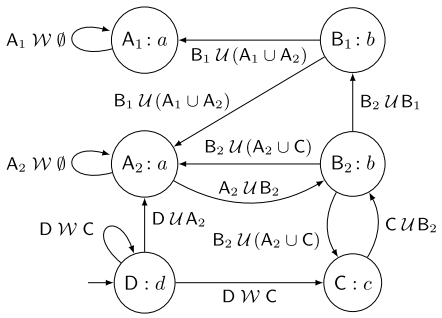

Fig. 3. A part of the quotient system $G/\mathcal{R}$.

Definition 3. Let $G \in \mathcal{T}$, and $\mathcal{R}$ be a RSBS on G. The *robust stutter bisimulation quotient* is

$$G/\mathcal{R} = \langle S/\mathcal{R},\, \Sigma_{\mathcal{R}},\, \delta_{\mathcal{R}},\, S^{\circ}_{\mathcal{R}},\, \mathrm{AP},\, L_{\mathcal{R}} \rangle$$

where

$$\Sigma_{\mathcal{R}} = \left\{ \psi \mid \psi \text{ is an } \mathcal{R}\text{-superblock step formula from } P \in S/\mathcal{R} \text{ such that } P \subseteq \overline{\mathcal{EC}}_G(\psi) \right\}; \tag{1}$$

$$\delta_{\mathcal{R}} = \left\{ (P, \psi, T') \in S/\mathcal{R} \times \Sigma_{\mathcal{R}} \times S/\mathcal{R} \mid \psi \equiv P\ \mathcal{V}\ T \text{ such that } T' \in S/\mathcal{R} \text{ and } T' \subseteq T \right\} \cup \left\{ (P, \psi, P) \in S/\mathcal{R} \times \Sigma_{\mathcal{R}} \times S/\mathcal{R} \mid \psi \equiv P\ \mathcal{W}\ T \text{ for some } T \in \mathrm{SB}(\mathcal{R}) \right\}; \tag{2}$$

$$S^{\circ}_{\mathcal{R}} = \left\{ [s^{\circ}]_{\mathcal{R}} \mid s^{\circ} \in S^{\circ} \right\};$$

and $L_{\mathcal{R}}([s]_{\mathcal{R}}) = L(s)$ for all $s \in S$.

The transition labels $\Sigma_{\mathcal{R}}$ are $\mathcal{R}$-superblock step formulas of the form $P\ \mathcal{V}\ T$ that can be enforced from the source set P in the concrete system G. The quotient transitions $\delta_{\mathcal{R}}$ are then defined based on the formulas these labels represent. A label $P\ \mathcal{V}\ T$, where P is an equivalence class and T is a superblock, is attached to transitions from P to each equivalence class (each abstract state) that constitutes T, representing the controller's ability to force the system to T without being able to determine the precise concrete state entered. A label $P\ \mathcal{W}\ T$ is additionally attached to a selfloop transition on P, representing the fact that the system may also stay in P. This definition works precisely because it distinguishes between the different $\mathcal{R}$-superblock step formulas $P\,\mathcal{U}\,T$ and $P\,\mathcal{W}\,T$, and generates transitions that ensure that all possible control actions from P are accounted for.

Example 2. Consider again G and RSBS $\mathcal{R}$ of Fig. 2. Its quotient system $G/\mathcal{R}$ is given in Fig. 3. The states are the six equivalence classes and their labels match the corresponding concrete states' labels. The transition labels in $\Sigma_{\mathcal{R}}$ are constructed based on the existence of positional controllers. A positional controller $\overline{C}(s) = \{\sigma_1\}$ enforces different $\mathcal{R}$-superblock step formulas from different equivalence classes. For instance, $\langle \overline{C}/G, a_i \rangle \vDash \mathsf{A}_1\ \mathcal{W}\ \emptyset$ for $a_i \in \mathsf{A}_1$, so $\mathsf{A}_1 \subseteq \overline{\mathcal{EC}}_G(\mathsf{A}_1\ \mathcal{W}\ \emptyset)$. Clearly, $\mathsf{A}_1 \in S/\mathcal{R}$, so $\psi_1 \equiv \mathsf{A}_1\ \mathcal{W}\ \emptyset \in \Sigma_{\mathcal{R}}$ by (1). Other formulas such as $\mathsf{A}_1\ \mathcal{W}\ \mathsf{B}_1$ are implied by ψ_1 and can also be enforced, so a total of 31 formulas $\mathsf{A}_1\ \mathcal{W}\ T$ with $T \subseteq \{\mathsf{A}_2, \mathsf{B}_1, \mathsf{B}_2, \mathsf{C}, \mathsf{D}\}$ are included in $\Sigma_{\mathcal{R}}$ (not all shown in the figure). According to (2), the label $\mathsf{A}_1\ \mathcal{W}\ \emptyset$ is attached to the selfloop transition $\mathsf{A}_1 \to \mathsf{A}_1$, and $\mathsf{A}_1\ \mathcal{W}\ \mathsf{B}_1$ is attached to $\mathsf{A}_1 \to \mathsf{A}_1$ and $\mathsf{A}_1 \to \mathsf{B}_1$, etc. On the other hand, no stutter step formula $\mathsf{A}_1\ \mathcal{U}\ T$ is enforceable from A_1, so those formulas do not appear in $\Sigma_{\mathcal{R}}$. From state d_2, it is not possible to force a transition to only one of d_1 or c_1, but it holds that $\langle \overline{C}/G, d_i \rangle \vDash \mathsf{D}\ \mathcal{W}\ \mathsf{C}$. Hence, $\mathsf{D}\ \mathcal{W}\ \mathsf{C} \in \Sigma_{\mathcal{R}}$ along with implied formulas. □

The following theorem confirms that the quotient $G/\mathcal{R}$ of a concrete $G \in \mathcal{T}$ with respect to a RSBS $\mathcal{R}$ is robust stutter bisimilar to G.

Theorem 3. *Let $G \in \mathcal{T}$, and $\mathcal{R}$ be a RSBS on G. Then $G \approx G/\mathcal{R}$.*

Proof. It will be shown that the following relation $\hat{\mathcal{R}}$ on $\hat{G} = G \cup (G/\mathcal{R})$ fulfils the conditions of Definition 2.

$$\hat{\mathcal{R}} = \mathcal{R} \cup \{ (\tilde{s}, \tilde{s}) \mid \tilde{s} \in S/\mathcal{R} \} \cup \{ (s, [s]_{\mathcal{R}}) \mid s \in S \} \cup \{ ([s]_{\mathcal{R}}, s) \mid s \in S \}. \tag{3}$$

Note that $\hat{\mathcal{R}}$ is an equivalence relation by construction. It is clear that condition (i) holds by definition of $\mathcal{R}$ and $L_{\mathcal{R}}$. For condition (ii), let $(s,t) \in \hat{\mathcal{R}}$, and let $\hat{\psi} \equiv [s]_{\hat{\mathcal{R}}}\ \mathcal{V}\ \hat{T}$ be an $\hat{\mathcal{R}}$-superblock step formula such that $s \in \overline{\mathcal{EC}}_{\hat{G}}(\hat{\psi})$. Also let $T = \hat{T} \cap S \in \mathrm{SB}(\mathcal{R})$ and $\psi \equiv [s]_{\mathcal{R}}\ \mathcal{V}\ T$. It is to be shown that $t \in \overline{\mathcal{EC}}_{\hat{G}}(\hat{\psi})$. Consider four cases.

Case 1: $s, t \in S$. As $\hat{G}$ is a disjoint union of G and $G/\mathcal{R}$, it follows that $s \in \overline{\mathcal{EC}}_{\hat{G}}(\hat{\psi}) \cap S = \overline{\mathcal{EC}}_{\hat{G}}(([s]_{\hat{\mathcal{R}}} \cap S)\ \mathcal{V}\ (\hat{T} \cap S)) = \overline{\mathcal{EC}}_G(\psi)$. As $(s,t) \in \mathcal{R}$ and $\mathcal{R}$ is an RSBS, it holds that $t \in \overline{\mathcal{EC}}_G(\psi)$. By the equality above, it also holds that $t \in \overline{\mathcal{EC}}_{\hat{G}}(\hat{\psi})$.

Case 2: $s \in S$, $t \in S/\mathcal{R}$. As above, it follows that $s \in \overline{\mathcal{EC}}_G(\psi)$. Note that $t = [s]_{\mathcal{R}}$ by definition of $\hat{\mathcal{R}}$ so $t \subseteq \overline{\mathcal{EC}}_G(\psi)$ as $\mathcal{R}$ is an RSBS. By (1) in Definition 3 it follows that $\psi \in \Sigma_{\mathcal{R}}$. Consider a positional controller $\overline{C}$ for G such that $\overline{C}([s]_{\mathcal{R}}) = \{\psi\}$. Then $\langle \overline{C}/\hat{G}, [s]_{\mathcal{R}} \rangle \vDash \psi$ by construction (2) and thus $t \in [s]_{\mathcal{R}} \in \overline{\mathcal{EC}}_G(\psi) \subseteq \overline{\mathcal{EC}}_{\hat{G}}(\hat{\psi})$ as in Case 1.

Case 3: $s \in S/\mathcal{R}$, $t \in S$. Then $s \in \overline{\mathcal{EC}}_{\hat{G}}(\hat{\psi}) \cap S/\mathcal{R} = \overline{\mathcal{EC}}_{G/\mathcal{R}}(\{s\}\ \mathcal{V}\ (\hat{T} \cap S/\mathcal{R}))$. Let $T_{\mathcal{R}} = \hat{T} \cap S/\mathcal{R}$ and $\psi_{\mathcal{R}} = \{s\}\ \mathcal{V}\ T_{\mathcal{R}}$. Then there exists $\hat{C} \in \overline{C}_{G/\mathcal{R}}$ such that $\langle \hat{C}/(G/\mathcal{R}), s \rangle \vDash \psi_{\mathcal{R}}$. As $\hat{C}$ is deadlock-free, there exists $\theta \in \Sigma_{\mathcal{R}}$ such that $\theta \in \hat{C}(s)$ where $\theta \equiv s\ \mathcal{V}_\theta\ T_\theta$ for some $T_\theta \in \mathrm{SB}(S)$ by (2).

It is next shown that $T_\theta \subseteq \hat{T}$. Let $t' \in T_\theta$. Then $[t']_{\mathcal{R}} \subseteq T_\theta$ as $T_\theta \in \mathrm{SB}(S)$, and $(s, \theta, [t']_{\mathcal{R}}) \in \delta_{\mathcal{R}}$ by (2). As $\langle \hat{C}/(G/\mathcal{R}), s \rangle \vDash \psi_{\mathcal{R}} \equiv \{s\}\ \mathcal{V}\ T_{\mathcal{R}}$ and $\theta \in \hat{C}(s)$, it follows that $[t']_{\mathcal{R}} \in \{s\} \cup T_{\mathcal{R}}$. Also $s \cap T_\theta = \emptyset$ as θ is a superblock step formula, so that $[t']_{\mathcal{R}} \in T_{\mathcal{R}} \subseteq \hat{T}$. Finally, as $(t', [t']_{\mathcal{R}}) \in \hat{\mathcal{R}}$ and $\hat{T} \in \mathrm{SB}(\hat{\mathcal{R}})$, it follows that $t' \in \hat{T}$.

As $\theta \in \Sigma_{\mathcal{R}}$, it follows by (1) that $t \in s \subseteq \overline{\mathcal{EC}}_G(\theta) = \overline{\mathcal{EC}}_G(s\ \mathcal{V}_\theta\ T_\theta) \subseteq \overline{\mathcal{EC}}_G(s\ \mathcal{V}_\theta\ (\hat{T} \cap S)) \subseteq \overline{\mathcal{EC}}_{\hat{G}}([s]_{\hat{\mathcal{R}}}\ \mathcal{V}_\theta\ \hat{T})$. Further, if $\mathcal{V} = \mathcal{U}$, then it follows from $\langle \hat{C}/(G/\mathcal{R}), s \rangle \vDash \psi_{\mathcal{R}}$ that $s^\omega \notin \mathrm{Frags}^\omega(\hat{C}/(G/\mathcal{R}))$, which means $(s, \theta, s) \notin \delta_{\mathcal{R}}$ by (2) and then $\mathcal{V}_\theta = \mathcal{U}$. Therefore, $t \in \overline{\mathcal{EC}}_{\hat{G}}([s]_{\hat{\mathcal{R}}}\ \mathcal{V}_\theta\ \hat{T}) \subseteq \overline{\mathcal{EC}}_{\hat{G}}([s]_{\hat{\mathcal{R}}}\ \mathcal{V}\ \hat{T}) = \overline{\mathcal{EC}}_{\hat{G}}(\hat{\psi})$.

Case 4: $s, t \in S/\mathcal{R}$. This case is trivial as $s = t$ by (3).

Lastly, Definition 2 requires that states in S° must be matched by states in $S^\circ_{\mathcal{R}}$, and vice versa, and this follows directly from the construction of $S^\circ_{\mathcal{R}}$ and $\hat{\mathcal{R}}$. □

6. Constructing a concrete controller

Given $G, \tilde{G} \in \mathcal{T}$ such that $G \approx \tilde{G}$, and a controller $\tilde{C}$ enforcing an $\mathrm{LTL}_{\setminus\circ}$ formula φ on $\tilde{G}$, this section shows how to construct a controller C that enforces φ on G. In the case where $\tilde{G} = G/\mathcal{R}$, this solves the Controller Construction Problem.

As $G \approx \tilde{G}$, by Definition 2 there exists an RSBS on $G \cup \tilde{G}$. Then $\tilde{C}$ can be regarded to enforce φ from all states $\tilde{s}$ of $G \cup \tilde{G}$ that are initial states of $\tilde{G}$, and C to enforce φ from equivalent initial states s of G. Thus, it is enough to consider an RSBS $\mathcal{R}$ on a single $G \in \mathcal{T}$ that may be a union transition system, and to consider two equivalent states $(s, \tilde{s}) \in \mathcal{R}$. Given $\tilde{C}$ such that $\langle \tilde{C}/G, \tilde{s} \rangle \vDash \varphi$, the goal is to construct C such that $\langle C/G, s \rangle \vDash \varphi$.

The solution to the Controller Construction Problem is such that if C observes a path fragment $\rho \in S^+$ with $s \sqsubseteq \rho$, this path fragment is mapped by a function $M : S^+ \to S^+$ to a nonempty stutter equivalent path fragment $\tilde{\rho}$ permitted by $\tilde{C}$, such that

$\tilde{s} \sqsubseteq \tilde{\rho}$. The control action of C after ρ is then guided by the control action of $\tilde{C}$ after $\tilde{\rho}$.

Assume that some path fragment $\rho = \rho' u$, with $\rho' \in S^*$ and $u \in S$, has been mapped to $\tilde{\rho} = \tilde{\rho}'\tilde{u}$, with $\tilde{\rho}' \in S^*$ and $\tilde{u} \in S$. If G under control of C now enters a state $v \in S$ in a new equivalence class, i.e., $(u, v) \notin \mathcal{R}$, then the construction of M considers all continuations of $\tilde{\rho} = \tilde{\rho}'\tilde{u}$ permitted by $\tilde{C}$ that remain in the equivalence class of $\tilde{u}$ until reaching a state $\tilde{v}$ equivalent to v, namely

$$F(\tilde{\rho}'\tilde{u}, v) = \left\{ \tilde{\rho}'\tilde{u}\tilde{\tau}\tilde{v} \in \mathrm{Frags}^*(\tilde{C}, G) \mid \tilde{\tau} \in [\tilde{u}]^*_{\mathcal{R}} \text{ and } (v, \tilde{v}) \in \mathcal{R} \right\}. \tag{4}$$

Then M is defined by choosing one of these paths. The full recursive definition of $M(\rho)$ for $\rho \in S^+$ with $s \sqsubseteq \rho$ is:

$$M(s) = \tilde{s}\ ; \tag{5}$$

$$M(\rho' uv) = M(\rho' u) \qquad \text{if } (u, v) \in \mathcal{R}\ ; \tag{6}$$

$$M(\rho' uv) = \tau \in F(M(\rho' u), v) \qquad \text{if } (u, v) \notin \mathcal{R}\ ; \tag{7}$$

where τ is an arbitrary but fixed choice for each $\rho' uv$. If $M(\rho' u)$ is undefined or $F(M(\rho' u), v) = \emptyset$ in (6) or (7), then $M(\rho' uv)$ is undefined. Later on, the controller C will be constructed so that $M(\rho)$ is defined for all $\rho \in \mathrm{Frags}^*(C, G)$ with $s \sqsubseteq \rho$. If $M(\rho)$ is defined, then $M(\rho) \in \mathrm{Frags}^*(\tilde{C}, G)$, $\tilde{s} \sqsubseteq M(\rho)$, and $M(\rho)$ and ρ are $\mathcal{R}$-stutter equivalent. Furthermore, M is prefix-preserving, i.e., $\rho' \sqsubseteq \rho$ implies $M(\rho') \sqsubseteq M(\rho)$.

Since $\tilde{C} \in \mathcal{C}$, it enforces at least one $\mathcal{R}$-superblock step formula ψ from $\tilde{u}$ after seeing $\tilde{\rho}$. As $u \approx \tilde{u}$, the construction of C can make use of the fact that $u, \tilde{u} \in \overline{\mathcal{EC}}_G(\psi)$ to ensure that a path permitted by C visits the same equivalence classes as a path permitted by $\tilde{C}$. Formally, ψ is chosen such that its target set is the *superblock reached* by $\tilde{C}$ after $\tilde{\rho} = \tilde{\rho}'\tilde{u}$, defined by

$$T\langle \tilde{C}, \tilde{\rho}'\tilde{u} \rangle = \bigcup \left\{ [\tilde{v}]_{\mathcal{R}} \mid \tilde{\rho}'\tilde{u}\tilde{\tau}\tilde{v} \in \mathrm{Frags}^*(\tilde{C}, G) \text{ where } \tilde{\tau} \in [\tilde{u}]^*_{\mathcal{R}} \text{ and } (\tilde{u}, \tilde{v}) \notin \mathcal{R} \right\}. \tag{8}$$

The formula ψ is chosen to be the *$\mathcal{R}$-superblock step formula enforced* by $\tilde{C}$ after $\tilde{\rho} = \tilde{\rho}'\tilde{u}$, defined as

$$\Psi\langle \tilde{C}, \tilde{\rho}'\tilde{u} \rangle \equiv \begin{cases} [\tilde{u}]_{\mathcal{R}}\ \mathcal{W}\ T\langle \tilde{C}, \tilde{\rho}'\tilde{u} \rangle, & \text{if } \mathrm{Frags}^\omega(\tilde{C}, G) \cap \tilde{\rho}[\tilde{u}]^\omega_{\mathcal{R}} \neq \emptyset\ ; \\ [\tilde{u}]_{\mathcal{R}}\ \mathcal{U}\ T\langle \tilde{C}, \tilde{\rho}'\tilde{u} \rangle, & \text{otherwise}\ . \end{cases} \tag{9}$$

The superblock reached by $\tilde{C}$ after $\tilde{\rho}$ contains exactly the states in equivalence classes that can be entered directly after the equivalence class of $\tilde{u}$. The $\mathcal{R}$-superblock step formula enforced by $\tilde{C}$ after $\tilde{\rho}$ is a condition in the form of a weak or strong until formula that describes how the controller behaves within the equivalence class of $\tilde{u}$ when reached by $\tilde{\rho}$. If $\tilde{C}$ permits some path fragment that remains in the equivalence class of $\tilde{u}$ forever, then the enforced $\mathcal{R}$-superblock step formula is a weak until formula. Otherwise all permitted path fragments eventually leave the equivalence class of $\tilde{u}$, and the enforced formula is instead a strong until formula. In both cases, the source set is the equivalence class of the current state $\tilde{u}$, and the target set is the superblock reached after $\tilde{\rho}$, which contains the states that can be entered after the equivalence class of $\tilde{u}$. The following lemma shows that $\Psi\langle \tilde{C}, \tilde{\rho}'\tilde{u} \rangle$ is enforceable from $\tilde{u}$.

Lemma 4. *Let $\tilde{\rho} = \tilde{\rho}'\tilde{u} \in \mathrm{Frags}^*(\tilde{C}, G)$ with $\tilde{\rho}' \in S^*$ and $\tilde{u} \in S$. Then $\tilde{u} \in \mathcal{EC}_G(\Psi\langle \tilde{C}, \tilde{\rho} \rangle)$.*

Proof. Define a controller $\tilde{C}_{\tilde{\rho}} : S^+ \to 2^\Sigma$ such that $\tilde{C}_{\tilde{\rho}}(\tilde{u}\tau) = \tilde{C}(\tilde{\rho}\tau)$ for any $\tau \in S^*$. It will be shown that $\langle \tilde{C}_{\tilde{\rho}}/G, \tilde{u} \rangle \models \Psi\langle \tilde{C}, \tilde{\rho} \rangle$. Let $\pi = \tilde{u}u_0u_1 \cdots \in \mathrm{Frags}^\omega(\tilde{C}_{\tilde{\rho}}, G)$, and note that $\tilde{\rho}'\pi \in \mathrm{Frags}^\omega(\tilde{C}, G)$. Either all the states u_i are equivalent to $\tilde{u}$ or not. If they are, then $\tilde{\rho}'\pi \in \mathrm{Frags}^\omega(\tilde{C}, G)$ with $\pi \in [\tilde{u}]^\omega_{\mathcal{R}}$, and $\pi \models [\tilde{u}]_{\mathcal{R}}\ \mathcal{W}\ T\langle \tilde{C}, \tilde{\rho} \rangle \equiv \Psi\langle \tilde{C}, \tilde{\rho} \rangle$ by (9).

If not all u_i are equivalent to $\tilde{u}$, then π can be written as $\pi = \tilde{u}u_0u_1 \cdots u_m v\pi'$ where $(\tilde{u}, u_i) \in \mathcal{R}$ and $(\tilde{u}, v) \notin \mathcal{R}$. Then $\tilde{\rho}'\tilde{u}u_0 \cdots u_m v \in \mathrm{Frags}^*(\tilde{C}, G)$, and so $v \in T\langle \tilde{C}, \tilde{\rho} \rangle$ by (8). As $\tilde{u}u_0 \cdots u_m \in [\tilde{u}]_{\mathcal{R}}$ it is clear that $\pi \models [\tilde{u}]_{\mathcal{R}}\ \mathcal{U}\ T\langle \tilde{C}, \tilde{\rho} \rangle$ and then $\pi \models \Psi\langle \tilde{C}, \tilde{\rho} \rangle$ independently of which case of (9) applies.

Combining the two cases, $\pi \models \Psi\langle \tilde{C}, \tilde{\rho} \rangle$ for every $\pi \in \mathrm{Frags}^\omega(\tilde{C}_{\tilde{\rho}}, G)$ with $\tilde{u} \sqsubseteq \pi$. Therefore $\langle \tilde{C}_{\tilde{\rho}}/G, \tilde{u} \rangle \models \Psi\langle \tilde{C}, \tilde{\rho} \rangle$, which implies $\tilde{u} \in \mathcal{EC}_G(\Psi\langle \tilde{C}, \tilde{\rho} \rangle)$. □

In other words, there exists a (general) controller enforcing ψ from every state equivalent to the end state $\tilde{u}$ of $\tilde{\rho}$. As ψ is an $\mathcal{R}$-superblock step formula, there exists also a positional controller enforcing ψ from $[\tilde{u}]_{\mathcal{R}}$. By choosing one such positional controller and using it while staying in the equivalence class of $\tilde{u}$, the control decision of the concrete controller C can finally be defined. First let $\underline{\psi} \equiv P\ \mathcal{V}\ T$ be a stutter step formula for transition system G. Let $\overline{C}\langle \psi \rangle \in \overline{\mathcal{C}}$ be such that $\langle \overline{C}\langle \psi \rangle / G, u \rangle \models \underline{\psi}$ for all $u \in P$, if such controller exists. Then the concrete controller can be constructed as follows:

Definition 4. Let $\mathcal{R}$ be a RSBS on $G \in \mathcal{T}$, $\tilde{C}$ a controller for G, and M and Ψ defined as above. The *concrete controller* $C_C : S^+ \to 2^\Sigma$ for G based on $\tilde{C}$ is then

$$C_C(\rho' u) = \begin{cases} \overline{C}\langle \Psi\langle \tilde{C}, M(\rho' u) \rangle \rangle(u), & \text{if } M(\rho' u) \text{ is defined}\ ; \\ \Sigma, & \text{otherwise}\ . \end{cases}$$

Theorem 5 confirms, using Definition 4, that the same $\mathrm{LTL}_{\setminus\circ}$ formulas can be enforced from robust stutter bisimilar states. Corollary 6 lifts this result to robust stutter bisimilar transition systems, showing the preservation of $\mathrm{LTL}_{\setminus\circ}$ synthesis results under RSBS.

Theorem 5. *Let $\mathcal{R}$ be a RSBS on $G \in \mathcal{T}$, and φ an $\mathrm{LTL}_{\setminus\circ}$ formula, then:*

$$\forall (\tilde{s}, s) \in \mathcal{R}.\, \tilde{s} \in \mathcal{EC}_G(\varphi) \Rightarrow s \in \mathcal{EC}_G(\varphi).$$

Proof. Let $(\tilde{s}, s) \in \mathcal{R}$ such that $\tilde{s} \in \mathcal{EC}_G(\varphi)$, and let $\tilde{C}$ be such a controller enforcing φ from $\tilde{s}$. To show $s \in \mathcal{EC}_G(\varphi)$, it is shown that C_C in Definition 4 enforces φ from s, i.e., $\langle C_C/G, s \rangle \models \varphi$.

The proof hinges on showing that $M(\rho)$ is always defined for path fragments ρ permitted by C_C with $s \sqsubseteq \rho$, and that $\overline{C}\langle \Psi\langle \tilde{C}, M(\rho) \rangle \rangle$ always exists. It is also shown that $M(\rho) \in \mathrm{Frags}^*(\tilde{C}, G)$, and that ρ and $M(\rho)$ are $\mathcal{R}$-stutter equivalent. This latter property is used to show that C_C and $\tilde{C}$ enforce the same $\mathrm{LTL}_{\setminus\circ}$ formulas.

By induction over ρ, it is shown that $M(\rho)$ is defined, $M(\rho) \in \mathrm{Frags}^*(\tilde{C}, G)$, and ρ and $M(\rho)$ are $\mathcal{R}$-stutter equivalent. The base case follows by construction of M.

For the inductive step, consider $\rho = \rho' uv$ with $\rho' \in S^*$ and $u, v \in S$. The inductive hypothesis is that $M(\rho' u)$ is defined, that $M(\rho' u) = \tilde{\rho}'\tilde{u} \in \mathrm{Frags}^*(\tilde{C}, G)$, and that $\rho' u$ and $\tilde{\rho}'\tilde{u}$ are $\mathcal{R}$-stutter equivalent. There are two cases depending on whether $(u, v) \in \mathcal{R}$ or not. In the first case, $(u, v) \in \mathcal{R}$, the inductive step follows from the inductive hypothesis and the construction of M.

In the second case, $(u, v) \notin \mathcal{R}$, it must be shown that $F(M(\rho' u), v)$ in (7) is nonempty. Let $\psi \equiv \Psi\langle \tilde{C}, \tilde{\rho}'\tilde{u} \rangle$. As $\tilde{\rho}'\tilde{u} \in \mathrm{Frags}^*(\tilde{C}, G)$, it follows by Lemma 4 and Proposition 2 that $\tilde{u} \in \mathcal{EC}_G(\psi) = \overline{\mathcal{EC}}_G(\psi)$. Since $\mathcal{R}$ is an RSBS and $(\tilde{u}, u) \in \mathcal{R}$, it follows that $u \in \overline{\mathcal{EC}}_G(\psi)$. Thus, there exists a controller $\overline{C}\langle \psi \rangle$ enforcing ψ from u. By Definition 4 it follows that $C_C(\rho' u) = \overline{C}\langle \psi \rangle(u)$. Together with $\rho = \rho' uv \in \mathrm{Frags}^*(C_C, G)$, this implies that $uv \in$

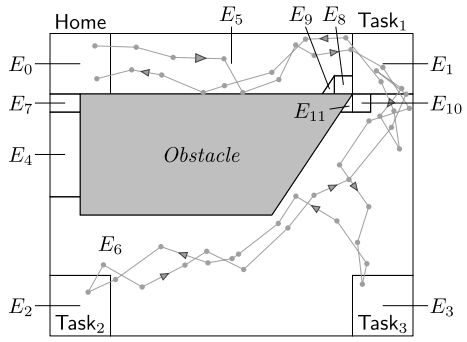

Fig. 4. The equivalence classes of the coarsest robust stutter bisimulation for the robot navigation example.

$\text{Frags}^*(\overline{C}\langle\psi\rangle, G)$, and then $v \in T\langle\tilde{C}, \tilde{\rho}'\tilde{u}\rangle$ by (9). By (8) there exists $\tilde{\rho} = \tilde{\rho}'\tilde{u}\tilde{\tau}\tilde{v} \in \text{Frags}(\tilde{C}, G)$ with $\tilde{\tau} \in [\tilde{u}]^*_{\mathcal{R}}$, and then $\tilde{\rho} \in F$ by (4), so $F(M(\rho'u), v)$ is nonempty. Also, $M(\rho) \in F(\tilde{\rho}'\tilde{u}, v) \subseteq \text{Frags}^*(\tilde{C}, G)$ by (4) and (7). Lastly, any $M(\rho) \in F(\tilde{\rho}'\tilde{u}, v)$ is $\mathcal{R}$-stutter equivalent to $\tilde{\rho}'\tilde{u}\tilde{v}$, so as $(\tilde{v}, v) \in \mathcal{R}$, it follows that $\rho'uv$ and $\tilde{\rho}'\tilde{u}\tilde{v}$ and $M(\rho)$ are $\mathcal{R}$-stutter equivalent. This concludes the induction.

To show that $\langle C_C/G, s\rangle \models \varphi$, by Theorem 1 it suffices to show that each infinite path fragment $\pi \in \text{Frags}^\omega(C_C, G)$ with $s \sqsubseteq \pi$ is stutter equivalent to some infinite path fragment $\tilde{\pi} \in \text{Frags}^\omega(\tilde{C}, G)$ with $\tilde{s} \sqsubseteq \tilde{\pi}$. Consider two cases.

In the first case, π can be written as $\pi = \rho'u[u]^\omega_{\mathcal{R}}$. Let $M(\rho'u) = \tilde{\rho}'\tilde{u}$, with $\tilde{\rho}' \in S^*$ and $\tilde{u} \in S$. By Definition 4, $C_C(\rho'u) = \overline{C}\langle\Psi\langle\tilde{C}, M(\rho'u)\rangle\rangle(u)$, which means that $\Psi\langle\tilde{C}, M(\rho'u)\rangle$ must be of the form $P \mathcal{W} T$, which by (9) means that $\tilde{\pi} = \tilde{\rho}'[\tilde{u}]^\omega_{\mathcal{R}} \in \text{Frags}(\tilde{C}, G)$. As $\rho'u$ and $\tilde{\rho}'\tilde{u}$ are $\mathcal{R}$-stutter equivalent, so are π and $\tilde{\pi}$.

In the second case, π does not stay in one equivalence class forever. Let $\tilde{\pi}$ be such that for all finite prefixes $\rho \sqsubset \pi$ with $s \sqsubseteq \rho$, it holds that $M(\rho) \sqsubset \tilde{\pi}$. Since M is a prefix preserving map, $\tilde{\pi}$ is well defined and unique. Furthermore, it holds that $\rho \in \text{Frags}^*(C_C, G)$, and thus $M(\rho) \in \text{Frags}^*(\tilde{C}, G)$. Because M is prefix preserving, it follows that $\tilde{\pi} \in \text{Frags}^\omega(\tilde{C}, G)$, and as ρ and $M(\rho)$ are $\mathcal{R}$-stutter equivalent, so are π and $\tilde{\pi}$.

In both cases where an arbitrary infinite path fragment π is picked from $\text{Frags}^\omega(C_C, G)$, there exists an $\mathcal{R}$-stutter equivalent path fragment $\tilde{\pi} \in \text{Frags}^\omega(\tilde{C}, G)$. $\mathcal{R}$ is an RSBS, so π and $\tilde{\pi}$ are stutter equivalent. □

Corollary 6. *Let $G, \tilde{G} \in \mathcal{T}$, let $\mathcal{R}$ be a RSBS on $G \cup \tilde{G}$ and let φ be an $\text{LTL}_{\setminus\circ}$ formula. If $\exists \tilde{C} \in \mathcal{C}$ for $\tilde{G}$ such that $\tilde{C}/\tilde{G} \models \varphi$, then $\exists C \in \mathcal{C}$ for G such that $C/G \models \varphi$.*

Constructing as by Definition 4 the concrete controller C_C from a controller $\tilde{C}$ for an abstract transition system $\tilde{G}$ that is robust stutter bisimilar to a concrete transition system G solves the Controller Construction Problem.

7. Robot navigation example

This section applies the abstraction and controller construction to a problem inspired by Mohajerani et al. (2021), where a robot navigates a two-dimensional space while avoiding the *Obstacle* shown in Fig. 4. The main difference to the previous work is an area, E_4, too narrow for the robot to pass due to disturbances.

The robot movement is described by the following discrete-time linear equation with disturbances

$$x(k+1) = x(k) + u(k) + w(k)\,, \tag{10}$$

where the state is $x \in X = ([0, 6] \times [0, 5]) \setminus \textit{Obstacle}$, the control input is $u \in U = [-0.6, 0.6]^2$, and the disturbance is $w \in W = [-0.3, 0.3]^2$. From the dynamics described by (10) a transition system is constructed as $G = \langle X, U, \delta, X^\circ, \text{AP}, L\rangle$ (Tabuada, 2009); if for $u(k) \in U$, there exists $w(k) \in W$ such that $x_1 = x_0 + u + w$ then there is a transition $(x_0, u, x_1) \in \delta$. The set of initial states S° is the top left corner, $E_0 = [0, 1] \times [4, 5]$. The set of propositions is $\text{AP} = \{\text{Home}, \text{Task}_1, \text{Task}_2, \text{Task}_3\}$, and the state labelling function L assigns these to the corner sets, see Fig. 4. The $\text{LTL}_{\setminus\circ}$ formula to enforce is:

$$\varphi = \square\lozenge\text{Home} \wedge \square\lozenge\text{Task}_1 \wedge \square\lozenge\text{Task}_2 \wedge \square\lozenge\text{Task}_3\,,$$

meaning that the robot should visit each of the four corner regions infinitely often.

As shown above, the Abstraction Problem can be solved by constructing a quotient with respect to a robust stutter bisimulation $\mathcal{R}$. While such a relation $\mathcal{R}$ always exists (e.g., the identity relation), it is desirable to obtain the coarsest robust stutter bisimulation on a given transition system, for which the quotient has the smallest number of states possible. This can be achieved by *partition refinement* (Paige & Tarjan, 1987), where an initial partition is repeatedly modified by splitting equivalence classes until a robust stutter bisimulation is found. Details can be found in Krook et al. (2022).

Such a partition refinement applied to G results in a robust stutter bisimulation $\mathcal{R}$ consisting of the twelve equivalence classes $E_0, \ldots, E_{11}$, see Fig. 4. It is possible to keep the robot indefinitely in E_0, E_1, E_2, and E_3, since the disturbance cannot push the robot out of these regions from their centre and the control input has higher magnitude than any possible disturbance. For example $E_0 \mathcal{W} \emptyset$ is enforceable from all states in E_0, i.e., $E_0 \subseteq \overline{\mathcal{E}\mathcal{C}}_G(E_0 \mathcal{W} \emptyset)$ and thus $E_0 \mathcal{W} \emptyset \in \Sigma_{\mathcal{R}}$.

The region between the corners is split into eight equivalence classes. E_4 is split off from the other regions because it is so narrow that, despite the control input, the disturbance may push the robot into the *Obstacle* from anywhere within E_4, and consequently there is no $\mathcal{R}$-superblock step formula from E_4 in $\Sigma_{\mathcal{R}}$. From all other regions some enforceable $\mathcal{R}$-superblock step formula exists. The two equivalence classes E_5 and E_6 are dissimilar since the robot can be forced from E_5 to E_0 without visiting another region ($E_5 \mathcal{U} E_0 \in \Sigma_{\mathcal{R}}$), but this cannot be enforced from E_6 ($E_6 \mathcal{U} E_0 \notin \Sigma_{\mathcal{R}}$). The equivalence classes E_7, E_8, E_9, E_{10}, and E_{11} are so small in relation to the disturbance that the controller cannot force the robot to enter or stay in these classes. E_7 is different from E_5, since from E_7 the robot can only be forced into E_0, not into E_1, so $E_7 \mathcal{U} E_1 \notin \Sigma_{\mathcal{R}}$ but $E_5 \mathcal{U} E_1 \in \Sigma_{\mathcal{R}}$. The equivalence classes E_8 and E_9 are split off from E_5 since the robot can be forced directly into the superblock $E_6 \cup E_{10} \cup E_{11}$ from $E_8 \cup E_9$ but not from E_5. Similar reasons cause $E_{10} \cup E_{11}$ to be split off from E_6. Then E_8 and E_9 are split since the robot can be forced to E_1 from E_8 but not from E_9. The split of E_8 and E_9 propagates, causing a separation of E_{10} and E_{11}, for example $E_{11} \mathcal{U} (E_5 \cup E_9) \in \Sigma_{\mathcal{R}}$ and $E_{10} \mathcal{U} (E_5 \cup E_8 \cup E_9) \in \Sigma_{\mathcal{R}}$.

From all the superblock step formulas enforceable from the twelve regions, an abstract transition system $\tilde{G} = G/\mathcal{R}$ is constructed as by Definition 3. An abstract controller $\tilde{C}$ is synthesised for $\tilde{G}$ to enforce φ, and from $\tilde{C}$ is then constructed a concrete controller C_C according to Definition 4. Fig. 4 shows a path fragment where the robot completes the three tasks and returns to the home region despite the disturbance causing somewhat erratic behaviour.

8. Conclusions

This paper proposes a method to synthesise controllers that enforce requirements specified in $\text{LTL}_{\setminus\circ}$ for cyber-physical systems subject to disturbances. This is done by constructing a

finite-state abstraction of the system and then synthesising a controller for this abstraction. The *robust stutter bisimulation* relation is shown to characterise the relevant abstraction accurately, and it is shown how this relation can be used to construct a concrete controller. Though the main focus is to handle process noise, robust stutter bisimulation is general and can be used on any system representable as transition systems.

Krook et al. (2022) outline a simple algorithm for computing the coarsest RSBS for a transition system, and a more efficient algorithm for computing the coarsest RSBS is future work.

This paper considers disturbances added to the state transitions, but for some applications there is considerable measurement noise that results in unobservable transitions. Future work might investigate how such disturbances can be incorporated in the abstraction.

References

Alur, Rajeev, Henzinger, Thomas A., Kupferman, Orna, & Vardi, Moshe Y. (1998). Alternating refinement relations. In Davide Sangiorgi, & Robert de Simone (Eds.), *LNCS*: *1466*, *9th int. conf. concurrency theory* (pp. 163–178). Springer, http://dx.doi.org/10.1007/BFb0055622.

Alur, Rajeev, Henzinger, Thomas A., Lafferriere, Gerardo, & Pappas, George J. (2000). Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, *88*(7), 971–984. http://dx.doi.org/10.1109/5.871304.

Baier, Christel, & Katoen, Joost-Pieter (2008). *Principles of model checking*. MIT Press.

Belta, Calin, Yordanov, Boyan, & Gol, Ebru Aydin (2017). *Formal methods for discrete-time dynamical systems* (1st ed.). Springer.

Girard, Antoine, & Pappas, George J. (2007). Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, *52*(5), 782–798. http://dx.doi.org/10.1109/TAC.2007.895849.

Kloetzer, Marius, & Belta, Calin (2008). A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, *53*(1), 287–297. http://dx.doi.org/10.1109/TAC.2007.914952.

Krook, Jonas, Malik, Robi, Mohajerani, Sahar, & Fabian, Martin (2022). Robust stutter bisimulation for abstraction and controller synthesis with disturbance: Proofs. arXiv e-prints, arXiv:2205.13959 [eess.SY].

Lee, Edward A. (2015). The past, present and future of cyber-physical systems: A focus on models. *Sensors*, *15*, 4837–4869. http://dx.doi.org/10.3390/s150304837.

Liu, Jun, & Ozay, Necmiye (2016). Finite abstractions with robustness margins for temporal logic-based control synthesis. *Nonlinear Analysis. Hybrid Systems*, *22*, 1–15. http://dx.doi.org/10.1016/j.nahs.2016.02.002.

Milner, Robin (1989). *Series in computer science, Communication and concurrency*. Prentice-Hall.

Mohajerani, Sahar, Malik, Robi, Wintenberg, Andrew, Lafortune, Stéphane, & Ozay, Necmiye (2021). Divergent stutter bisimulation abstraction for controller synthesis with linear temporal logic specifications. *Automatica*, *130*, Article 109723. http://dx.doi.org/10.1016/j.automatica.2021.109723.

Nilsson, Petter, Ozay, Necmiye, & Liu, Jun (2017). Augmented finite transition systems as abstractions for control synthesis. *Discrete Event Dynamic Systems*, *27*, 301–340. http://dx.doi.org/10.1007/s10626-017-0243-z.

Nilsson, Petter, Özay, Necmiye, Topcu, Ufuk, & Murray, Richard M. (2012). Temporal logic control of switched affine systems with an application in fuel balancing. In *2012 American control conf.* (pp. 5302–5309). http://dx.doi.org/10.1109/ACC.2012.6315141.

Paige, Robert, & Tarjan, Robert E. (1987). Three partition refinement algorithms. *SIAM Journal on Computing*, *16*(6), 973–989. http://dx.doi.org/10.1137/0216062.

Pola, Giordano, & Tabuada, Paulo (2009). Symbolic models for nonlinear control systems: Alternating approximate bisimulations. *SIAM Journal on Control and Optimization*, *48*(2), 719–733. http://dx.doi.org/10.1137/070698580.

Ramadge, P. J. G. (1989). Some tractable supervisory control problems for discrete-event systems modeled by Buchi automata. *IEEE Transactions on Automatic Control*, *34*(1), 10–19. http://dx.doi.org/10.1109/9.8645.

Reissig, Gunther, Weber, Alexander, & Rungger, Matthias (2016). Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, *62*(4), 1781–1796. http://dx.doi.org/10.1109/TAC.2016.2593947.

Tabuada, P. (2006). Symbolic control of linear systems based on symbolic subsystems. *IEEE Transactions on Automatic Control*, *51*(6), 1003–1013. http://dx.doi.org/10.1109/TAC.2006.876946.

Tabuada, Paulo (2009). *Verification and control of hybrid systems: A symbolic approach*. Springer Science & Business Media.

Wagenmaker, Andrew J., & Ozay, Necmiye (2016). A bisimulation-like algorithm for abstracting control systems. In *54th allerton conf. communication, control and computing* (pp. 569–576). http://dx.doi.org/10.1109/ALLERTON.2016.7852282.

Zamani, Majid, Pola, Giordano, Mazo, Manuel, & Tabuada, Paulo (2011). Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control*, *57*(7), 1804–1809. http://dx.doi.org/10.1109/TAC.2011.2176409.

Jonas Krook received his M.Sc. in Complex Adaptive Systems and his Ph.D. in Electrical Engineering from Chalmers University of Technology in 2013 and 2022, respectively. He worked at Volvo Cars Corporation for five years, where he developed Advanced Driver Assistance Systems. At Volvo he received the Volvo Cars Technology Award together with his team, for developing Auto Brake in Intersections. Since 2022, he is working at Zenseact developing Advanced Driver Assistance Systems. His research interest is in safe decision making for vehicles.

Robi Malik received the M.S. and Ph.D. degree in computer science from the University of Kaiserslautern, Germany, in 1993 and 1997, respectively. From 1998 to 2002, he worked at Siemens Corporate Research in Munich, Germany, where he was involved in the development and application of modelling and analysis software for discrete event systems. Since 2003, he is lecturing Computer Science at the University of Waikato in Hamilton, New Zealand. He is participating in the development of the Supremica software for modelling and analysis of discrete event systems. His current research interests are in the area of model checking and synthesis of large discrete event systems and other finite-state machine models.

Sahar Mohajerani received her M.S. degree in Systems, Control, and Mechatronics from and her Ph.D. in Automation from Chalmers University of Technology, in 2009 and 2015, respectively. She worked at Volvo Cars Corporation on function verification for two years, until 2017. She was a post-doctoral fellow at the University of Michigan from 2017 to 2019, where she worked on security and privacy verification of cyber-physical systems. She received the Best Student Paper Award at IFAC-IEEE International Workshop on Discrete Event Systems in 2014 (for a paper co-authored with R. Malik and M. Fabian). She is currently working as senior software engineer at Mathworks. Her research interests include formal methods for verification and synthesis of large discrete event systems and cyber-physical systems.

Martin Fabian received his Ph.D. degree in Control Engineering in 1995 from Chalmers University of Technology, Gothenburg, Sweden. He is currently Full Professor with the Department of Electrical Engineering, Chalmers University of Technology, and head of the Automation Research group. His research interests involve modelling and supervisory control of discrete event systems, modular and compositional methods for complex systems, and verification of autonomous systems and state-based smart contracts. He is co-developer of the Supremica software for modelling and analysis of discrete event systems. He is also a member of the WASP Faculty.