A Simpler Alternative
Minimizing Transition Systems Modulo Alternating Simulation Equivalence

Gleizer, Gabriel De Albuquerque; Madnani, Khushraj; Mazo, Manuel

DOI
10.1145/3501710.3519534

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 25th ACM International Conference on Hybrid Systems (HSCC 2022)

A Simpler Alternative: Minimizing Transition Systems Modulo Alternating Simulation Equivalence

Gabriel de A. Gleizer*
g.gleizer@tudelft.nl
TU Delft
Delft, The Netherlands

Khushraj Madnani*
K.N.Madnani-1@tudelft.nl
TU Delft
Delft, The Netherlands

Manuel Mazo Jr.
m.mazo@tudelft.nl
TU Delft
Delft, The Netherlands

ABSTRACT

This paper studies the reduction (abstraction) of finite-state transition systems for control synthesis problems. We revisit the notion of alternating simulation equivalence (ASE), a more relaxed condition than alternating bisimulations, to relate systems and their abstractions. As with alternating bisimulations, ASE preserves the property that the existence of a controller for the abstraction is necessary and sufficient for a controller to exist for the original system. Moreover, being a less stringent condition, ASE can reduce systems further to produce smaller abstractions. We provide an algorithm that produces minimal AS equivalent abstractions. The theoretical results are then applied to obtain (un)schedulability certificates of periodic event-triggered control systems sharing a communication channel. A numerical example illustrates the results.

CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Theory of computation** → *Abstraction*; • **Networks** → *Cyber-physical networks.*

KEYWORDS

Alternating Simulation, Minimization, Controller Synthesis, Event Triggered Control, Scheduling.

ACM Reference Format:
Gabriel de A. Gleizer, Khushraj Madnani, and Manuel Mazo Jr.. 2022. A Simpler Alternative: Minimizing Transition Systems Modulo Alternating Simulation Equivalence. In *25th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '22), May 4–6, 2022, Milan, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3501710.3519534

1 INTRODUCTION

Control synthesis for finite transition systems (FTS), the problem of finding a controller (a strategy) that enforces specifications on a closed-loop system, is a long investigated problem [29]. Supervisory control, as it is often also referred to, has many applications in e.g. automation of manufacturing plants, traffic control, scheduling and planning, and control of dynamical and hybrid systems [8, 32]. The clearest advantage of using finite transition systems to model a control problem is that a large class of control problems in finite transition systems are *decidable*, meaning that the controller can be obtained automatically through an algorithm, or that a definitive answer that no controller can enforce the specifications is obtained. The disadvantage is often a very practical one: the problem may be too large to be solved in practice, owing to the large number of states and transitions the control problem may have. In particular, this is the case of scheduling the transmissions of event-triggered control (ETC) systems in a shared network [27], whose traffic models can be abstracted as FTSs [12]: often it is not possible to synthesize schedulers for more than a handful of ETC systems, due to the state explosion of the composed system. This state-space explosion problem is pervasive, and thus significant attention has been devoted to reducing transition systems. The reduction requires a formal relation between original and reduced system; for verification purposes, the most well-known relation is that of *simulation* [5, 28]. Algorithms to reduce systems modulo simulations soon emerged: the first being a reduction modulo *bisimulation*, where algorithms using *quotient systems* are often used [5]; later, minimization modulo *simulation equivalence* was devised in [7]. Simulation equivalence is a weaker relation than bisimulation but allows to verify most of the same properties; in particular, any linear temporal logic (LTL) property that can be verified on a system also holds for a simulation equivalent system.[1]

For control synthesis, reducing the system using mere simulation notions is not enough. Control synthesis can be seen as a game over a finite alphabet, where the controller plays against an antagonistic environment, and simulations preserve all possible moves from both players, *including moves that are irrational* for the game. The notion that appropriately captures the game aspect of control synthesis problems is that of *alternating simulation,* introduced for multi-agent systems by Alur et al. in [2]. Surprisingly, though, there has been little investigation of the problem of reducing systems modulo *alternating bisimulations* or *alternating simulation equivalence.* Reducing systems using alternating simulation notions has many practical benefits: not only the synthesis problems become smaller, and by extension the obtained controllers, making them easier to implement in limited hardware; but it becomes even more important, we argue, when solving control synthesis problems on a parallel composition of systems, one classic example being scheduling. In this case, the size of the game grows exponentially with

*Both authors contributed equally to this research.

HSCC '22, May 4–6, 2022, Milan, Italy

ACM ISBN 978-1-4503-9196-2/22/05.
https://doi.org/10.1145/3501710.3519534

[1] Larger classes of logic properties can be verified, such ACTL*, ECTL*, ECTL, ACTL as its sublogics, see [7]. For control, we are typically interested in LTL specifications.

the number of systems to be scheduled, hence any reduction on the individual systems results in an exponential reduction of the size of the composed game.

In this work we present a novel algorithm to reduce systems w.r.t. alternating-simulation equivalence (ASE), a different and relaxed notion than the more popular relative *alternating-bisimulation relation*. ASE is nonetheless stronger than alternating simulation relations, as it guarantees not only that controllers can be transferred from abstraction (the reduced system) to concrete (the original system), but also that non-existence of a controller in the abstraction implies non-existence of a controller for the concrete system. Hence the reduction via ASE is sound and complete for control synthesis. We prove that our algorithm in fact obtains a minimal system that is alternating-simulation equivalent to the original. The algorithm is composed of five steps: (i) computing the maximal alternating simulation relation from the system to itself; (ii) forming the quotient system; (iii) eliminating irrational and/or redundant actions from the controller; (iv) eliminating irrational transitions from the environment; and (v) deleting states which are inaccessible from any of the initial states. The complexity of the algorithm is $O(m^2)$, where m is the number of transitions in the system to be reduced. This result is a very interesting theoretical contribution on its own right, generalizing the results in [7]. Because these simulation relations are closed under composition, the presented algorithm has a strong practical relevance for synthesis over composed systems. We demonstrate these benefits on a case study — one which in fact motivated the investigation of our problem: scheduling of multiple periodic event-triggered control (PETC) [4, 18, 31] systems on a shared channel. The insights from our algorithm allow to prove that, under some conditions, ETC and self-triggered control (STC, [3, 26, 33]) are equally schedulable. Additionally, we use our algorithm on a numerical case study, obtaining in the best case a system 50x smaller than the original one. This resulted in a reduction in CPU time of the scheduling problem of several orders of magnitude in some cases. Furthermore, the reduced systems also provide important insights to the user, as the reduced system indicates somehow the bottlenecks that must be addressed to improve schedulability.

1.1 Related Work

Algorithms for reducing state space preserving bisimulation using quotient systems have been extensively studied [22, 24], see [5, 6] for an overview. For many practical results, simulation equivalence, a coarser equivalence relation, is preferable. Various algorithms to obtain quotients based on simulation equivalence have been proposed,e.g., [19, 30], as well as their associated quotients [10]. However, unlike bisimulation, creating quotients based on simulation equivalence does not result in minimization [7]. Our algorithm and results are akin to those of [7]; we have here a generalization of its results, as alternating simulation reduces to simulations if one of the players has only one choice in every state.

The reduction of systems using alternating simulation equivalence has been addressed in [21, 25]. Different from the current work, Majumdar et al. propose a semi-algorithm that aims at reducing infinite systems into finite systems (not necessarily minimal); instead, here we want to minimize finite systems by reducing the number of states and transitions. These two approaches are complimentary and can be used in combination to obtain minimal finite realizations of certain classes of infinite systems (namely, class 2 systems as per [25]).

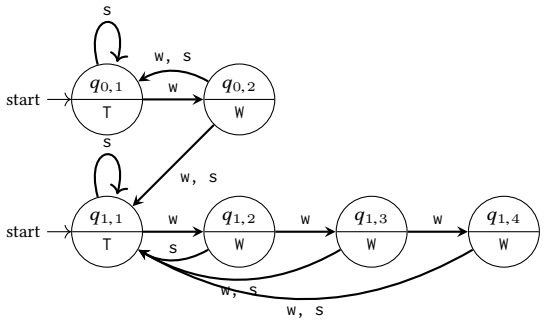


Figure 1: A finite LTS representing a PETC traffic model with scheduler actions. Node labels are states (top) and their outputs (bottom), and edge labels are actions.

Reduction of other types of finite transition systems has been addressed, as in, e.g., [16] for alternating Büchi automata modulo different notions of simulations, namely direct, fair, and delayed simulations. Although such automata also represent games, they are defined differently than what is usual for control: an alternating Büchi automaton accepts a word if the controller can ensure it by playing against the environment; every such word forms the language of the automaton, and simulations must preserve this language in some sense. This is fundamentally different than most control problems, where one is not interested in specific words, but rather that the set of all words generated by the system satisfies some specifications. In addition, [16] does not contain results on minimality.

1.2 Notation

We denote by $\mathbb{N}_0$ the set of natural numbers including zero, $\mathbb{N} := \mathbb{N}_0 \setminus \{0\}$, $\mathbb{N}_{\leq n} := \{1, 2, ..., n\}$. For a relation $R \subseteq X_a \times X_b$, its inverse is denoted as $R^{-1} = \{(x_b, x_a) \in X_b \times X_a \mid (x_a, x_b) \in R\}$. Every function $F : X_a \mapsto X_b$ can be read as a relation, namely $\{(x_a, x_b) \in X_a \times X_b \mid x_b = F(x_a)\}$.

2 PRELIMINARIES

2.1 Labelled Transition Systems

A (finite) LTS is a 6-tuple $\mathcal{S} := (X, X_0, U, Y, \delta, H)$, where X is a (finite) set of states, $X_0 \subseteq X$ is the set of initial states, U is the (finite) set of edge labels called inputs or actions, Y is the set of outputs or observations, $\delta \subset X \times U \times X$ is the set of transitions and $H : X \mapsto Y$, the output map, maps states to their corresponding outputs. Figure 1 shows one example of a finite LTS, which is our running example throughout this paper; its meaning is going to be explained in Section 4.

The *size* of an LTS, denoted by $|\mathcal{S}|$, is the triplet $(|X|, |X_0|, |\delta|)$. This induces a partial order amongst systems sizes using the natural extension of $\leq$ on numbers, i.e., $|(X, X_0, U, Y, \delta, H)| \leq |(X', X_0', U', Y', \delta', H')|$ iff $|X| <= |X'|$, $|X_0| \leq |X_0'|$, and $|\delta| \leq |\delta'|$.

For any $u \in U$ and $x, x' \in X$, We use $x \xrightarrow{u} x'$ to denote the fact that $(x, u, x') \in \delta$. We denote by $U(x) := \{u \in U \mid \exists x' \in X, x \xrightarrow{u} x'\}$ the set of input labels available at state x, $\mathrm{Post}(x, u) := \{x' \in X \mid x \xrightarrow{u} x'\}$ the set of u-successors of x and $\mathrm{Pre}(x, u) := \{x' \in X \mid x' \xrightarrow{u} x\}$. When the system $\mathcal{S}$ is not clear from context, we use, respectively, $x \xrightarrow[\mathcal{S}]{u} x'$, $\mathrm{Post}^{\mathcal{S}}(x, u)$, and $U^{\mathcal{S}}(x)$. System $\mathcal{S}$ is said to be deterministic if for every $x \in X$ and $u \in U(x)$, we have $|\mathrm{Post}(x, u)| = 1$. For a state $x \in X$, we denote by $\mathcal{S}(x) := \{X, x, U, Y, \delta, H\}$ the system $\mathcal{S}$ initialized at x.

Finite LTSs represent dynamical systems that evolve in discrete state spaces upon the occurrence of actions or events in U. They can represent computer programs, machines or factories, but also infinite dynamical systems through the method of abstractions, see [32]. The problem of control design in finite LTSs is to design a *controller* or *strategy* that chooses the action in U at any point of the run r of the system such that a given specification ϕ is satisfied. This has a game aspect in that the controller must ensure ϕ no matter what the *environment* does; hence, one can see the environment, i.e., the entity that picks transitions in δ given the outbound state x and the control action u, as antagonist to the controller objectives. The specification ϕ is typically given in terms of linear temporal logic (LTL), from which two popular particular cases are safety and reachability. In our scheduling case study (§4), we have a safety problem, which is to avoid collisions during transmissions over a shared communication channel.

2.2 Alternating Simulation and Equivalence

The concept of alternating simulations was first proposed by [2] for multi-player games on structures called alternating transition systems. It was later simplified by Tabuada for a two-player game, where the *controller* chooses actions in U to meet some specification against an antagonist *environment* that chooses the transitions. The following definition is an adaptation of Tabuada's [32]:

Definition 2.1 (Alternating simulation (AS)). Consider two systems $\mathcal{S}_a := (X_a, X_{a0}, U_a, Y_a, \delta_a, H_a)$ and $\mathcal{S}_b = (X_b, X_{b0}, U_b, Y_b, \delta_b, , H_b)$. We say that $\mathcal{S}_b$ is an *alternating simulation* of $\mathcal{S}_a$, denoted by $\mathcal{S}_a \preceq_{AS} \mathcal{S}_b$, if there exists a relation $R \subseteq X_a \times X_b$ satisfying following requirements:

(i) $\forall x_{b0} \in X_{b0}\ \exists x_{a0} \in X_{a0}$ such that $(x_{a0}, x_{b0}) \in R$;
(ii) $\forall (x_a, x_b) \in R$, it holds that $H(x_a) = H(x_b)$;
(iii) $\forall (x_a, x_b) \in R, \forall u_a \in U_a(x_a)\ \exists u_b \in U_b(x_b)$ such that $\forall x'_b \in \mathrm{Post}^{\mathcal{S}_b}(x_b, u_b)$, $\exists x'_a \in \mathrm{Post}^{\mathcal{S}_a}(x_a, u_a)$ s.t. $(x'_a, x'_b) \in R$.

We call R an *alternating simulation relation (ASR)* from $\mathcal{S}_a$ to $\mathcal{S}_b$. When using a specific relation R, we use the notation $\mathcal{S}_a \preceq_R \mathcal{S}_b$.

It is easy to see that if two relations R_1 and R_2 satisfy $\mathcal{S}_a \preceq_{R_1} \mathcal{S}_b$ and $\mathcal{S}_a \preceq_{R_2} \mathcal{S}_b$, then $\mathcal{S}_a \preceq_{R_1 \cup R_2} \mathcal{S}_b$. The union of all ASRs from $\mathcal{S}_a$ to $\mathcal{S}_b$ is called the *maximal alternating simulation relation* from $\mathcal{S}_a$ to $\mathcal{S}_b$.

Intuitively, given LTS $\mathcal{S}_a$ and $\mathcal{S}_b$, an ASR from an LTS $\mathcal{S}_a$ to $\mathcal{S}_b$, implies that every controller move of $\mathcal{S}_a$ can be "replicated" by the controller of $\mathcal{S}_b$ and every environment move of $\mathcal{S}_b$ can be "replicated" by that of $\mathcal{S}_a$. Informally, this means that the controller of $\mathcal{S}_b$ is at least as powerful as that of $\mathcal{S}_a$ and the environment of $\mathcal{S}_a$ is at least as powerful as that of $\mathcal{S}_b$. This interpretation is also behind our modification of the definition w.r.t. [32], where condition (i) is reversed: in our definition, the "environment" picks the initial state, so every initial state in $\mathcal{S}_b$ must be matched in $\mathcal{S}_a$.[2]

The importance of alternating simulations for control stem from the following fact: given any temporal-logic specification ϕ over the alphabet Y, if $\mathcal{S}_a \preceq_{AS} \mathcal{S}_b$, then the existence of a controller for $\mathcal{S}_a$ such that the closed-loop system satisfies ϕ implies that there exists a controller for $\mathcal{S}_b$ meeting the same specification; in fact, the strategy for $\mathcal{S}_a$ can be refined for $\mathcal{S}_b$. Moreover, for any specification ϕ, if $(x, x') \in R$ and the controller can ensure ϕ from x, then it can ensure ϕ from x'; the symmetric notion holds: if the controller cannot ensure ϕ from x', then it cannot ensure it from x. An additional reason for the importance of AS is that it commutes with composition, making this notion suitable for control design of a composition of systems, such as the scheduling problem we tackle in §4. For a thorough exposition about these facts and how to synthesize controllers for several types of specifications we refer the reader to [32]. Here we are interested in reducing a system $\mathcal{S}_a$ preserving an *if-and-only-if* property; namely, *for any specification* there exists a controller for $\mathcal{S}_a$ iff there exists a controller for $\mathcal{S}_b$. The most known notion for this is that of alternating bisimulation:

Definition 2.2 (Alternating bisimulation). Two LTSs $\mathcal{S}_a$ and $\mathcal{S}_b$ are said to be *alternatingly bisimilar*, denoted by $\mathcal{S}_a \cong_{AS} \mathcal{S}_b$, if there is an ASR R from $\mathcal{S}_a$ to $\mathcal{S}_b$ such that its inverse R^{-1} is an ASR from $\mathcal{S}_b$ to $\mathcal{S}_a$.

A relaxed notion w.r.t. bisimulation is that of equivalence:

Definition 2.3 (Alternating simulation equivalence (ASE)). Two LTSs $\mathcal{S}_a$ and $\mathcal{S}_b$ are said to be *alternating-simulation equivalent*, denoted by $\mathcal{S}_a \simeq_{AS} \mathcal{S}_b$, if there is an ASR R from $\mathcal{S}_a$ to $\mathcal{S}_b$ and an ASR R' from $\mathcal{S}_b$ to $\mathcal{S}_a$.

ASE reduces to bisimulation when $R' = R^{-1}$; nevertheless, it preserves by definition the if-and-only-if property we are interested in. Moreover, a second relation is an extra degree of freedom to find a reduced system that is ASE to the original. There is a price to pay for this freedom: the controller designed for the reduced system will not be as *permissive* as the best controller that could be created by the original system; in other words, it may contain fewer actions available to pick from at any point in the system's run. Nonetheless, this can be regarded as a benefit, considering the sheer size the strategies for large LTSs can have.

3 MAIN RESULT

In this section, we present our main result:[3] given an LTS $\mathcal{S}$, there exists a polynomial time algorithm that constructs a minimal LTS $\mathcal{S}_{\min}$ equivalent to $\mathcal{S}$ modulo alternating simulation (AS). That is, $\mathcal{S}_{\min} \simeq_{AS} \mathcal{S}$ and $|\mathcal{S}_{\min}| \leq |\mathcal{S}'|$ for any $\mathcal{S}'$ satisfying $\mathcal{S}' \simeq_{AS} \mathcal{S}$. (i) We first provide an overview of the algorithm to obtain such a minimal system. (ii) We then provide the details of the each step of

[2] Note that Tabuada's definition and ours are not fundamentally different. In both cases, one could have a single initial state, and condition (i) of Def. 2.1 would be a consequence of condition (iii) by adding silent transitions from the initial state to the "real" initial state set; for Tabuada's definition, condition (i) would be derived by (iii) if instead the controller would have a different action for each of these transitions.

[3] When a proof is not right after the result statement, please refer to the arXiv preprint of this paper, arXiv:2203.01672.

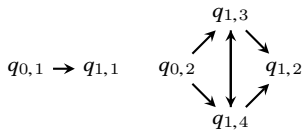


Figure 2: Maximal alternating simulation relation $R^{\max}$ for the system in Fig. 1: $q \to q'$ means that $(q, q') \in R^{\max}$. Self-loops and relations implied from transitivity are omitted.

the algorithm and prove its correctness by showing that all steps preserve alternating simulation equivalence. (iii) We show that the output of the algorithm is indeed the unique minimum LTS (up to isomorphism) alternating-simulation equivalent to the input LTS $\mathcal{S}$. This, in turn, implies that for every LTS there is a unique minimum LTS equivalent modulo AS system that can be constructed using our algorithm.

3.1 Overview of the algorithm

The algorithm can be summarized as follows. For a system $S := (X, X_0, U, Y, \delta, H)$, we denote by $\mathsf{TranSize}(S) := |\delta| + |X_0|$, a measure for number of transitions in the system [4]. Let $|X| = n$ and $\mathsf{TranSize}(S) = m$.

Step 0: Construct the maximal alternating simulation relation, denoted by $R^{\max}$, from $\mathcal{S}$ to itself. This could be constructed using fixed-point algorithms as in [2] or the more efficient algorithm presented in [9], whose complexity is $O(m^2)$.

Step 1: Create a quotient system using $R^{\max}$ of $\mathcal{S}$ by combining all the equivalent states (and hence all their incoming transitions and outgoing transitions) to get a quotient of the system modulo AS. This requires $O(n + m)$ computations given the partition $\mathcal{P}$ (which can be constructed while building $R^{\max}$) as constructing the quotient transition relation from δ requires taking the union of all the outgoing transitions from any state in the given partition.

Recall that, if $(q, q') \in R^{\max}$ then if the controller can meet a specification from the state q then it will definitely meet it from state q'. Moreover, if the controller fails to meet the specification from q' it will definitely fail from q. In other words, q' (resp. q) is more advantageous position for the controller (resp. environment) as compared to q (resp. q'). This intuition is central to the next two steps.

Step 2: Remove irrational choices and redundant choices for the controller: For every $x \in X$ and every $a, b \in U(x), a \neq b$ if for every $x_b \in \mathrm{Post}(x, b)$ there exists an $x_a \in \mathrm{Post}(x, a)$ such that $(x_a, x_b) \in R^{\max}$, then delete all transitions from x on a. In other words, remove a from $U(x)$. This is because, for every possible environment move on taking an action b leads to a more (or equally) advantageous state for the controller as compared to any possible state the system can end up on action a by controller. To check this, every transition is compared with every other transition at most once. Hence, the complexity of this step in the worst case is bounded by $O(m^2)$.

[4]We add the cardinality of X_0 to total number of transitions because in principle the results we use from [9] assumes that there is a unique initial state. Multiple initial states can be simulated by adding silent transitions from a dummy initial state to all the states in X_0 which requires $|X_0|$ extra transitions

Step 3: Remove sub-optimal irrational choices for the environment: For every pair $x_1, x_2 \in X_0$, if $(x_1, x_2) \in R^{\max}$, then the choice of environment to start from x_2 will be irrational as x_1 is more advantageous position for the environment to start with. Hence, we remove x_2 from the initial state set (which is clearly an irrational move for the environment). Similarly, if $(x_1, x_2) \in R^{\max}$, then for every $a \in U$ if $x' \in \mathrm{Pre}(x_1, a) \cap \mathrm{Pre}(x_2, a)$, remove transition (x', a, x_2) from δ. This is because, if the system is at x', and if the controller chooses an action a, the choice of moving to x_2 instead of x_1 is irrational for the environment as x_1 is more advantageous state for the environment. hence, we delete the transition (x', a, x_2). Similarly to step 3, before its deletion (or not), any transition is compared with all other transitions at most once. Hence, the worst case complexity is bounded by $O(m^2)$.

Step 4: Remove Inaccessible States: Finally remove all the states that are not accessible from any initial state. This is a routine step with complexity $O(n+m)$. Note that while it seems that Steps 3 and 4 only remove transitions, this does not mean that they do not contribute in the reduction of number of states. Due to the removal of transitions, it could happen that a large fraction of the graph becomes unreachable. This is the step that cashes in the benefit of steps 3 and 4 in terms of reduction in state size.

The maximal alternating simulation relation from our working example (Fig. 1) is depicted in Fig. 2. Figures 3 and 4 illustrate the successive application of each step 1–4 on it.

3.2 Preserving equivalence Modulo AS: correctness results

In this section, we formally present the construction/reduction mentioned in each step 1–4 and show that those reductions preserve equivalence modulo AS. We also present results on the dimension reduction resulting from each step. We fix $\mathcal{S} := (X, X_0, U, Y, \delta, H)$ for this section as a given LTS and apply our reduction steps. For any $i \in \{1, 2, 3, 4\}$ the system resulting of applying step i: $S^i(\mathcal{S})$ is the system S_i.

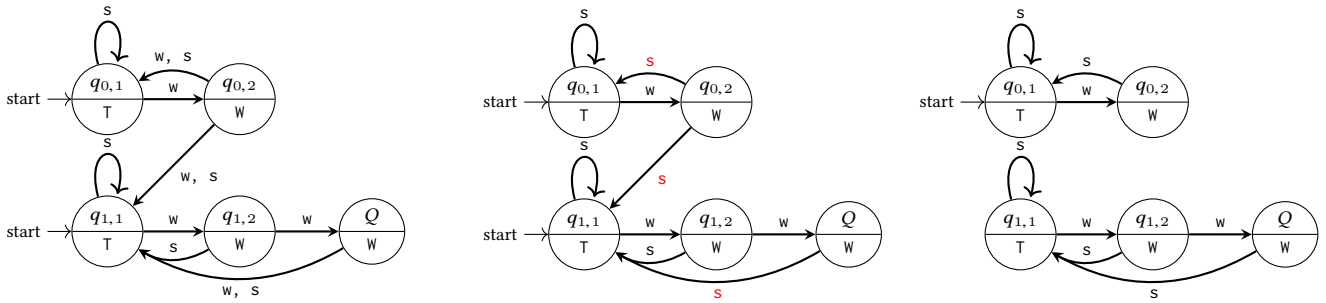
Step 1: Creating a quotient system. First, a quotient system $\mathcal{S}_1$ of $\mathcal{S}$ is created using $R^{\max}$ as follows. Consider the partition $\mathcal{P} = \{Q_1, Q_2, \ldots, Q_m\}$ of X where each Q_i is the maximal subset of X such that for any states $(p, q) \in Q_i$ it holds that $(p, q) \in R^{\max}$ and $(q, p) \in R^{\max}$.

Definition 3.1 (Alternating simulation quotient). The system $\mathcal{S}_1 := (X_1, X_{0,1}, U_1, \delta_1, Y, H_1)$ is called the *alternating simulation quotient* of $\mathcal{S}$ w.r.t. $R^{\max}$ iff $X_1 = \mathcal{P}$, $X_{0,1} = \{Q \mid Q \in X \wedge \exists q \in Q.\ q \in X_0\}$, $U_1 = U$, $\delta_1 = \{(Q, u, Q') \mid \exists q \in Q.\ \exists q' \in Q.\ (q, u, q)' \in \delta\}$, $\forall Q \in X_1 . H_1(Q) = H(q)$ for any $q \in Q$ (H_1 is well-defined as $\forall q, q' \in Q.\ H(q) = H(q')$).

This construction is similar to the celebrated quotient systems used for simulation and bisimulation; here we just make use of the already existing $R^{\max}$instead of performing a refinement algorithm, like it has been done for simulation equivalence [7]. Step 1 preserves equivalence modulo AS:

Lemma 3.2. $\mathcal{S} \simeq_{AS} \mathcal{S}_1$.

Let $\mathrm{Part} : X \mapsto X_1$ be the function that maps every state to its corresponding partition, and $R_1^{\max} \subseteq X_1 \times X_1$ be the smallest relation satisfying (I) $\forall (p, q) \in R^{\max}.(\mathrm{Part}(p), \mathrm{Part}(q)) \in R_1^{\max}$ and,



(a) Step 1: quotient system.

(b) Step 2: redundant actions removed (affected transitions in red).

(c) Step 3: irrational transitions removed.

Figure 3: System of Fig. 1 after steps 1 (left), 2 (middle) and 3 (right), where $Q = \{(q_{1,3}), (q_{1,4})\}$.

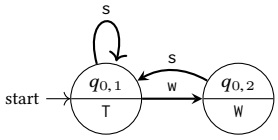


Figure 4: System of Fig. 1 after step 4. This is a minimal system modulo ASE.

(II) $\forall (P,Q) \in R_1^{\max}.\exists p \in P.\exists q \in Q.(p,q) \in R^{\max}$. Note that $\forall P, Q \in X_1.\exists p \in P.\exists q \in Q.(p,q) \in R^{\max} \Rightarrow \forall p' \in P.\forall q' \in Q.(p',q') \in R^{\max}$. This is because every $P, Q \in X_1$ are sets containing states of $\mathcal{S}$ which are equivalent modulo $R^{\max}$. Hence, if any element of P is related to any element Q with respect to $R^{\max}$, then by transitivity of $R^{\max}$ all elements of P are related to all elements of Q. Hence, (II) implies (III) $\forall (P,Q) \in R_1^{\max}.\forall p \in P.\forall q \in Q.(p,q) \in R^{\max}$. The following fact holds:

Lemma 3.3. *(1) $R_1^{\max}$ is the maximal ASR from $\mathcal{S}_1$ to itself. Moreover, (2) $R_1^{\max}$ is a partial order.*

In fact, if $R^{\max}$ is a partial order (i.e., $(p,q) \in R^{\max} \implies (q,p) \notin R^{\max}$ for every $p \neq q$), then step 1 does not affect $\mathcal{S}$.

Proposition 3.4. *$|X_1| \leq |X|$ and $\mathsf{TranSize}(S_1) \leq \mathsf{TranSize}(S)$, and no pair P, Q of X_1 is equivalent modulo AS. Moreover, if $R^{\max}$ is not antisymmetric, then $|X_1| < |X|$.*

Step 2: Removing irrational and redundant controller choices. We construct $\mathcal{S}_2 := (X_2, X_{0,2}, U_2, \delta_2, Y_2, H_2)$ from $\mathcal{S}$ as follows. $X_2 = X$, $X_{0,2} = X_0$, $U_2 = U$, $Y_2 = Y$, $H_2 = H$. Before defining δ_2, we define an ordering $\sqsubseteq_{\mathcal{S}}$ on elements of $X \times U$: $(p',u') \sqsubseteq_{\mathcal{S}} (p,u) \iff u \in U(p) \wedge \forall (p,u,q) \in \delta.\exists (p',u',q') \in \delta.(q',q) \in R^{\max}$. Note that $\sqsubseteq$ is a transitive relation. We say that an action u' is an *irrational move at a state p* of an LTS $\mathcal{S}$ iff $\exists u.(p,u) \sqsubseteq_{\mathcal{S}} (p,u') \wedge \neg((p,u') \sqsubseteq_{\mathcal{S}} (p,u))$. State p in this case is said to have *irrational moves*. Similarly, we say that u, u' are *equally rational* at a state p of an LTS iff $(p,u) \sqsubseteq_{\mathcal{S}} (p,u') \wedge ((p,u') \sqsubseteq_{\mathcal{S}} (p,u))$. Moreover, if u and u' are distinct then the state p, in this case, is said to have *redundant moves*. We construct δ_2 by removing all the transitions on irrational actions at p. Followed by this, we make available only one of the equally rational actions. This procedure preserves equivalence modulo AS. Let $\mathcal{I} : X \mapsto X$ be the identity function.

Lemma 3.5. *$\mathcal{S}_2 \preceq_{\mathcal{I}} \mathcal{S} \preceq_{R^{\max}} \mathcal{S}_2$. Hence, $\mathcal{S} \simeq_{AS} \mathcal{S}_2$.*

Proposition 3.6. *$|X_2| = |X|$, $\mathsf{TranSize}(\mathcal{S}_2) \leq \mathsf{TranSize}(\mathcal{S})$, and for every state $q \in X_2$, $U_2(q)$ only contains non-redundant rational actions. Moreover, if there are irrational or redundant actions available from any state q in $\mathcal{S}$, then $\mathsf{TranSize}(\mathcal{S}_2) < \mathsf{TranSize}(\mathcal{S})$.*

Step 3: Eliminating Irrational Choices for Environment. We construct $\mathcal{S}_3 := (X_3, X_{0,3}, U_3, \delta_3, Y_3, H_3)$ from $\mathcal{S}$ as follows. $X_3 = X$, $U_3 = U$, $Y_3 = Y$, $H_3 = H$. Before the construction of $X_{0,3}$ and δ_3 we define a new relation amongst transitions: any transition (p,u,q') in δ is called a *younger sibling* of a transition (p,u,q) in δ with respect to $\mathcal{S}$ iff $(q,q') \in R^{\max} \wedge (q',q) \notin R^{\max}$. Similarly, an initial state q_0' is called a younger sibling of yet another initial state q_0 with respect to $\mathcal{S}$ iff $(q_0,q_0') \in R^{\max} \wedge (q_0',q_0) \notin R^{\max}$. Then, $X_{0,3}$ and δ_3 are constructed from X_0 and δ by deleting all the younger siblings. In other words, given any state p and $u \in U(q)$, if there are two transitions (p,u,q') and (p,u,q) in δ and if $q \preceq_{AS} q'$ but not vice-versa (i.e., q is strictly more advantageous position for the environment as compared to q') then delete the transition (p,u,q') from $\mathcal{S}$, as the environment has no reason to choose q' over q. Note that this definition is similar to the *younger brother* definition of [7], but here we need to take the label of the transitions into account while defining the "sibling" relationship due to the definition of AS.

Lemma 3.7. *$\mathcal{S} \preceq_{\mathcal{I}} \mathcal{S}_3 \preceq_{R^{\max}} \mathcal{S}$. Thus, $\mathcal{S}_3 \simeq_{AS} \mathcal{S}$.*

Proposition 3.8. *$|X_3| = |X|$, $\mathsf{TranSize}(\mathcal{S}_3) \leq \mathsf{TranSize}(\mathcal{S})$, and $\mathcal{S}_3$ contains no transitions or initial states that are younger siblings of another transition or initial state, respectively. Moreover, if there is any younger sibling transition or initial state in $\mathcal{S}$, then $\mathsf{TranSize}(\mathcal{S}_3) < \mathsf{TranSize}(\mathcal{S})$.*

Step 4: Removing states inaccessible from initial state set X_0 in $\mathcal{S}$. Let X_∞ be the set of such states inaccessible from any initial state in X_0. Then $\mathcal{S}_4 := (X_4, X_0, U, \delta_4, Y, H)$, where $X_4 = X \setminus X_\infty$, $\delta_4 = \delta \cap (X_4 \times U_4 \times X_4)$.

Lemma 3.9. *$\mathcal{S}_4 \cong_{AS} \mathcal{S}$*

Proposition 3.10. *$|X_4| \leq |X|$, $\mathsf{TranSize}(\mathcal{S}_4) \leq \mathsf{TranSize}(\mathcal{S})$, and all states in $\mathcal{S}_4$ are accessible from $X_{0,4}$. Moreover, if X_∞ is non-empty then $|X_4| < |X|$.*

The combination of Lemmas 3.2, 3.5, 3.7 and 3.9 gives our main correctness result:

Theorem 3.11. $\mathcal{S} \simeq_{AS} S^4(S^3(S^2(S^1(\mathcal{S}))))$.

3.3 Optimality results

Theorem 3.12 (Necessary Condition for Minimal Equivalent System modulo AS). *Given any LTS $\mathcal{S}$, a minimal LTS $\mathcal{S}_{\min} := (X_{min}, X_{0,min}, U_{min}\delta_{min}, Y_{min}, H_{min})$ equivalent to the former modulo AS necessarily satisfies the following conditions:*

- N_1 *For any $p, q \in X_{min}$, $(\mathcal{S}_{\min}(p) \simeq_{AS} \mathcal{S}_{\min}(q)) \implies p = q$. That is, no two distinct states are equivalent modulo AS to each other.*
- N_2 *For any $p \in X_{min}$, p does not have any irrational or redundant moves.*
- N_3 *$\nexists\, t_1, t_2 \in \delta_{min}, x_1, x_2 \in X_{0,min}$ such that t_1 is a younger sibling of t_2 or x_1 is a younger sibling of x_2.*
- N_4 *All the states in X_{min} are connected from some $x_0 \in X_{0,min}$.*

Proof. This theorem is a consequence of Propositions 3.4,3.6,3.8, 3.10. If any condition $i \in \{1, 2, 3, 4\}$ is violated by $\mathcal{S}_{\min}$, Step i can be applied to get a strictly smaller system preserving equivalence modulo AS which contradicts that $\mathcal{S}_{\min}$ is minimal. □

Lemma 3.13. *$\mathcal{S}_{out} = S^4(S^3(S^2(S^1(\mathcal{S}))))$, satisfies the necessary conditions in Theorem 3.12.*

By Proposition 3.4, we know that after step 1 we get a $S^1(\mathcal{S})$ that satisfies N_1. The proof then shows that after performing each step i, we get a system satisfying N_i. Moreover, if the input to the system satisfied any of the previous properties, they will continue to respect it.

We call any LTS satisfying the conditions in Theorem 3.12 as *potentially minimal systems.*

In the following we show that the conditions in Theorem 3.12 are also sufficient for minimality modulo ASE. In fact, we prove something stronger: such a minimal system is unique up to a variant of isomorphism which we introduce as bijective alternating bisimulation isomorphism (BABI). We show this by proving that any two potentially minimum systems $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $\mathcal{S}_1 \simeq_{AS} \mathcal{S}_2$ implies that they are BABI to each other. It is important to note that for two structures to be connected via a BABI implies the existence of a bijective alternating bisimulation relation, but the converse is not necessarily true. Hence, the former is stricter than the latter. In fact, the existence of a bijective alternating bisimulation does not necessarily preserve the transition size [5].

Definition 3.14 (Bijective Alternating Bisimulation Isomorphism). Given any two systems $\mathcal{S}_j := (X_j, X_{0,j}, U_j\delta_j, Y_j, H_j)$, $j \in \{1, 2\}$, we say that $\mathcal{S}_1 \cong_{\text{is}} \mathcal{S}_2$ iff there exists a bijective function $\mathcal{A} : X_1 \mapsto X_2$ such that $\forall p \in X_1.\mathcal{A}(p) = q$ implies:

(1) $p \in X_{0,1} \iff q \in X_{0,2}$.
(2) $H_1(p) = H_2(q)$. Vertex labelling is preserved.
(3) There exists a bijection $G_{p,q} : U_1(p) \mapsto U_2(q)$ such that $\forall a \in U_1(p).\, \text{Post}^{\mathcal{S}_2}(q, b) = \{\mathcal{A}(p') \mid p' \in \text{Post}^{\mathcal{S}_1}(p, a)\}$ where $b = G_{p,q}(a)$.

[5] Consider single state systems B_1, B_2 one with self loop on a and other with two self loops each on a and $\tilde{a}$.

Hence, $\mathcal{S}_1 \cong_{\text{is}} \mathcal{S}_2$ implies $|X_0| = |X_1|$ (implied by the existence of bijection $\mathcal{A}$), $|X_{0,1}| = |X_{0,2}|$ (implied by 1 and bijectivity of $\mathcal{A}$), total number of transitions are equal in both $\mathcal{S}_1, \mathcal{S}_2$ (implied by 3). Hence, $|X_1| = |X_2| \wedge \text{TranSize}(\mathcal{S}_1) = \text{TranSize}(\mathcal{S}_2)$.

Lemma 3.15. *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be any potentially minimal systems. Then, $\mathcal{S}_1 \simeq_{AS} \mathcal{S}_2$ implies $\mathcal{S}_1 \cong_{\text{is}} \mathcal{S}_2$.*

Proof. Given potentially minimal systems $j \in 1, 2$, $\mathcal{S}_j := (X_j, X_{0,j}, , U_j, \delta_j, Y_j, H_j)$, such that $\mathcal{S}_1 \simeq_{AS} \mathcal{S}_2$ we show that $\mathcal{S}_1 \cong_{\text{is}} \mathcal{S}_2$. As $\mathcal{S}_1 \simeq_{AS} \mathcal{S}_2$, denote the maximal ASR from $\mathcal{S}_1$ to $\mathcal{S}_2$ by $R_1^{\max}$ and that from $\mathcal{S}_2$ to $\mathcal{S}_1$ by $R_2^{\max}$. Let $\mathcal{A} \subseteq X_1 \times X_2$ such that $\mathcal{A} := \{(p, q) \mid (p, q) \in R_1^{\max} \wedge (q, p) \in R_2^{\max}\} = R_1^{\max} \cap (R_2^{\max})^{-1}$. Note that any pair $(p, q) \in \mathcal{A}$ iff $\mathcal{S}_1(p) \simeq_{AS} \mathcal{S}_2(q)$. We prove the result by showing that $\mathcal{A}$ is a bijection satisfying all the 3 conditions of the Def. 3.14. Condition 2 is straightforward: every pair of states occurring in $\mathcal{A}$ are equivalent modulo Alternating Simulation and hence have identical labels.

Now let us focus on Condition 3. We show that $\mathcal{A}$ is a relation satisfying condition 3 of Def. 3.14. For that, we construct a relation $G_{p,q}$ satisfying condition 3; then we see it is a bijection. Note that any $(p, q) \in \mathcal{A}$ implies (C1) $(p, q) \in R_1^{\max} \wedge$ (C2) $(q, p) \in R_2^{\max}$.

Construct a candidate relation $G'_{p,q}$ satisfying the consequent of condition 3 of Def. 3.14. The former implies (C1.1) for every $a \in U_1(p)$ we can *choose* a $b \in U_2(q)$ such that for every state $q' \in \text{Post}^{\mathcal{S}_2}(q, b)$ we can find a state $p' \in \text{Post}^{\mathcal{S}_1}(p, a)$ such that $p' \preceq_{AS} q'$ $((p', q') \in R_1^{\max})$.
(C2) and (C1.1) together imply (C2.1) for the b chosen in previous step (C1.1) we can *find* an $a' \in U_1(p)$ such that for every state $p'' \in \text{Post}^{\mathcal{S}_1}(p, a')$ we can find a state $q'' \in \text{Post}^{\mathcal{S}_2}(q, b)$ such that $q'' \preceq_{AS} p''$ $((q'', p'') \in R_2^{\max}$.

Combining (C1.1 and C2.1) we get (C3.1)$\forall a \in U_1(p).\exists b \in U_2(q).$ $\exists a' \in U_1(p)$ such that for every state $p'' \in \text{Post}^{\mathcal{S}_1}(p, a')$ there exists a state $q'' \in \text{Post}^{\mathcal{S}_2}(q, b)$ such that $(q'', p'') \in R_2^{\max}$. Moreover, for this q'' we can find a state $p' \in \text{Post}^{\mathcal{S}_1}(p, a)$ such that $(p', q'') \in R_1^{\max}(p' \preceq_{AS} q'')$. Hence, by transitivity of alternating simulation pre-order, for every state $p'' \in \text{Post}^{\mathcal{S}_1}(p, a')$ there exists a state $p' \in \text{Post}^{\mathcal{S}_1}(p, a)$ such that $p'' \preceq_{AS} p'$. Hence, $(p, a) \sqsubseteq_{\mathcal{S}_1} (p, a')$. Thus, if $a \neq a'$ then a is either redundant or an irrational choice for the controller at state p in LTS $\mathcal{S}_1$. This contradicts the assumption that $\mathcal{S}_1$ satisfies condition N_2. Hence, (C4)$a = a'$.

Thus combining (C3.1) and (C4) we get (C3) for any $a \in U_1(p)$ we can find $b \in U_2(q)$ such that for every state in $p' \in \text{Post}^{\mathcal{S}_1}(p, a)$ we can find $q' \in \text{Post}^{\mathcal{S}_2}(q, b)$ such. that $q' \preceq_{AS} p'$ $((q', p') \in R_2^{\max})$; at the same time, by (C1.1), for every state $q'' \in \text{Post}^{\mathcal{S}_2}(q, b)$ there exists a state in $p'' \in \text{Post}^{\mathcal{S}_1}(p, a)$ such that $p'' \preceq_{AS} q''$ $((p'', q'') \in R_1^{\max})$.

Note that (C3) is equivalent to $\psi(p, q) := \forall a \in U_1(p).\exists b \in U_2(q).\varphi(p, q, a, b)$, where $\varphi(p, q, a, b) = \varphi_1(p, q, a, b) \wedge \varphi_2(p, q, a, b)$, $\varphi_1 := \forall p' \in \text{Post}^{\mathcal{S}_1}(p, a).\ \exists q' \in \text{Post}^{\mathcal{S}_2}(q, b).\ (q', p') \in R_2^{\max}$ and $\varphi_2 := \forall q'' \in \text{Post}^{\mathcal{S}_2}(q, b).\ \exists p'' \in \text{Post}^{\mathcal{S}_1}(p, a).\ (p'', q'') \in R_1^{\max}$. Let $G'_{p,q} \subseteq U_1(p) \times U_2(q)$ such that $(a, b) \in G'_{p,q}$ iff $\varphi(p, q, a, b)$ holds.

Verify that $G'_{p,q}$ satisfies the consequent of condition 3. Note that for every $(a, b) \in G'_{p,q}$ we have that every state in $p' \in \text{Post}^{\mathcal{S}_1}(p, a)$ some state in $q' \in \text{Post}^{\mathcal{S}_2}(q, b)$ such that $q' \preceq_{AS} p'$

$((q', p') \in R_2^{\max}$, due to φ_1), which in turn, due to φ_2, satisfies $q' \preceq_{AS} p''$ for some state $p'' \in \text{Post}^{\mathcal{S}_1}(p, a)$ (i.e, $(p'', q') \in R_1^{\max}$).

Now we prove that $p' = p''$ by contradiction. Suppose that $p' \neq p''$. Then, by transitivity of alternating simulation, $p' \preceq_{AS} p''$. Hence, transition (p, a, p'') is a younger sibling of transition (p, a, p') which contradicts the assumption that N_3 is satisfied by $\mathcal{S}_1$. Hence (C5) $p' = p''$

Thus, (C5.1) for any $(a, b) \in G'_{p,q}$, for each $p' \in \text{Post}^{\mathcal{S}_1}(p, a)$ there is a state $q' \in \text{Post}^{\mathcal{S}_2}(q, b)$ such that $p' \preceq_{AS} q'$ (by φ_1). Moreover, this q' is in turn alternately simulates p' (by C5 and φ_2). Hence, $(p', q') \in \mathcal{A}$. Now we prove that there is a unique q' such that $(p', q') \in \mathcal{A}$. (C5.2) Suppose there exists a $p' \in \text{Post}^{\mathcal{S}_1}(p, a)$ that $\geq$ two distinct states $q', q'' \in \text{Post}^{\mathcal{S}_2}(q, b)$, then by (C5.1) $(p', q') \in \mathcal{A}$ and $(p', q'') \in \mathcal{A}$. This would imply that q' and q'' are equivalent modulo AS. This contradicts the assumption that $\mathcal{S}_2$ satisfies N_1. Hence, for every $p' \in \text{Post}^{\mathcal{S}_1}(p, a)$ there exists a unique $q' \in \text{Post}^{\mathcal{S}_2}(q, b)$ such that $(p', q') \in \mathcal{A}$. By symmetry of condition φ, for every $q' \in \text{Post}^{\mathcal{S}_2}(q, b)$ there exists a unique $p' \in \text{Post}^{\mathcal{S}_1}(p, a)$ such that $(p', q') \in \mathcal{A}$.

This implies (C6) $\text{Post}^{\mathcal{S}_2}(q, b) = \{q' \mid (p', q') \in \mathcal{A}$ and $p' \in \text{Post}^{\mathcal{S}_1}(p, a)\}$. Hence, by $\psi(p, q)$ we have (C7) i.e.For any $a \in U_1(p)$ we can find $b \in U_2(q)$ such that $(a, b) \in G'_{p,q}$.

By symmetry, repeating all steps starting from (C2), we get (C8) for any $(p, q) \in \mathcal{A}$ we can construct a relation $G''_{q,p} \subseteq U_2(q) \times U_1(p)$ such that $\text{Post}^{\mathcal{S}_1}(p, a) = \{p' | (q', p') \in \mathcal{A}^{-1} \wedge q' \in \text{Post}^{\mathcal{S}_1}(q, a)\}$, reading (9) $\forall b \in U_2(q). \exists a \in U_1(p). (b, a) \in G''_{q,p}$.

Building the bijection $G_{p,q}$. We now prove that $G_{p,q} := G'_{p,q} \cap G''^{-1}_{q,p}$ is a well-defined bijective function such that for any $a \in U_1(p), b \in U_2(q)$, $b = G_{p,q}(a) \implies \text{Post}^{\mathcal{S}_2}(q, b) = \{q' \mid (p', q') \in \mathcal{A}$ and $p' \in \text{Post}^{\mathcal{S}_1}(p, a)\}$. This proves that $\mathcal{A}$ satisfies the required condition 3.

(10) For $G_{p,q}$ to be a well-defined function, we need to show that for any $a \in U_1(p)$, there is (A) at least 1 and (B) at most 1 $b \in U_2(q)$ such that $(a, b) \in G_{p,q}$; (A) is implied by (C7).

For (B), assume that for distinct $b_1, b_2 \in U_2(q)$ $(a, b_1), (a, b_2) \in G_{p,q}$. By (C6), we get that $\text{Post}^{\mathcal{S}_2}(q, b_1) = \{q' \mid (p', q') \in \mathcal{A}$ and $p \in \text{Post}^{\mathcal{S}_1}(p, a)\} = \text{Post}^{\mathcal{S}_2}(q, b_2)$. But this implies that b_1 is a redundant controller choice at state q in LTS $\mathcal{S}_2$ which contradicts N_2 for system $\mathcal{S}_2$. Hence, $G_{p,q}$ is a well-defined function. Applying the same reasoning on $G^{-1}_{q,p} = G^{-1}_{p,q} \cap G''_{q,p}$, we get that $G^{-1}_{q,p}$ is also a well-defined function, proving that $G_{p,q}$ is a bijection.

As $G_{p,q}$ contains elements from $G'_{p,q}$, any $(a, b) \in G_{p,q}$ satisfies (C6). Hence $G_{p,q}$ is the required bijection for condition 3 in Def. 3.14.

$\mathcal{A}$ is a bijection and satisfies condition 1: (C11) First we show that every initial state is related to a unique initial state. That is, (C11.1 $\mathcal{A}_0 := \mathcal{A} \cap (X_{0,1} \times X_{0,2})$ is a bijection between $X_{0,1}$ and $X_{0,2}$. We first show by contradiction that $\mathcal{A}_0$ is a well-defined function. If it is not, then there exists a state $p \in X_{0,1}$ such that (C11.2) either p is not related to any state q in $\mathcal{A}_0$ or, (C11.3) $\exists q, q' \in X_{0,2}. (p, q) \in \mathcal{A}_0 \wedge (p, q') \in \mathcal{A}_0 \wedge q \neq q'$. Note that $R_2^{\max}$ is an ASR from $\mathcal{S}_2$ to $\mathcal{S}_1$, hence from condition (C11.1) of Def. 2.1, p being an initial state of $\mathcal{S}_1$ implies $\exists q'. (q', p) \in R_2^{\max}$. Now, (due to similar restrictions imposed by condition (C11.1) for $R_1^{\max}$ being an ASR from $\mathcal{S}_1$ to $\mathcal{S}_2$) this q' is related with some initial state p' of $\mathcal{S}_1$. Hence, $\exists p'. (p', q') \in R_1^{\max}$. Now note that if $p' = p$, then (p, q') should be in $\mathcal{A}_0$ (by definition) which contradicts the assumption that (C11.2) holds. If $p' \neq p$, we have $\mathcal{S}_1(p) \preceq_{AS} \mathcal{S}_2(q') \wedge \mathcal{S}_1(q') \preceq_{AS} \mathcal{S}_2, (p) \wedge p \neq p'$. Hence, by transitivity of $\preceq_{AS}$, $p \preceq_{AS} p', p \neq p'$ and both are initial states. This implies that p is an initial state which is younger sibling of p', which contradicts the assumption that $\mathcal{S}_1$ satisfies condition N_3. Note that to prove $\mathcal{A}_0$ is a bijection, it suffices to show that $\mathcal{A}_0^{-1}$ is a well-defined function, which is a symmetrical proof to that of $\mathcal{A}_0$.

(C12) **Now we show that $\mathcal{A}$ is a partial function.** That is, every $p \in X_1$ is mapped to a unique $q \in X_2$ via $\mathcal{A}$. Suppose it is not, i.e., there exists a state $p \in X_1$ which is related to two distinct states $q, q' \in X_2$. Hence, $(p, q), (p, q') \in \mathcal{A}$. By definition of $\mathcal{A}$, we have that $(q, p) \in R_2^{\max}$ and $(p, q') \in R_1^{\max}$, implying (by transitivity) that $q \preceq_{AS} q'$; symmetrically, $(p, q) \in R_1^{\max}$ and $(q', p) \in R_2^{\max}$, implying that $q' \preceq_{AS} q$. Thus, q and q' are equivalent modulo AS which is a contradiction as $\mathcal{S}_2$ satisfies N_1. Symmetrically, $\mathcal{A}^{-1}$ is a partial function relation.

(C13) Note that by (C11) every initial state is mapped to some initial state. By (C12), every state is mapped to a unique state. Hence, every initial state can only be mapped to a *unique* initial state.

We now show that $\mathcal{A}$ (and by symmetry $\mathcal{A}^{-1}$) is a well-defined function. We already showed that $\mathcal{A}$ (and $\mathcal{A}^{-1}$) are partial functions (C12). It remains to be proved that a state in X_1 can be mapped to at least one state in X_2 under $\mathcal{A}$ (and vice-versa under $\mathcal{A}^{-1}$). We already showed the latter for states in X_0; we now show it for the remaining states. We prove this using contradiction. Assume that there exists a state in X_1 that is not mapped to any state in X_2 under $\mathcal{A}$. Let $\mathcal{P}$ be the set of all such states. As X_1 is a finite set, so is $\mathcal{P}$. Note that by assumption N_4, $\mathcal{S}_1$ does not contain any inaccessible state. Hence, every state in $p \in X_1$ can be reached from some initial state in $p_0 \in X_{0,1}$ in $|X_1|$ or less steps. Let c be the minimum number of steps required to reach the state $p' \in \mathcal{P}$ that is the nearest to the initial state set. That is, no state in $\mathcal{P}$ can be reached in $c - 1$ or less steps and there is at least 1 state $p' \in \mathcal{P}$ that is reachable from initial state in c steps. Consider a state $p \in \text{Pre}(p', a)$ for some $a \in U_1$. Because p is reachable in $c - 1$ steps, there exists a $q \in X_2$ such that $(p, q) \in \mathcal{A}$. Now we recover (C5.2): for every $a \in U_1(p). \exists b \in U_2(q). \forall p'' \in \text{Post}^{\mathcal{S}_1}(p, a)$ there exists a unique $q'' \in \text{Post}^{\mathcal{S}_2}(q, b)$ such that $(p'', q'') \in \mathcal{A}$. This implies that for p' too there exists a unique $q' \in \text{Post}^{\mathcal{S}_2}(q, b)$ such that $(p', q') \in \mathcal{A}$. This leads to the contradiction, thus $\mathcal{A}$ is a well-defined function. By symmetry, the same holds for $\mathcal{A}^{-1}$. This implies $\mathcal{A}$ is a bijection. □

Lemmas 3.13 and 3.15 imply our main optimality result:

THEOREM 3.16. *The system $\mathcal{S}_{out} = S^4(S^3(S^2(S^1(\mathcal{S}))))$ is the unique (up to BABI) minimal system that is ASE to $\mathcal{S}$.*

4 CASE STUDY: SCHEDULING PETC SYSTEMS

Event-triggered control (ETC) is an aperiodic sampled-data control paradigm where a *plant* samples its state and sends it to a *controller* upon the occurrence of a designed event. Immediately after, the controller calculates a *control input* that is sent to the actuators of the plant. Despite reducing control-related traffic, ETC's aperiodic traffic makes it challenging to accommodate multiple ETC loops

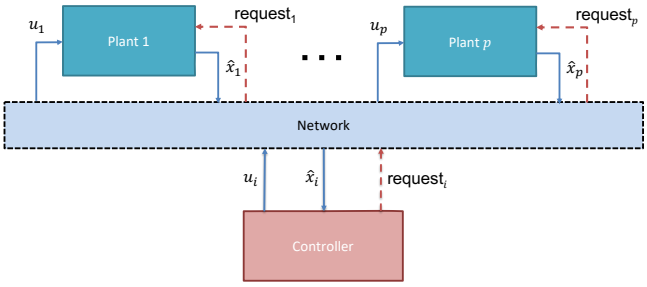


Figure 5: A network of p ETC systems. Plant i can decide (based on the event occurrence) when to send its state sample $\hat{\boldsymbol{x}}_i$ or the controller can request it.

sharing a communication channel: packet collisions are bound to happen, putting the stability of the controlled plants at risk. Therefore, a scheduler must be introduced in the system, in order to adjust the traffic and prevent said collisions, while ensuring stability and performance of the individual plants.

Figure 5 depicts a networked control system (NCS) with multiple control loops sharing a single communication channel. The plants are described by an ordinary differential equation (ODE), and the controller runs individual *control functions* for each of the plants, as follows:

$$\begin{aligned} \dot{\boldsymbol{x}}_i(t) &= f_i(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t), \boldsymbol{w}_i(t)), \\ \boldsymbol{u}_i(t) &= g_i(\hat{\boldsymbol{x}}_i(t)), \end{aligned} \tag{1}$$

where $\boldsymbol{x}_i(t) \in \mathbb{R}^{n_i}$ is the state of plant i, $\boldsymbol{u}_i(t) \in \mathbb{R}^{m_i}$ is its control input, and $\boldsymbol{w}_i(t) \in \mathbb{R}^{d_i}$ represents the external disturbances that act on it. The variable $\hat{\boldsymbol{x}}$ represents the sampled-and-held version of state $\boldsymbol{x}$, satisfying

$$\hat{\boldsymbol{x}}_i(t) = \begin{cases} \boldsymbol{x}(t_{i,k}), & \text{if } t \in [t_{i,k}, t_{i,k+1}), \\ \boldsymbol{0}, & \text{otherwise,} \end{cases} \tag{2}$$

where $t_{i,k} \in \mathbb{R}_+$ represents the k-th communication instant for the data of plant i. In regular ETC, the communication instants are dictated by a *triggering condition*, such as the seminal one proposed in [31]:

$$\begin{aligned} t_{i,k+1} = t_{i,k+1}^{\text{trigger}} &:= \sup\{t \in h\mathbb{N} \mid t > t_{i,k} \text{ and } \phi(\boldsymbol{x}_i(t), \hat{\boldsymbol{x}}_i(t)) \leq 0\}, \\ \phi(\boldsymbol{x}_i(t), \hat{\boldsymbol{x}}_i(t)) &= |\boldsymbol{x}_i(t) - \hat{\boldsymbol{x}}_i(t)| - \sigma_i |\boldsymbol{x}_i(t)|, \end{aligned} \tag{3}$$

where $\sigma_i \in [0, 1)$ is a design parameter. The parameter h discretizes the time axis, meaning that events can only take place in multiples of h. This represents, for simplicity, also the *channel occupancy time*, which is the time it takes for a state measurement $\hat{\boldsymbol{x}}_i(t)$ and the subsequent control action $u_i(t)$ to be sent over the network. In fact, this discretization makes the sampling effectively a periodic event-triggered control, or PETC [18].

If multiple control systems operate with communication instants dictated by (3), it is generally impossible to prevent communication conflicts in the network; hence we introduce a possibility for the controller to *request* a state sample for any plant *before* its event actually happens. This can prevent collisions, while it is also sound from a control-systems perspective: in ETC, events are designed to happen before an underlying Lyapunov function stops decreasing sufficiently fast, thus ensuring closed-loop stability, see, e.g., [17, 31]. This makes *early sampling* a safe choice from a control performance perspective, and this feature has been extensively exploited in the event-based literature [3, 26], including in the context of scheduling of ETC systems [12]. Therefore, the sampling times $t_{i,k}$ can either occur upon triggering of the condition as in (3), or be *requested* earlier by the scheduler, satisfying

$$t_{i,k+1} \in \{t \in h\mathbb{N} \mid t > t_{i,k} \text{ and } t \leq t_{i,k+1}^{\text{trigger}}\}. \tag{4}$$

The quantity $\tau_{i,k} := t_{i,k+1} - t_{i,k}$ is called *inter-sample time*. When given these degrees of freedom, the most fundamental question one needs to answer is whether it is possible for a scheduler to coordinate the traffic generated by the p PETC loops while avoiding collisions and ensuring that the communications are timely. We assume that the device that runs the scheduler is capable of listening to all traffic, thus having access to the sampled states of all systems. In fact, this can be the same device that runs the control functions, which is the case depicted in Fig. 5.

The early-sampling PETC schedulability problem. Consider a network containing p control-loops (1) and $C < p$ communication channels with channel occupancy time h. Our main goal is to determine whether there exists a strategy that, at every time $t \in h\mathbb{N}$, given the available sampled states $\hat{\boldsymbol{x}}_i(t_{i,k}), \forall i, k$ such that $t_{i,k} < t$, determines which (if any) loops must send their samples to the controller. The number of loops sending their samples must be no greater than c, and for each loop i, $t \leq t_{i,k+1}^{\text{trigger}}$ must hold; that is, no controller can miss its *deadline* $t_{i,k+1}^{\text{trigger}}$. If a scheduler can be found, we also want to retrieve one such scheduling strategy for real-time implementation.

For simplicity, we assume for the rest of this paper that the time units are selected such that $h = 1$.

4.1 PETC traffic models as finite-state transition systems

The problem described above can be seen as a safety control synthesis problem for a hybrid system, which is in general undecidable [1, 20]. To deal with decidable problems, the control loops i have been abstracted as timed-game automata (TGA) in [23], and later as regular transition systems in [12], by assuming the same discrete nature of sampling instants as we assume here. For details on how to construct such abstractions, see [12, 13] for linear systems without disturbance, and [15] for general perturbed nonlinear systems. In these abstractions, each state q is a different region $\mathcal{R}_q \subset \mathbb{R}^n$ and associated an interval of possible inter-sample times $\{\tau_q^{\text{low}}, \tau_q^{\text{low}} + 1, \ldots \tau_q^{\text{high}}\}$ at which a trigger can occur. The scheduler can choose to sample earlier than τ_q^{low}, or sample during the aforementioned interval as long as a trigger has not yet occurred. From each state q, the set of possible regions reached depends on the chosen inter-sample time τ, regardless of whether the sample is determined by the scheduler or the triggering condition. Hence, the abstraction process outputs a set of transitions $\Delta \subset Q \times T \times Q$, where $(q, c, q') \in \Delta$ means that $q' \in Q$ can be reached from $q \in Q$ if the inter-sample time is $\tau \in T$. From this, we derive the following definition of PETC traffic model:

Definition 4.1 (PETC traffic model). A finite PETC traffic model with scheduler actions is the transition system $\mathcal{S}_{\text{PETC}} := (X, X_0, \{\mathtt{w}, \mathtt{s}\}, \delta_{\text{wait}} \cup \delta_{\text{sched}} \cup \delta_{\text{trigger}}, H)$ where

- $X = \{(q, c) \mid q \in Q, c \in \{1, 2, ..., \tau_q^{\text{high}}\}\}$,
- $\delta_{\text{wait}} = \{(q, c), \mathtt{w}, (q, c+1) \mid (q, c) \in X \text{ and } c < \tau_q^{\text{high}}\}$,
- $\delta_{\text{sched}} = \{(q, c), \mathtt{s}, (q', 0) \mid (q, c) \in X \text{ and } (q, c, q') \in \Delta\}$,
- $\delta_{\text{trigger}} = \{(q, c), \mathtt{w}, (q', 0) \mid (q, c) \in X \text{ and } c \geq \tau_q^{\text{low}}$ and $(q, c, q') \in \Delta\}$,
- $H(q, c) = \mathtt{T}$ if $c = 0$, or $\mathtt{W}$ otherwise.

The actions w (for wait) and s (for sample) are the scheduler actions; as the spontaneous trigger of a given loop is out of the control of the scheduler, these transitions are considered (adversarial) nondeterminism for the scheduler. This is why the set δ_{trigger} is a set of sampling transitions, but they occur when the action wait is chosen. The output T represents when a transmission has just occurred, while W means that the loop waited. The initial state depends on the particularities of the scheduling problem and will be discussed later.

Our running example, Fig. 1, depicts a simple PETC traffic model with only two regions. This example contains only two regions $\mathcal{R}_0$ and $\mathcal{R}_1$, mapped into q_0 and q_1, respectively, with $\tau_{q_0}^{\text{low}} = \tau_{q_0}^{\text{high}} = 2$, and $\tau_{q_1}^{\text{low}} = 3$ and $\tau_{q_1}^{\text{high}} = 4$. The states $(q_1, 3)$ and $(q_1, 4)$ represent the *triggering phase* of place q_1: even if the scheduler decides to wait, the sampling can occur in any of these states.

4.2 A general result on ETC scheduling

Using the reduction in Section 3, a first general result can be derived for scheduling of PETC.

Definition 4.2 (Reduced PETC traffic model). A reduced PETC traffic model with scheduler actions is the transition system $\mathcal{S}'_{\text{PETC}} := (X', X_0 \cap X', \{\mathtt{w}, \mathtt{s}\}, \delta'_{\text{wait}} \cup \delta'_{\text{sched}}, H)$ where

- $X' = \{(q, c) \mid q \in Q, c \in \{1, 2, ..., \tau_q^{\text{low}}\}\}$,
- $\delta'_{\text{wait}} = \{(q, c), \mathtt{w}, (q, c+1) \mid (q, c) \in X' \text{ and } c < \tau_q^{\text{low}}\}$,
- $\delta'_{\text{sched}} = \{(q, c), \mathtt{s}, (q', 0) \mid (q, c) \in X' \text{ and } (q, c, q') \in \Delta\}$,
- $H(q, c) = \mathtt{T}$ if $c = 0$, or $\mathtt{W}$ otherwise.

The difference between Def. 4.2 and Def. 4.1 is that, in the former, the sampling always happens at most at τ_q^{low} for every q, and that this point in time it is a scheduled sampling. In other words, there is no event-based sampling anymore, but the scheduler may decide to sample at the first moment in which it knows that an event trigger could occur. This is very similar in spirit to *self-triggered control* (STC, see [3, 26]), where the controller chooses the sampling time by predicting a worst-case situation in which the event-triggered control would occur. Thus, Def. 4.2 is can also be regarded as a traffic model for STC systems, again allowing early sampling. Fig. 6 shows the reduced model from Fig. 1. The interesting fact is that these two approaches are equivalent from a schedulability perspective:

Proposition 4.3. [6] *The PETC traffic model from Def. 4.1 and its reduced model from Def. 4.2 are alternating-simulation equivalent, provided* $X_0 \subseteq X'$.

[6] See the proof in the arXiv preprint of this paper, arXiv:2203.01672.

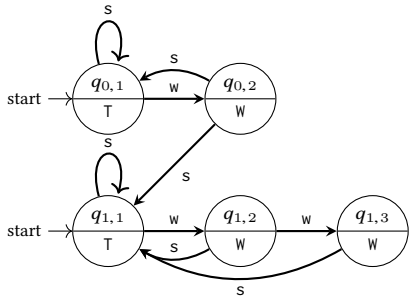


Figure 6: Reduced PETC traffic model of Fig. 1.

The interpretation of this result is simple: the choice of waiting at time τ_q^{low} has no advantage over sampling, because in the worst case the environment may choose to sample anyway. Hence, from a schedulability perspective, *ETC brings no benefit over a STC-like sampling strategy that chooses to trigger on the earliest ETC triggering time.* Naturally, this general result does not give the minimal system, which depends on the structure of the particular abstraction, as will be illustrated in the next section.

4.3 Numerical example

Consider p two-dimensional open-loop-unstable linear systems, borrowed from [31], of the form (1) where

$$f_i(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t), \boldsymbol{w}_i(t)) := \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} \boldsymbol{x}_i(t) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{u}_i(t), \qquad g_i(\hat{\boldsymbol{x}}_i(t)) := \begin{bmatrix} 1 & -4 \end{bmatrix} \hat{\boldsymbol{x}}_i(t). \tag{5}$$

The triggering condition is (3) with $\sigma = 0.7$. Since all systems have the same model, only one traffic abstraction is needed. We use the abstraction method in [11], where a parameter $l \in \mathbb{N}$ is given to define the depth of the abstraction process: the higher l is, the tighter the simulation relation is w.r.t. the original infinite system. Denote by $\Delta_l \subseteq Q_l \times T \times Q_l$ the transition relation from the abstraction using depth l, and the resulting PETC traffic model (Def. 4.1) by $\mathcal{S}_l$.

We consider the problem of scheduling on a single channel. From a practical perspective, the scheduling problem requires an initialization phase. When the systems are connected to the network, their states will only be known to the scheduler (and the controller) after the first sample. Because there is only one channel, the timing of the initial transmissions have to be decided by the scheduler, and this timing must be bounded to keep the plant's state under a reasonable distance from its initial value. Let T_0 be this time bound (in number of steps). To model this initialization phase, we append to $\mathcal{S}_l$ the states $i_1, i_2, ...i_{T_0}$ and transitions $i_k \xrightarrow{\mathtt{w}} i_{k+1}$ for all $k < T_0$ and $i_k \xrightarrow{\mathtt{s}} (q, 0)$ for all $k \leq T_0$ and $q \in Q_l$. The initial set is simply $X_0 = \{i_1\}$. In this example, T_0 was set to 10.

We implemented our minimization algorithm in Python and performed the minimization on $\mathcal{S}_l, l = 1, 2, 3$. The statistics of the traffic model before and after minimization modulo ASE are displayed in Table 1. The additional reduction w.r.t. only step 1 (quotient system) is evident in all cases. The most interesting phenomenon is the striking reduction of the traffic models for $l = 1, 2$ to

Table 1: Size of abstractions before and after minimization, and CPU time to minimize the system (in all cases $|X_0| = 1$).

		Original		Quotient		Minimal		CPU
		$\lvert X\rvert$	$\lvert\delta\rvert$	$\lvert X\rvert$	$\lvert\delta\rvert$	$\lvert X\rvert$	$\lvert\delta\rvert$	time
l	1	153	832	118	571	**11**	**21**	657 ms
	2	518	1879	405	1566	**11**	**21**	8.24 s
	3	683	2412	604	2262	587	2126	15 s

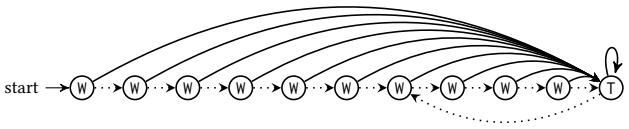


Figure 7: Minimized system for the numerical example, $l \in \{1, 2\}$. State labels are their outputs, dashed lines are `w` actions and full lines are `s` actions.

Table 2: Scheduler size and CPU time using BDDs.

		Original		Minimal	
p	l	Schedule size	CPU time	Schedule size	CPU time
2	1	3 kB	3 ms	894 B	498 µs
3	1	7.6 kB	8 ms	1.9 kB	783 µs
4	1	19 kB	16 ms	4.2 kB	1.4 ms
5	1	47 kB	36 ms	9.5 kB	2.5 ms
6	1	None	**7.35 s**	None	101 ms
6	2	None	**15.4 min**	None	**84 ms**
6	3	None	35.5 min	None	28.1 min

a system with only 11 states and 21 transitions, which is depicted in Fig. 7. Not only this is a massive reduction which greatly simplifies the scheduling problem, it also informs the user that refining the traffic model by increasing l from 1 to 2 is *irrelevant when it comes to schedulability*. As Fig. 7 suggests, these traffic models reduce to a single task with recurring deadline of five steps, after the initial phase. Only with $l = 3$ more complex behavior can be enforced by the scheduler, which becomes apparent by the fact that the minimization is not so impactfull: 14% in states and 12% in transitions. This is to be expected because the original systems we abstract are deterministic, and higher values of l reduce the nondeterminism of the abstraction, giving less room for transition elimination in our algorithm. In all cases, the CPU times are within seconds, with an approximately quadratic dependence on the size of the original system. It is worth noting that our Python implementation uses the naive fixed-point algorithm to get the MAS relation, and this step dominated the CPU time of the reduction. Since the times were satisfactory, no performance optimizations were attempted.

Because of the refinement properties of the abstractions $\mathcal{S}_l$ (namely $\mathcal{S}_{l+1} \preceq_{AS} \mathcal{S}_l \preceq \mathcal{S}_{l+1}$), scheduling with these abstractions is sound but not complete: if p ETC plants are detected to be unschedulable for l, one may still find a schedule using a higher value of l. Thus, we employed the following scenario: first, set $l = 1$ and $p = 2$ and increase p until the systems are unschedulable; then, increase l and try again. We used the ETCetera tool [14] to solve the scheduling problem, which has the functionality to create the traffic models $\mathcal{S}_l$, perform the parallel composition, and solve the safety game: always avoid a state whose output contains more than one T. Our first attempt used a Python implementation of the composition and safety game solution, where the transitions are encoded with dictionaries. Without minimization and with $p = 2$, the scheduling problem took only 801 ms to be concluded, a number close to the 657 ms taken to minimize each system; this is expected, given the quadratic complexity of the minimization algorithm. However, with only $p = 3$ *the scheduling problem without reduction crashed due to memory overflow.*[7] *After performing the minimization, we were able to compute a scheduler for* $p = 5$, a process that took 28.7 min to conclude. With $p = 6$, memory overflow also occurred with the minimal systems. Our second attempt to solve the scheduling problem used BDDs to encode the transition systems. Table 2 summarizes the results of this experiment. As expected, for all cases in $l \in \{1, 2\}$ the problem was solved significantly faster with the minimized systems. The difference is much more significant in the non-schedulable cases, which is to be expected because it often requires more iterations in the fixed-point algorithm to detect that no schedule is viable. The difference is particularly massive for $l = 2$, owing to the immense reduction of the system dimensions in this case. For the case with $l = 3$ the time reduction was not as significant as in the aforementioned cases, which is in par with the smaller system size reduction that was obtained in this case.

5 CONCLUSION AND FUTURE WORK

We have revisited the notion of alternating simulation equivalence, and argued about the benefits it can bring for size reduction of finite transition systems in the context of controller synthesis. An algorithm was devised to produce minimal abstractions modulo alternating simulation equivalence. The applicability of these theoretical developments was then illustrated in the context of scheduling, providing interesting insights for the analysis of schedulability of event triggered systems.

This work opens the door to several further investigations, in particular: (i) extending the ASE notion to weighted transition systems to produce abstractions preserving quantitative properties; (ii) extensions of these same ideas to timed games; (iii) designing on-the-fly versions of the proposed reduction algorithm; and (iv) implementing symbolically the abstraction algorithm employing binary decision diagrams.

ACKNOWLEDGMENTS

This work is supported by the European Research Council through the SENTIENT project, Grant No. ERC-2017-STG #755953 (https://cordis.europa.eu/project/id/755953).

REFERENCES

[1] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A Henzinger, P-H Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. 1995. The algorithmic analysis of hybrid systems. *Theoretical computer science* 138, 1 (1995), 3–34.

[7] The experiments were performed in a Intel(R) Xeon(R) W-2145 CPU @ 3.70GHz with 31 GB RAM.

[2] Rajeev Alur, Thomas A Henzinger, Orna Kupferman, and Moshe Y Vardi. 1998. Alternating refinement relations. In *International Conference on Concurrency Theory*. Springer, 163–178.

[3] Adolfo Anta and Paulo Tabuada. 2008. Self-triggered stabilization of homogeneous control systems. In *American Control Conference, 2008.* IEEE, 4129–4134.

[4] Karl Johan Åström and Bo Bernhardsson. 2002. Comparison of Riemann and Lebesgue sampling for first order stochastic systems. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002*, Vol. 2. IEEE, 2011–2016.

[5] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking*. MIT press.

[6] Jan A Bergstra, Alban Ponse, and Scott A Smolka. 2001. *Handbook of process algebra*. Elsevier.

[7] Doron Bustan and Orna Grumberg. 2003. Simulation-based minimization. *ACM Transactions on Computational Logic (TOCL)* 4, 2 (2003), 181–206.

[8] Christos G Cassandras, Stephane Lafortune, et al. 2008. *Introduction to discrete event systems*. Vol. 2. Springer.

[9] Krishnendu Chatterjee, Siddhesh Chaubal, and Pritish Kamath. 2012. Faster Algorithms for Alternating Refinement Relations. *Computer Science Logic 2012* (2012), 167.

[10] Rance Cleaveland and Oleg Sokolsky. 2001. Equivalence and preorder checking for finite-state systems. *Handbook of Process Algebra* (2001), 391–424.

[11] Gabriel de A. Gleizer, Khushraj Madnani, and Manuel Mazo Jr. 2021. Self-Triggered Control for Near-Maximal Average Inter-Sample Time. In *60th IEEE Conference on Decision and Control (accepted)*.

[12] Gabriel de A. Gleizer and Manuel Mazo Jr. 2020. Scalable traffic models for scheduling of linear periodic event-triggered controllers. *IFAC-PapersOnLine* 53, 2 (2020), 2726–2732.

[13] Gabriel de A. Gleizer and M. Mazo Jr. 2021. Computing the sampling performance of event-triggered control. In *Proc. of the 24th Int'l Conf. on Hybrid Systems: Computation and Control* (Nashville, TN, USA) *(HSCC '21)*. ACM, Article 20, 7 pages.

[14] Giannis Delimpaltadakis, Gabriel de A. Gleizer, Ivo van Stralen, and M. Mazo Jr. 2022. ETCetera: beyond Event-Triggered Control. In *Proc. of the 25th Int'l Conf. on Hybrid Systems: Computation and Control* (Milano, Italy) *(HSCC '22)*. ACM, 11 pages.

[15] Giannis Delimpaltadakis and Manuel Mazo Jr. 2021. Abstracting the Traffic of Nonlinear Event-Triggered Control Systems. *arXiv preprint arXiv:2010.12341, under review* (2021).

[16] Carsten Fritz and Thomas Wilke. 2002. State space reductions for alternating büchi automata quotienting by simulation equivalences. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer, 157–168.

[17] WPMH Heemels, Karl Henrik Johansson, and Paulo Tabuada. 2012. An introduction to event-triggered and self-triggered control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 3270–3285.

[18] W. P. M. H. Heemels, M. C. F. Donkers, and Andrew R. Teel. 2013. Periodic event-triggered control for linear systems. *IEEE Trans. Automat. Control* 58, 4 (2013), 847–861.

[19] Monika Rauch Henzinger, Thomas A Henzinger, and Peter W Kopke. 1995. Computing simulations on finite and infinite graphs. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE, 453–462.

[20] Thomas A Henzinger and Peter W Kopke. 1995. *Undecidability results for hybrid systems*. Technical Report. Cornell University.

[21] Thomas A Henzinger, Rupak Majumdar, and Jean-François Raskin. 2005. A classification of symbolic transition systems. *ACM Transactions on Computational Logic (TOCL)* 6, 1 (2005), 1–32.

[22] Paris C Kanellakis and Scott A Smolka. 1990. CCS expressions, finite state processes, and three problems of equivalence. *Information and computation* 86, 1 (1990), 43–68.

[23] Arman Sharifi Kolarijani and Manuel Mazo Jr. 2015. Traffic Characterization of LTI Event-triggered Control Systems: a Formal Approach. *arXiv preprint arXiv:1503.05816* (2015).

[24] David Lee and Mihalis Yannakakis. 1992. Online minimization of transition systems. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*. 264–274.

[25] Rupak Majumdar. 2003. *Symbolic algorithms for verification and control*. University of California, Berkeley.

[26] Manuel Mazo Jr., Adolfo Anta, and Paulo Tabuada. 2010. An ISS self-triggered implementation of linear controllers. *Automatica* 46, 8 (2010), 1310–1314.

[27] M Mazo Jr, A Sharifi Kolarijani, D Adzkiya, and C Hop. 2018. Abstracted Models for Scheduling of Event-Triggered Control Data Traffic. In *Control Subject to Computational and Communication Constraints*. Springer, 197–217.

[28] Robin Milner. 1971. An algebraic definition of simulation between programs. In *Proceedings of the 2nd international joint conference on Artificial intelligence*. 481–489.

[29] Peter JG Ramadge and W Murray Wonham. 1989. The control of discrete event systems. *Proc. IEEE* 77, 1 (1989), 81–98.

[30] Francesco Ranzato and Francesco Tapparo. 2007. A New Efficient Simulation Equivalence Algorithm. In *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*. 171–180. https://doi.org/10.1109/LICS.2007.8

[31] Paulo Tabuada. 2007. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Trans. Automat. Control* 52, 9 (2007), 1680–1685.

[32] Paulo Tabuada. 2009. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media.

[33] Manel Velasco, Josep Fuertes, and Pau Marti. 2003. The self triggered task model for real-time control systems. In *Work-in-Progress Session of the 24th IEEE Real-Time Systems Symposium (RTSS03)*, Vol. 384.