Summary of Supervisory Control Synthesis of Timed Automata Using Forcible Events*

M.A. Reniers

Eindhoven University of Technology, Netherlands

Abstract

This is a summary of [1], which offers an approach to synthesize safe supervisors for systems specified by means of Timed Automata.

Keywords

Discrete event systems, Supervisory control synthesis, Tmed Automata

1. Introduction

Supervisory control theory (SCT) was first introduced by Ramadge-Wonham to control discrete-event systems (DES) [2]. SCT provides a synthesis method resulting in a supervisor that restricts the plant behavior towards a given desired behavior. The synthesized supervisor additionally satisfies controllability, nonblockingness, and maximal permissiveness [3].

DES, such as communication networks, manufacturing and traffic systems, are typically modeled using extended finite automata (EFA) [4], which are finite automata extended with discrete variables. In EFA, transitions are labeled by events and associated with constraints on variables (guards), where variables may be updated after the occurrence of an event [4]. The dynamics of DES depend entirely on the ordering of the event occurrences, and so are independent of time [5]. Due to lack of timing information, modeling systems as EFA is not suitable for model checking and verification of real-time systems [6]. To make this possible, among others the concept of timed automata (TA) has been introduced [6]. A TA consists of a finite set of locations and a finite set of real-valued clocks [7]. A clock constraint, called an *invariant*, is associated to each location and determines the maximal time that the system is allowed to stay in that location. Each edge between two locations is labeled by an event, a clock constraint called the *guard*, and a set of clocks that are reset to zero, called the *reset*, upon an occurrence of that event. The formalism of TA and accompanying analysis methods have originally been introduced for modelling timed systems and model checking temporal logic

OVERLAY 2023: 5th Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, November XX, 2023, Rome, Italy

* This short paper present an overview of the paper with the same title published as [1]. IT IS NOT OUR INTENTION TO PUBLISH THIS PAPER, BUT MERELY TO PRESENT IT IN THE WORKSHOP AND TO DISCUSS THE RELEVANCE FOR SYSTEMS IN WHICH AN AI-ENGINE PLAYS A ROLE.

m.a.reniers@tue.nl (M.A. Reniers)
https://research.tue.nl/en/persons/michel-a-reniers (M.A. Reniers)
0000-0002-9283-4074 (M.A. Reniers)

properties for those [6]. In general, the control of TA is challenging due to the clock variables, making the state space of TA infinite. In [1], supervisor synthesis for TA is provided such that:

- an algorithm is proposed that works with automata instead of languages to ease integration of an implementation in a tool set such as CIF or Supremica [8, 9],
- the synthesized supervisor is maximally permissive, controllable, and nonblocking,
- forcible events (i.e., events which the supervisor can force to occur (if the plant allows the event) from TDES [10] are used to allow a supervisor to preempt time progress, and
- the notion of clock regions of timed automata is adapted in a specific way for supervisory control synthesis (details are available in [1]).

The synthesis of maximally permissive supervisors is especially relevant in cases where an AI-engine is assumed to assist in making decisions about the future behaviour of a plant. Putting a synthesized maximally permissive supervisor in between the plant and the AI-engine guarantees that the AI-engine may only invoke safe commands to the plant.

2. Overview of the synthesis problem and the synthesis method

In supervisory control we use a model of the uncontrolled system (and a model of the requirements the controlled system should satisfy). We use deterministic TA for both purposes. Informally speaking, a TA is an FA extended with a finite set of real-valued clocks. To model the timing behavior of TA, the temporal conditions to switch between different modes (locations) or to stay in the current one are represented by clock constraints [6, 11]. Clock constraints are of the form $x \sim n$ or $x - y \sim n$ for clocks $x, y \in C$, natural number $n \in \mathbb{N}$ and comparison operator $\sim \in \{<, =, >\}$ and may be combined with conjunction ($\wedge$) and disjunction ($\vee$). Invariants of locations are indicated inside the locations. Absence of an invariant in a location represents the invariant that always holds. The initial location is depicted by a dangling incoming arrow, and the marked locations by double circles.

Applications are typically modeled by a network of automata, where each automaton represents a single component or subsystem. A single automaton representing the network of automata can then be generated as the synchronous product of the constituent automata. If two TA share a clock variable, then this clock variable can only be reset if both TA do so. A consequence of this is that such a reset can only occur on an edge with a shared event.

The *Basic TSC Synthesis Problem* is: Given a plant model G as a TA, synthesize a timed supervisor S as a TA, such that

- S is controllable w.r.t. G, which means that the supervisor may not disable any uncontrollable events from occurring in the plant,
- $S||G$ is nonblocking, i.e., from any reachable state a marked state should be reachable,
- S is maximally permissive, which requires that the supervisor restricts the behaviour of the plant as little as possible in order to achieve controllability and nonblocking.

In [1], an algorithm is proposed that computes a solution to this synthesis problem. It is also shown that this algorithm terminates, and results in a supervisor that has the aforementioned properties. A high-level overview of the operation of this algorithm is given in Figure 1.

The algorithm iteratively strengthens the invariants and guards of the plant automaton in order make blocking states unreachable while respecting controllability (i.e., the guards of uncontrollable events are never strengthened). Both the nonblocking condition and bad-state condition computations are similar to those used in synthesis algorithms for EFA [12].

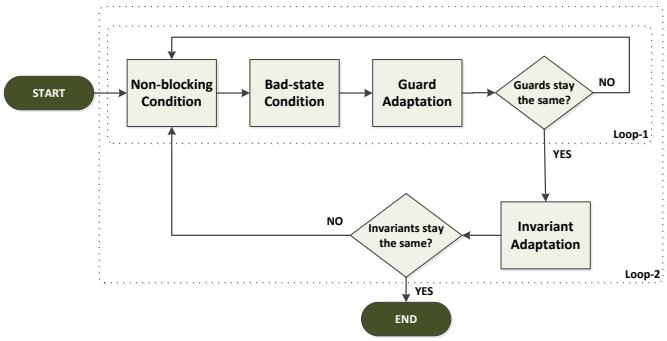

Figure 1: Overview of the synthesis algorithm.

The synthesis procedure has two loops. In Loop-1 the guards are adapted to obtain a supervisor that prevents reaching the bad states by adapting the guard of each edge labeled by a controllable event. The effect of forcible events preempting time events is taken into account in the invariant adaptation (Loop-2). The invariant of a location can be changed only if there exists an edge labeled by a forcible event starting from that location.

In [1], also the problem where (safety) requirements are used is stated and solved. Such requirements are represented by timed automata as well (and do not share clocks with the plant). Intuitively speaking a supervisor achieves safety in case the controlled system behaviour is restricted to the behaviour specified by the requirements (i.e., does not violate the requirements). The solution relies on transforming the problem statement with safety requirements into the setting without. A detailed explanation of the transformation is provided in [1].

3. Train-gate controller

In this section, we consider the verification example from [6, 13] and modify it for synthesis. The TA representing the train and gate are taken from [6, 13] and are depicted in Figure 2. The events `approach` and `out` for the train, and the events `down` and `up` for the gate are assumed to be uncontrollable. Moreover, the events `raiseE` and `lowerE` of the gate are assumed to be forcible. The system in [6, 13] also involves an automatic controller, to open and close the gate in a railroad crossing. The control requirements are as follows [6]:

- Safety: if the train is inside the gate, the gate is closed.
- Liveness: the gate is never closed more than 10 time units.

In [6, 13], the system is assessed to be safe by analysing the timing constraints: they say that with the gate-controller, that is part of the system, the event `down` always precedes the event `in`, so the system is always safe. We do not consider such a controller to already be given. We

synthesize a supervisor that is correct-by-construction, and more importantly this supervisor guarantees controllability, nonblockingness, and maximal permissiveness.

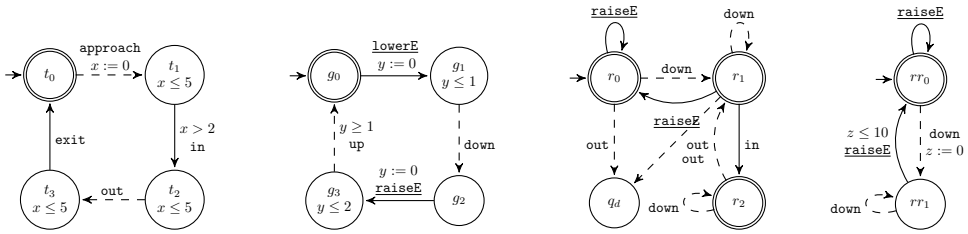

Figure 2: Train automaton and gate automaton. Safety and liveness requirements for train-gate system. Locations are represented by circles and the edges by arrows from the source location to the target location, labelled with the event, the guard and the reset. Edges representing uncontrollable events are dashed. Forcible events are underlined. The reset of a clock x is denoted by $x := 0$.

The safety and liveness requirements are represented by the two rightmost TA in Figure 2. The supervisor synthesized by applying the synthesis algorithm from [1] is given in Figure 3. In this figure, the synchronous product of the train-gate and control requirements is indicated in black and the adaptations made by the supervisor in red. The supervisor obtained through synthesis allows more behavior than the (manually constructed) controller given in [6, 13]. For instance, in their controller the event `lowerE` is not enabled before the train approaches, while this is allowed by the synthesized supervisor.

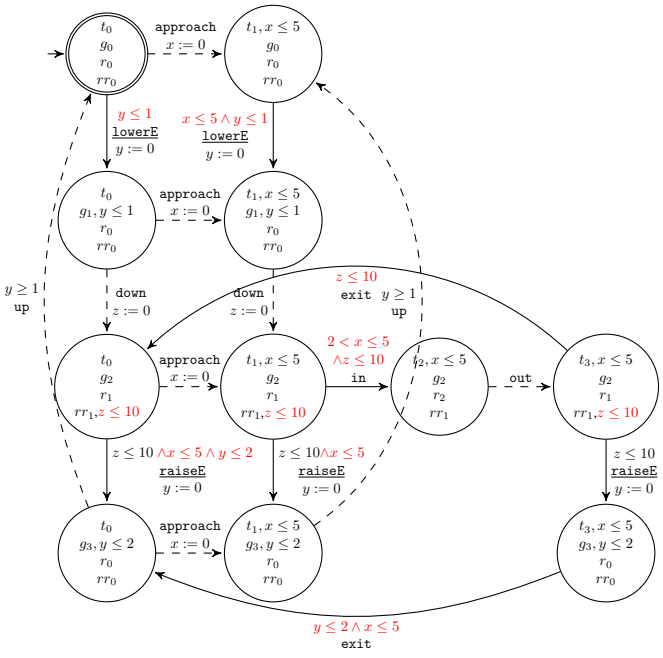

Figure 3: Synthesized supervisor for train-gate system.

References

[1] A. Rashidinejad, M. Reniers, M. Fabian, Supervisory control synthesis of timed automata using forcible events, IEEE Transactions on Automatic Control (2023) 1–8. doi:10.1109/TAC.2023.3275440.

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

[3] W. M. Wonham, Supervisory control of discrete-event systems, Encyclopedia of systems and control (2015) 1396–1404.

[4] M. Skoldstam, K. Åkesson, M. Fabian, Modeling of discrete event systems using finite automata with variables, in: 2007 46th IEEE Conference on Decision and Control, IEEE, 2007, pp. 3387–3392.

[5] C. G. Cassandras, S. Lafortune, Introduction to discrete event systems, Springer Science & Business Media, 2009.

[6] R. Alur, D. L. Dill, A theory of timed automata, Theoretical computer science 126 (1994) 183–235.

[7] A. Dubey, A discussion on supervisory control theory in real-time discrete event systems, ISIS 9 (2009) 112.

[8] D. A. van Beek, W. Fokkink, D. Hendriks, A. Hofkamp, J. Markovski, J. Van De Mortel-Fronczak, M. A. Reniers, CIF 3: Model-based engineering of supervisory controllers, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2014, pp. 575–580. URL: https://doi.org/10.1007/978-3-642-54862-8_48.

[9] K. Åkesson, M. Fabian, H. Flordal, R. Malik, Supremica–an integrated environment for verification, synthesis and simulation of discrete event systems, in: 2006 8th International Workshop on Discrete Event Systems, IEEE, 2006, pp. 384–385.

[10] B. A. Brandin, W. M. Wonham, Supervisory control of timed discrete-event systems, IEEE Transactions on Automatic Control 39 (1994) 329–342.

[11] J. Bengtsson, W. Yi, Timed Automata: Semantics, Algorithms and Tools, Springer Berlin Heidelberg, 2004, pp. 87–124.

[12] L. Ouedraogo, R. Kumar, R. Malik, K. Åkesson, Nonblocking and safe control of discrete-event systems modeled as extended finite automata, IEEE Transactions on Automation Science and Engineering 8 (2011) 560–569.

[13] R. Alur, Timed automata, in: International Conference on Computer Aided Verification, Springer, 1999, pp. 8–22.