

Dynamic Controllability of Temporal Networks via Supervisory Control

Matteo Zaverteri¹, Davide Bresolin¹, Romeo Rizzi² and Tiziano Villa²

¹Department of Mathematics, University of Padova, Italy

²Department of Computer Science, University of Verona, Italy

Abstract

Temporal networks are expressive formalisms employed in AI to model, validate, and execute temporal plans. The core parts of a temporal network are a finite set of real variables called time points and a finite set of constraints bounding the minimal and/or maximal temporal distance between pairs of time points. When uncontrollable choices are considered, a problem of interest is determining whether or not a network is dynamically controllable. That is, whether there exists a strategy that, based only on the values already assigned to uncontrollable variables, progressively assigns the controllable variables with their final values in such a way that all constraints will be met in the end. Current *single*-strategy synthesis approaches are mainly based on constraint propagation or controller synthesis for timed game automata. In this paper we show how to model a temporal network as a *Discrete Event System* (DES) so as to leverage on Supervisory Control Theory to synthesize all dynamic integer strategies within a finite time horizon.

Keywords

AI, formal methods, temporal network, discrete event system, supervisory control

1. Temporal Networks

In the Artificial Intelligence community, temporal networks are a framework to model temporal plans and check the consistency of temporal constraints imposing delays and deadlines between the occurrences of events in the plan [1]. Over the years the core formalism of *Simple Temporal Networks* [1] has been extended in several ways to cope with uncontrollable durations, (un)controllable choices, disjunctive and conditional constraints, see for example [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. The most expressive formalisms of temporal networks are those that simultaneously handle all such features. In this paper we stick with the restriction of CDT-NUs [5] (or CTNUds [11]) that only consider uncontrollable choices and conditional disjunctive constraints over the same pair of time points. We address the dynamic controllability problem from a maximally-permissive point of view. That is, we synthesize all strategies that correctly assign integer values to the controllable variables (within a finite time horizon) only depending on the already observed values assigned to the uncontrollable ones.

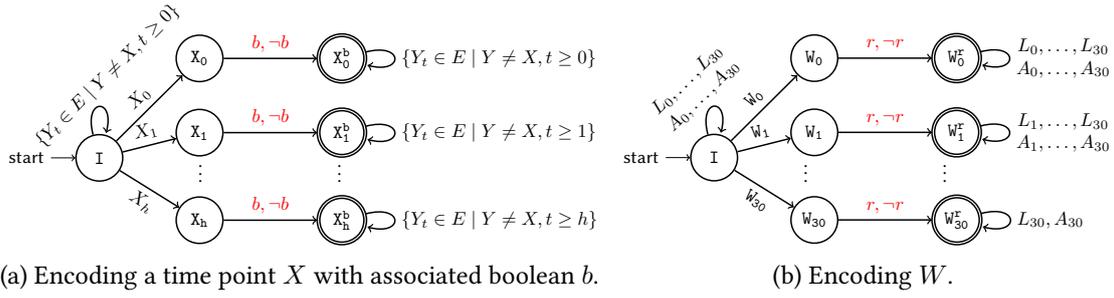
OVERLAY 2022: 4th Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, November 28, 2022, Udine, Italy

✉ matteo.zaverteri@unipd.it (M. Zaverteri); davide.bresolin@unipd.it (D. Bresolin); romeo.rizzi@univr.it (R. Rizzi); tiziano.villa@univr.it (T. Villa)



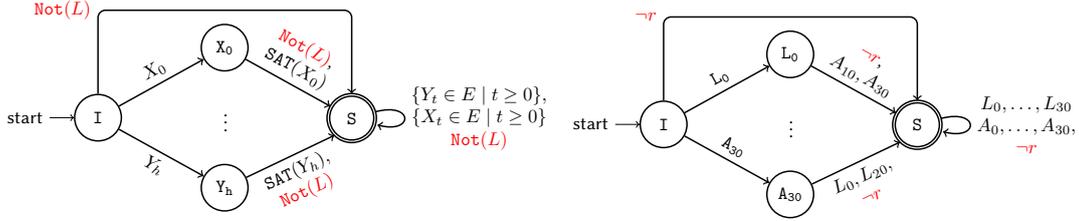
© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)



(a) Encoding a time point X with associated boolean b .

(b) Encoding W .



(c) Encoding a constraint $L \Rightarrow Y - X \in \bigcup_{i=1}^n [\ell_i, u_i]$. (d) Encoding $r \Rightarrow A - L \in [10, 10] \cup [30, 30]$.

Figure 2: Encoding Temporal Networks into Plant and Requirement Automata. Let $L \Rightarrow Y - X \in \bigcup_{i=1}^n [\ell_i, u_i]$. Then, $\text{SAT}(X_t) := \{Y_{t'} \in E \mid t' - t \in \bigcup_{i=1}^n [\ell_i, u_i]\}$, $\text{SAT}(Y_t)$ is defined symmetrically, $\text{Not}(L) := \{b_F \mid b \in L\} \cup \{b_T \mid \neg b \in L\}$.

either controllable or uncontrollable. When working with regular languages, we can model DESs by means of finite state automata. The control synthesis problem takes as input a set of automata whose concurrent behavior models the plant and another set of automata modeling requirements and it tries to build a controller that is maximally-permissive and non-blocking. Roughly speaking, the former means that all executions of the plant that are legal must not be blocked, whereas the latter means that no execution of the plant gets to a point at which it cannot “complete” (i.e., reach a marked state).

In the case of finite state automata, the synthesis algorithm starts from the parallel composition of the plant automata with the requirement automata and then removes states if they disable uncontrollable events or if there is no path toward a marked state. The algorithm ends when there are no more states to remove. Eventually, either all states are removed (and thus we have no supervisor) or we get a maximally, non-blocking supervisor.

3. Dynamic Controllability via Supervisory Control

Let $N := (\mathcal{T}, \mathcal{B}, \mathcal{I}, \mathcal{C})$ be a temporal network and let h be a finite time horizon. That is, a number “big enough” to guarantee that if N is dynamically controllable, then there exists some strategy that always schedules all events within h . In our example, $h = 30$ suffices because of the constraint $A - W \in [-\infty, 30]$. Also, when all numbers in the instance are integers, the network is dynamically controllable, and we assume instantaneous reaction semantics, then there exist integer strategies. We start from a set of events E that contains $h + 1$ controllable events X_0, \dots, X_h for each $X \in \mathcal{T}$, and two uncontrollable events $b, \neg b$ for each $b \in \mathcal{B}$. An

event X_t means that time point X is executed at time t . An event b (resp., $\neg b$) means that the boolean b has been observed true (resp., false).

The first thing that we need to do is to create the plant that guarantees the followings: (1) each time point is executed exactly once, (2) if a time point X has a boolean associated, then the boolean is assigned a truth value exactly once upon the execution of X , and (3) the time assigned to a time point is monotone non-decreasing (w.r.t. the time assigned to the already executed time points). To achieve this purpose, we create a plant automaton for each time point $X \in \mathcal{T}$ as shown in Figure 2a. Self loops at state I (initial state) allow the execution of other time points before the execution of X . The execution of X at time $t \in \{0, \dots, h\}$ happens by taking a transition X_t leading to the state X_t . If X has also a boolean b associated (i.e., $I(b) = X$), then the assignment of a truth value to b occurs by executing one of the events $b, \neg b$ leading to the marked state X_t^b . If X does not have any boolean associated, then the automaton lacks the transitions labeled by $b/\neg b$ as well as the state X_t^b , but in that case it is X_t to be marked. At marked states (X_t or X_t^b), the execution of all other time points is restricted to occur at a time $\geq t$ (self loops at marked states). The concurrent behavior of all these automata meets (1), (2), and (3) described above. To give an example, Figure 2b provides the encoding of the time point W of Figure 1.

The second thing that we need to do is to create the requirement automata. To achieve this purpose, we create a requirement automaton for each constraint $L \Rightarrow X - Y \in \bigcup_{i=1}^n [\ell_i, u_i]$ in \mathcal{C} . Figure 2c shows such encoding. Basically, the automaton is synchronized with the execution of X_t, Y_t (for $t \in \{0, \dots, h\}$) and the observation of the truth values of booleans in L . The states of the automaton handle all possible valid executions of these events. At any state X_t a set of transitions leading to S are labeled by events $Y_{t'}$ resembling the executions of Y that satisfy the constraint (event set $\text{SAT}(X_t)$). Symmetrically, for any state Y_t transitions to S are labeled with events from $\text{SAT}(Y_t)$. Also, from any state it is always possible to reach S with a boolean event that makes L false (event set $\text{Not}(L)$). Finally, at S there are self loop transitions to avoid blocking the execution once the constraint is satisfied. This way, all executions not satisfying the constraint will get stuck in some state different from S. Figure 2d shows the encoding of $r \Rightarrow A - L \in [10, 10] \cup [30, 30]$ of Figure 1. For example, if we could leave at 0 (state L_0), then either it does not rain and thus the constraint does not apply leading immediately to S, or we would need to arrive at work at time 10 or 30 in order to satisfy the constraint.

Finally, if we run the synthesis algorithm with these plant and requirement automata we can obtain a supervisor if and only if the temporal network is dynamically controllable.

4. Conclusions and Future Work

This paper places dynamic controllability of temporal networks at the intersection of Formal Methods and Artificial Intelligence, by modeling temporal networks as discrete event systems, so we leverage 40-more years of Supervisory Control to solve dynamic controllability in a maximally-permissive way. In the case of the example in Figure 1 the supervisor has 399 states and 398 transitions (52 states and 228 transitions, if minimized). As future work we plan to explore variations of classic supervisory control (e.g., [13, 14]) in order to reduce time and space complexity of the synthesis.

Acknowledgments

This work was partially supported by MIUR, Project *Italian Outstanding Departments, 2018-2022*, by INdAM, GNCS 2022, Project *Elaborazione del Linguaggio Naturale e Logica Temporale per la Formalizzazione di Testi*, and by the SID/BIRD project *Deep Graph Memory Networks*, Department of Mathematics, University of Padova.

References

- [1] R. Dechter, I. Meiri, J. Pearl, Temporal constraint networks, *Artif. Intell.* 49 (1991) 61–95.
- [2] T. Vidal, H. Fargier, Handling contingency in temporal constraint networks: from consistency to controllabilities, *Jour. of Exp. & Theor. Artif. Intell.* 11 (1999) 23–45.
- [3] P. R. Conrad, B. C. Williams, Drake: An efficient executive for temporal plans with choice, *JAIR* 42 (2011) 607–659.
- [4] I. Tsamardinos, T. Vidal, M. E. Pollack, CTP: A new constraint-based formalism for conditional, temporal planning, *Constraints* 8 (2003) 365–388.
- [5] A. Cimatti, L. Hunsberger, A. Micheli, R. Posenato, M. Roveri, Dynamic controllability via timed game automata, *Acta Informatica* 53 (2016) 681–722.
- [6] E. Karpas, S. J. Levine, P. Yu, B. C. Williams, Robust execution of plans for human-robot teams, in: *ICAPS '15*, AAAI Press, 2015, pp. 342–346.
- [7] S. J. Levine, B. C. Williams, Concurrent plan recognition and execution for human-robot teams, in: *ICAPS '14*, AAAI, 2014, pp. 490–498.
- [8] P. Yu, C. Fang, B. C. Williams, Resolving uncontrollable conditional temporal problems using continuous relaxations, in: *ICAPS '14*, AAAI, 2014, pp. 341–349.
- [9] M. Zaverterri, Conditional simple temporal networks with uncertainty and decisions, in: *TIME 2017*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017, pp. 23:1–23:17.
- [10] M. Zaverterri, L. Viganò, Conditional simple temporal networks with uncertainty and decisions, *Theoretical Computer Science* 797 (2019) 77–101.
- [11] M. Zaverterri, R. Rizzi, T. Villa, Dynamic controllability of temporal networks with instantaneous reaction, *Information Sciences* (2022). doi:<https://doi.org/10.1016/j.ins.2022.08.099>.
- [12] P. J. Ramadge, W. M. Wonham, Supervisory control of a class of discrete event processes, *SIAM Journal on Control and Optimization* 25 (1987) 206–230. doi:10.1137/0325013.
- [13] C. Ma, W. M. Wonham, Nonblocking supervisory control of state tree structures, *IEEE Transactions on Automatic Control* 51 (2006) 782–793.
- [14] L. Ouedraogo, R. Kumar, R. Malik, K. Akesson, Nonblocking and safe control of discrete-event systems modeled as extended finite automata, *IEEE Transactions on Automation Science and Engineering* 8 (2011) 560–569. doi:10.1109/TASE.2011.2124457.