Conditional Simple Temporal Networks with Uncertainty and Decisions*

Matteo Zavatteri

Department of Computer Science, University of Verona, Verona, Italy
matteo.zavatteri@univr.it

Abstract

A conditional simple temporal network with uncertainty (CSTNU) is a framework able to model temporal plans subject to both conditional constraints and uncertain durations. The combination of these two characteristics represents the uncontrollable part of the network. That is, before the network starts executing, we do not know completely which time points and constraints will be taken into consideration nor how long the uncertain durations will last. Dynamic controllability (DC) implies the existence of a strategy scheduling the time points of the network in real time depending on how the uncontrollable part behaves. Despite all this, CSTNUs fail to model temporal plans in which a few conditional constraints are under control and may therefore influence (or be influenced by) the uncontrollable part. To bridge this gap, this paper proposes *conditional simple temporal networks with uncertainty and decisions (CSTNUDs)* which introduce *decision time points* into the specification in order to operate on this conditional part under control. We model the dynamic controllability checking (DC-checking) of a CSTNUD as a two-player game in which each player makes his moves in his turn at a specific time instant. We give an encoding into timed game automata for a sound and complete DC-checking. We also synthesize memoryless execution strategies for CSTNUDs proved to be DC. The proposed approach is fully automated.

1998 ACM Subject Classification G.2.2 Graph Theory, F.1.1 Models of Computation, I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases cstnud, dynamic controllability, timed game automata, controller synthesis, advanced temporal planning under uncertainty.

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.23

1 Introduction

Temporal networks are a framework to model temporal plans and check the coherence of their temporal constraints which impose a minimal and maximal temporal distance between the occurrence of the events specified in the plan. Temporal plans mainly divide in plans having everything under control and plans having something out of control. The main components of a temporal network are *time points* and *constraints*. Time points are variables having continuous domain and model the occurrence of events as soon as these variables are assigned real values (i.e., executed). Constraints regulate the minimal and maximal temporal distance between the occurrence of pairs of events and are formalized as linear inequalities.

Whenever both these two components are under control we simply deal with a *consistency* problem asking us to find an assignment of real values to *all* time points satisfying all constraints. Simple temporal networks (STNs) model exactly this case [10], whereas Drake [9]

* The author wants to thank Luca Viganò for his valuable feedback.

addresses temporal plans with choices that are, however, under control; therefore, we keep dealing with a consistency problem asking us to further find suitable values for such choices.

Instead, when some component is out of control, satisfiability is, in general, not enough. In such a case, we deal with a *controllability* problem.

Conditional simple temporal networks (CSTNs) [14, 20] address conditional constraints to enable or disable some parts of the network (i.e., a subset of time points and constraints) during execution. Conditionals are expressed as labels consisting of conjunctions of literals whose atoms are Boolean propositions. The truth value assignments to such propositions are out of control and depend on the behavior of unpredictable external events which are only observed to occur while executing the network.

Simple temporal networks with uncertainty (STNUs) [17, 18] address uncertain (but bounded) durations. Such durations are modeled by contingent links, i.e. pairs of distinct time points specifying a range of allowed values between their distance. One of these time points is called *activation* and it is under control, whereas the other one is called *contingent* and it is not. The real value assignment to the contingent one depends again on the behavior of unpredictable external events which are only observed to occur while executing the network.

Conditional simple temporal networks with uncertainty (CSTNUs) [7, 13] merge the semantics of CSTNs and STNUs addressing conditional constraints and uncertain durations.

Controllability of a temporal network implies the existence of a *strategy* operating on the controllable part such that all constraints will eventually be satisfied. Controllability mainly divides in *weak*, *strong* and *dynamic*. Weak controllability ensures the existence of a (possible different) strategy to operate on the controllable part whenever we are able to predict how the entire uncontrollable part will behave before the execution starts. Strong controllability is the opposite case ensuring the existence of a strategy operating always the same way on the controllable part no matter how the uncontrollable part will behave. However, strong controllability is "too strong". If a temporal network is not strongly controllable, it could still be executable by operating on the controllable part reacting to the uncontrollable one as soon as it becomes known. Dynamic controllability addresses exactly this case.

However, none of the formalisms mentioned so far tackles temporal plans in which some conditional constraints under control may influence (or be influenced by) some uncontrollable part. An initial discussion is given in [3] where CSTNs are extended with decision nodes regulating the truth value assignments to some propositions under control.

We give here the first attempt to address temporal plans in which decisions may influence (or be influenced by) *both* conditional and temporal uncertainty.

Toward this aim our contributions are three-fold. First, we define *conditional simple temporal networks with uncertainty and decisions* (CSTNUDs) as a unified formalism for temporal networks expressing uncontrollable parts and model dynamic controllability as a two-player game in which players make moves in their turns. Second, we provide an encoding into timed game automata for a sound and complete DC-checking and synthesize execution strategies by means of the UPPAAL-TIGA software [2]. Third, we automate our approach by discussing a proof of concept tool we came up with.

The rest of the paper is organized as follows. Section 2 provides essential background on CSTNUs, timed game automata (TGAs) and the DC-checking of CSTNUs via TGAs. Section 3 introduces our main contribution: *CSTNUs with Decisions* along with a new semantics given in *move-based strategies*. Section 4 extends the encoding given in Section 2 to address the DC-checking of CSTNUDs. Section 5 discusses our tool and a preliminary experimental evaluation. Section 6 discusses the correctness and complexity of the encoding. Section 7 discusses related work. Section 8 draws conclusions and discusses future work.

2 Background: CSTNUs, TGAs and Dynamic Controllability

2.1 Conditional Simple Temporal Networks with Uncertainty

Given a set $\mathcal{P}$ of Boolean propositions, a *label* $\ell = \lambda_1 \dots \lambda_n$ is any finite conjunction of literals λ_i, where a literal is either a proposition p (positive literal) or its negation $\neg p$ (negative literal). The *empty label* is denoted by $\boxdot$. The *label universe of* $\mathcal{P}$, denoted by $\mathcal{P}^*$, is the set of all possible (consistent) labels drawn from $\mathcal{P}$; e.g., if $\mathcal{P} = \{p, q\}$, then $\mathcal{P}^* = \{\boxdot, p, q, \neg p, \neg q, p \wedge q, p \wedge \neg q, \neg p \wedge q, \neg p \wedge \neg q\}$. Two labels $\ell_1, \ell_2 \in \mathcal{P}^*$ are *consistent* if and only if their conjunction $\ell_1 \wedge \ell_2$ is satisfiable. A label ℓ_1 *entails* a label ℓ_2 (written $\ell_1 \Rightarrow \ell_2$) if and only if all literals in ℓ_2 appear in ℓ_1 too (i.e., if ℓ_1 is more *specific* than ℓ_2). A label ℓ_1 *falsifies* a label ℓ_2 iff $\ell_1 \wedge \ell_2$ is inconsistent. For instance, if $\ell_1 = p \wedge \neg q$ and $\ell_2 = p$, then ℓ_1 and ℓ_2 are consistent since $p \wedge \neg q \wedge p$ is satisfiable, and ℓ_1 entails ℓ_2 since $p \wedge \neg q \Rightarrow p$.

▶ **Definition 1** (CSTNU). A *Conditional Simple Temporal Network with Uncertainty* (*CSTNU*) is a tuple $\langle \mathcal{T}, \mathcal{OT}, \mathcal{P}, O, L, \mathcal{L}, \mathcal{C} \rangle$, where:

- $\mathcal{T} = \{X, Y, \dots\}$ is a finite set of *time points* (i.e., variables with continuous domain).
- $\mathcal{OT} \subseteq \mathcal{T} = \{P?, Q?, \dots\}$ is a set of *observation time points.*
- $\mathcal{P} = \{p, q, \dots\}$ is a finite set of Boolean *propositions.*
- $O \colon \mathcal{P} \to \mathcal{OT}$ is a bijection associating a unique $P? \in \mathcal{OT}$ to each $p \in \mathcal{P}$ (i.e., $O(p) = P?$).
- $L \colon \mathcal{T} \to \mathcal{P}^*$ is a function assigning a *label* to each time point $X \in \mathcal{T}$.
- $\mathcal{L}$ is a finite set of *contingent links* (A, x, y, C), where $A, C \in \mathcal{T}$, $0 < x < y < \infty$ $(x, y \in \mathbb{R})$.
- $\mathcal{C}$ is a finite set of *labeled constraints* $(Y - X \leq k, \ell)$, where $X, Y \in \mathcal{T}$, $k \in \mathbb{R} \cup \{\pm\infty\}$ and $\ell \in \mathcal{P}^*$. If $(Y - X \leq k, \ell) \notin \mathcal{C}$ for some $\ell \in \mathcal{P}^*$, then $k = \infty$ (for *that* label).

A CSTNU is *well-defined* if and only if all the following properties hold.

- For each $X \in \mathcal{T}$, if $\lambda \in L(X)$, where $\lambda \in \{p, \neg p\}$, then $L(X) \Rightarrow L(O(p))$ and $(O(p) - X \leq -\epsilon, L(X)) \in \mathcal{C}$ for some $\epsilon > 0$ (*time point label honesty* [14]).
- For any $(A, x, y, C) \in \mathcal{L}$, $A \neq C$ and $L(A) = L(C)$
- For any pair $(A_1, x_1, y_1, C_1), (A_2, x_2, y_2, C_2) \in \mathcal{L}$ if $A_1 \neq A_2$, then $C_1 \neq C_2$.
- For each constraint $(Y - X \leq k, \ell) \in \mathcal{C}$, $\ell \Rightarrow L(Y) \wedge L(X)$ (*constraint label coherence* [14]), and for each literal $\lambda \in \ell$, where $\lambda \in \{p, \neg p\}$, $\ell \Rightarrow L(O(p))$ (*constraint label honesty* [14]).

We execute a time point by assigning it a real value (modeling the occurrence of some temporal event). We execute a CSTNU by executing all relevant non-contingent time points (see below). For any contingent link (A, x, y, C), A is the *activation time point*, whereas C is the *contingent time point*. A is under control, C is not. Once we execute A, we can merely observe the execution of C (by the environment). However, C is guaranteed to occur such that $C - A \in [x, y]$. A contingent link has a unique implicit label given by $\ell = L(A) = L(C)$.

Likewise, an observation time point $P? \in \mathcal{OT}$ is under control, whereas the truth value assignment to its associated Boolean proposition p is not. Once we execute $P?$ we can merely observe such an assignment. Before executing $P?$ the value of p is *unknown*, and after executing $P?$ is either $\top$ (true) or $\bot$ (false). As we execute observation time points, their truth value assignments to the associated propositions generate the *current partial scenario.* That is, a label $\ell_{cps} \in \mathcal{P}^*$ consisting of the conjunction of these literals. Initially, $\ell_{cps} = \boxdot$, and whenever a proposition is assigned a truth value, the resulting literal λ is appended to ℓ_{cps}. Time points and constraints are *relevant* if their labels are not falsified by ℓ_{cps}. Before executing the network all time points and constraints are relevant. If a time point turns irrelevant, we will not execute it. If a constraint does, we will not be obliged to satisfy it.

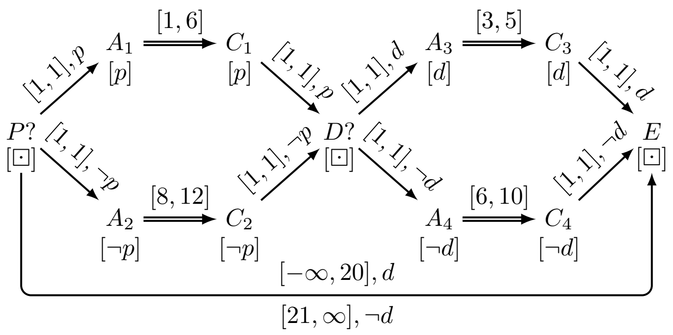


Figure 1 Example of uncontrollable CSTNU.

A CSTNU is said *dynamically controllable* (DC) if there exists a strategy executing all relevant non-contingent time points such that all (relevant) constraints are satisfied no matter which uncertain durations and truth value assignments turn out to be during execution.

We graphically represent a CSTNU as a labeled (multi)graph, where the set of nodes coincides with the set of time points (labels are shown below the nodes), whereas the set of edges divides in *contingent links* and *requirement links*. A contingent link (shown as a double arrow $A \Rightarrow C$ labeled by $[x, y]$) models $(A, x, y, C) \in \mathcal{L}$. A requirement link (shown as a single arrow $X \to Y$ labeled by $[k_1, k_2], \ell$) models the pair $(Y - X \leq k_2, \ell), (X - Y \leq -k_1, \ell) \in \mathcal{C}$.

Figure 1 shows an example of CSTNU having two observation time points $P?, D?$ and four contingent links $(A_1, 1, 6, C_1)$, $(A_2, 8, 12, C_2)$, $(A_3, 3, 5, C_3)$ and $(A_4, 6, 10, C_4)$. $P?$ is the first time point to execute, whereas E is the last. If $P?$ assigns $\top$ to p, then A_2 and C_2 along with the constraints labeled by $\neg p$ turn irrelevant as $\ell_{cps} = p$ falsifies $\neg p$. If $P?$ assigns $\bot$ to p, we will ignore A_1, C_1 and all constraints labeled by p. Likewise, if $D?$ assigns $\top$ (resp., $\bot$) to d, we will ignore A_4 and C_4 (resp., A_3 and C_3) and all constraints labeled by $\neg d$ (resp., d). The CSTNU in Figure 1 is uncontrollable. For example, assume that each contingent time point (if relevant) takes its maximal duration. If $\ell_{cps} = p \wedge \neg d$, then the execution sequence is $P? = 0$, $A_1 = 1$, $C_1 = 7$, $D? = 8$, $A_4 = 9$, $C_4 = 19$ and $E = 20$ (violating $[21, \infty], \neg d$ between $P?$ and E requiring that E is executed from 21 on.). If $\ell_{cps} = \neg p \wedge d$, then the execution sequence is $P? = 0$, $A_2 = 1$, $C_2 = 13$, $D? = 14$, $A_3 = 15$, $C_3 = 20$ and $E = 21$ (violating $[-\infty, 20], d$ between $P?$ and E requiring that E is executed within 20).

2.2 Timed Game Automata

A *timed automaton* (TA) [1] refines a finite automaton [12] by adding real-valued *clocks* and *clock constraints*. All clocks increase at the uniform rate and may by reset many times.

▶ **Definition 2** (TGA). A *Timed Automaton (TA)* is a tuple $\langle Loc, Act, \mathcal{X}, \to, Inv \rangle$, where

- Loc is a finite set of *locations* (one is initial). A location is *urgent* if time freezes in it.
- Act is a finite set of *actions* and $\mathcal{X}$ is a finite set of *real-valued clocks*.
- $\to \subseteq Loc \times \mathcal{H}(\mathcal{X}) \times Act \times 2^{\mathcal{X}} \times Loc$ is the transition relation. An edge (L_i, G, A, R, L_j) represents a transition from L_i to L_j realizing action A. $G \in \mathcal{H}(\mathcal{X})$ is a *guard* consisting of a conjunction of clock constraints having the form $c_1 \sim k$ or $c_1 - c_2 \sim k$ where $c_1, c_2 \in \mathcal{X}$, $k \in \mathbb{N}$ and $\sim \in \{<, \leq, =, >, \geq\}$. $R \subseteq 2^{\mathcal{X}}$ is the set of clocks to reset (i.e., set to 0).
- $Inv : Loc \to \mathcal{H}(\mathcal{X})$ is a function assigning an *invariant* (modeled as a conjunction of clock constraints) to each location. $Inv(L)$ says *when* the TA is allowed to remain in L.

A *Timed Game Automaton (TGA)* [15] extends a TA by dividing transitions into *controllable* and *uncontrollable*. Uncontrollable transitions have priority over controllable ones.

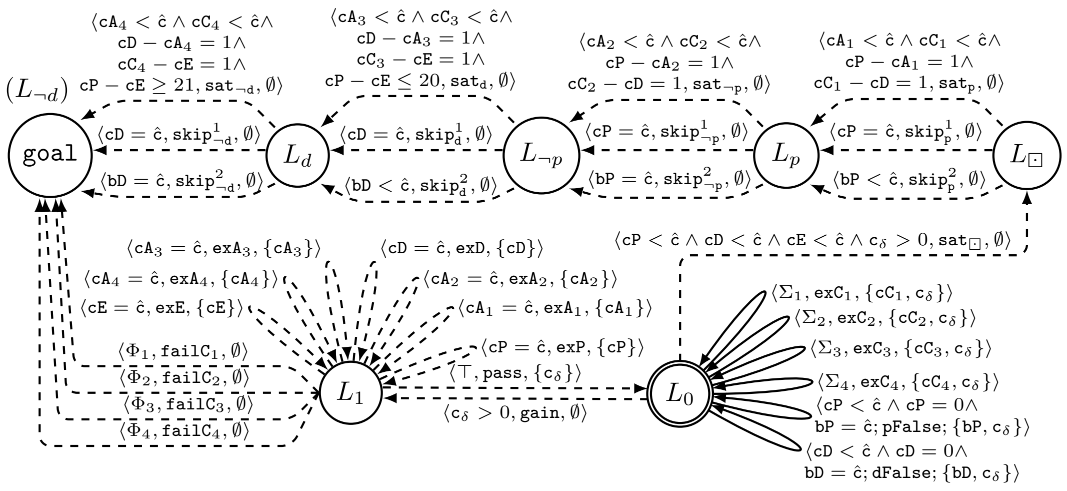


Figure 2 TGA encoding the CSTNU in Fig. 1: L_0 is the initial location, L_1, $L_{\boxdot}$, L_p, $L_{\neg p}$, L_d, `goal` are urgent. Solid (resp., dashed) edges model controllable (resp., uncontrollable) transitions. $\Sigma_1 : \mathtt{cA}_1 < \hat{\mathtt{c}} \wedge \mathtt{cC}_1 = \hat{\mathtt{c}} \wedge \mathtt{cA}_1 \geq 1 \wedge \mathtt{cA}_1 \leq 6$ and $\Phi_1 : \mathtt{cA}_1 < \hat{\mathtt{c}} \wedge \mathtt{cC}_1 = \hat{\mathtt{c}} \wedge \mathtt{cA}_1 > 6$. $\Sigma_2 : \mathtt{cA}_2 < \hat{\mathtt{c}} \wedge \mathtt{cC}_2 = \hat{\mathtt{c}} \wedge \mathtt{cA}_2 \geq 8 \wedge \mathtt{cA}_2 \leq 12$ and $\Phi_2 : \mathtt{cA}_2 < \hat{\mathtt{c}} \wedge \mathtt{cC}_2 = \hat{\mathtt{c}} \wedge \mathtt{cA}_2 > 12$. $\Sigma_3 : \mathtt{cA}_3 < \hat{\mathtt{c}} \wedge \mathtt{cC}_3 = \hat{\mathtt{c}} \wedge \mathtt{cA}_3 \geq 3 \wedge \mathtt{cA}_3 \leq 5$ and $\Phi_3 : \mathtt{cA}_3 < \hat{\mathtt{c}} \wedge \mathtt{cC}_3 = \hat{\mathtt{c}} \wedge \mathtt{cA}_3 > 5$. $\Sigma_4 : \mathtt{cA}_4 < \hat{\mathtt{c}} \wedge \mathtt{cC}_4 = \hat{\mathtt{c}} \wedge \mathtt{cA}_4 \geq 6 \wedge \mathtt{cA}_4 \leq 10$ and $\Phi_4 : \mathtt{cA}_4 < \hat{\mathtt{c}} \wedge \mathtt{cC}_4 = \hat{\mathtt{c}} \wedge \mathtt{cA}_4 \wedge \mathtt{cA}_4 > 10$.

We graphically represent a TGA as a (multi)graph where the set of nodes coincides with that of locations whereas the set of edges models controllable transitions (solid edges) and uncontrollable ones (dashed edges). Figure 2 depicts a TGA encoding the CSTNU in Figure 1. In what follows we sum up how this encoding is achieved and dynamic controllability checked.

2.3 Dynamic Controllability

The DC-checking problem is the problem of deciding if a CSTNU is DC. We can answer the DC-checking problem by using *sound* and *complete* TGA reachability algorithms [4, 5]. We model the DC-checking as a two-player game between a controller (`ctrl`) and the environment (`env`). The aim of `ctrl` is to reach a specific location as soon as all relevant time points have been executed and all constraints are satisfied, whereas `env`'s goal is to prevent `ctrl` from doing that. If `ctrl` wins, the network is DC, otherwise it is not. An important aspect of this encoding is that `ctrl` is assigned *uncontrollable* transitions, whereas `env` is assigned *controllable* ones. This is necessary to allow `env`'s instantaneous reactions as in the TGA semantics, uncontrollable transitions go first [4, 5, 6]. The encoding is as follows.

Clocks. $\mathcal{X}$ contains a clock $\mathtt{cX}$ for each time point $X \in \mathcal{T}$ and a clock $\mathtt{bP}$ for each proposition $p \in \mathcal{P}$. $\mathcal{X}$ also contains two special clocks $\hat{\mathtt{c}}$ (modeling the global time) and $\mathtt{c}_\delta$ (regulating the interplay of the game). $\mathtt{cX} = \hat{\mathtt{c}}$, means that X has not been executed, whereas $\mathtt{cX} < \hat{\mathtt{c}}$ means that X was executed at time $\hat{\mathtt{c}} - \mathtt{cX}$ (when this difference is > 0). Likewise, $\mathtt{bP} = \hat{\mathtt{c}}$ means that $p = \top$, whereas $\mathtt{bP} < \hat{\mathtt{c}}$ means that $p = \bot$ (both when $\mathtt{cP} < \hat{\mathtt{c}}$). Each $\mathtt{cX}$ and $\mathtt{bP}$ may be reset at most once. For our example we have $\mathcal{X} = \{\hat{\mathtt{c}}, \mathtt{c}_\delta, \mathtt{cP}, \mathtt{cA}_1, \mathtt{cC}_1, \mathtt{cA}_2, \mathtt{cC}_2, \mathtt{cD}, \mathtt{cA}_3, \mathtt{cC}_3, \mathtt{cA}_4, \mathtt{cC}_4, \mathtt{cE}, \mathtt{bP}, \mathtt{bD}\}$.

Locations. *Loc* contains three core locations L_0 (initial), L_1 (urgent) and `goal` (urgent), and $n-1$ urgent locations $L_{\ell_1}, \ldots, L_{\ell_{n-1}}$ where n is the number of distinct labels in the CSTNU. That is, $n = |\{L(X) \mid X \in \mathcal{T}\} \cup \{\ell \mid (Y - X \leq k, \ell) \in \mathcal{C}\}|$. For our example, $Loc = \{L_0, L_1, L_{\boxdot}, L_p, L_{\neg p}, L_d, \mathtt{goal}(= L_{\neg d}\}$ as the distinct labels are $\{\boxdot, p, \neg p, d, \neg d\}$.

Transitions. $\rightarrow$ contains controllable and uncontrollable transitions to model the following:

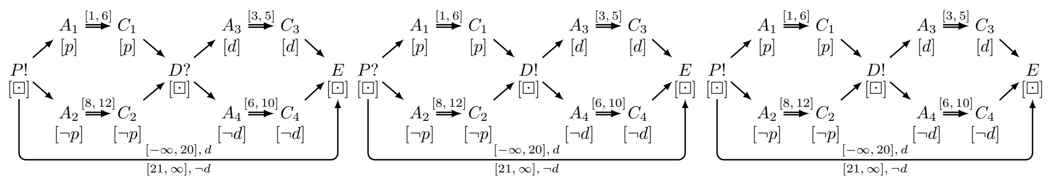
- *Game interplay.* `pass` and `gain` are uncontrollable transitions regulating the game interplay. In particular `gain` can be taken only when $\mathtt{c}_\delta > 0$ modeling the *reaction time* needed to observe how the uncontrollable part behaves.
- *Non-contingent time point executions.* For each non-contingent time point X there is an uncontrollable self-loop transition $\langle L_1, \mathtt{cX} = \hat{\mathtt{c}}, \mathtt{exX}, \{\mathtt{cX}\}, L_1\rangle$ modeling the execution of X. The guard says that X has not been executed yet, while the reset fixes the execution time of X to $\hat{\mathtt{c}} - \mathtt{cX}$ by resetting $\mathtt{cX}$.
- *Contingent time point executions.* For each contingent link $(A, x, y, C) \in \mathcal{L}$ there is a controllable self-loop transition $\langle L_0, \mathtt{cA} < \hat{\mathtt{c}} \wedge \mathtt{cC} = \hat{\mathtt{c}} \wedge \mathtt{cA} \geq x \wedge \mathtt{cA} \leq y, \mathtt{exC}, \{\mathtt{cC}, \mathtt{c}_\delta\}, L_0\rangle$ to allow `env` to execute the contingent time point C such that $C - A \in [x, y]$, and a fail transition $\langle L_0, \mathtt{cA} < \hat{\mathtt{c}} \wedge \mathtt{cC} = \hat{\mathtt{c}} \wedge \mathtt{cA} > y, \mathtt{failC}, \{\mathtt{cC}, \mathtt{c}_\delta\}, \mathtt{goal}\rangle$ to allow `ctrl` to move to `goal` if `env` fails or refuses to take the transition.
- *Truth value assignments.* For each proposition $p \in \mathcal{P}$ there is a controllable self-loop transition $\langle L_0, \mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{cP} = 0 \wedge \mathtt{bP} = \hat{\mathtt{c}}, \mathtt{pFalse}, \{\mathtt{bP}, \mathtt{c}_\delta\}, L_0\rangle$ to allow `env` to assign $\bot$ to p, if it decides so. If it does not, the truth value of p will remain forever $\top$.
- *Winning conditions.* To check that all relevant time points have been executed and all constraints are satisfied we connect each pair of locations $(L_{\ell_{i-1}}, L_{\ell_i})$ in the winning path $L_0 \rightarrow L_\boxdot \rightarrow \cdots \rightarrow L_{\ell_{n-1}} \rightarrow \mathtt{goal}$ by means of a set of uncontrollable transitions. Each set of transitions going from $L_{\ell_{i-1}}$ to L_{ℓ_i} verifies that if ℓ_{cps} does not falsify ℓ_i, then all time points labeled by ℓ_i must have been executed and all constraints labeled by ℓ_i are satisfied. If ℓ_{cps} falsifies ℓ_i, a `skip` transition allows us to ignore this check. In this way, the problem is decomposed with respect to the specific labels avoiding the combinatorial explosion of all arising cases. For example, the set of transitions going from $L_\boxdot$ to L_p is generated as follows. In the scenario where $P?$ has been executed and p assigned $\top$ (i.e., $\ell_{cps} = p$), then A_1 and C_1 must have been executed, and $A_1 - P? \leq 1$, $P? - A_1 \leq -1$, $D? - C_1 \leq 1$, $C_1 - D? \leq -1$ are satisfied. In other words, the meta conditional constraint $(\mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{bP} = \hat{\mathtt{c}}) \implies (\mathtt{cA}_1 < \hat{\mathtt{c}} \wedge \mathtt{cC}_1 < \hat{\mathtt{c}} \wedge \mathtt{cP} - \mathtt{cA}_1 = 1 \wedge \mathtt{cC}_1 - \mathtt{cD} = 1)$ refines to $\neg(\mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{bP} = \hat{\mathtt{c}}) \vee (\mathtt{cA}_1 < \hat{\mathtt{c}} \wedge \mathtt{cC}_1 < \hat{\mathtt{c}} \wedge \mathtt{cP} - \mathtt{cA}_1 = 1 \wedge \mathtt{cC}_1 - \mathtt{cD} \geq 1 = 1)$ simplifying to $(\mathtt{cP} = \hat{\mathtt{c}}) \vee (\mathtt{bP} < \hat{\mathtt{c}}) \vee (\mathtt{cA}_1 < \hat{\mathtt{c}} \wedge \mathtt{cC}_1 < \hat{\mathtt{c}} \wedge \mathtt{cP} - \mathtt{cA}_1 = 1 \wedge \mathtt{cC}_1 - \mathtt{cD} = 1)$ since TGAs do not allow negations or disjunctions of clock constraints in the guards. Finally, we generate a transition[1] for each disjunct ($\mathtt{sat}_\mathtt{p}$,$\mathtt{skip}^1_\mathtt{p}$,$\mathtt{skip}^2_\mathtt{p}$).

DC-checking is done by looking for a control strategy for `env` to always prevent `ctrl` from getting to `goal`. If such a strategy exists, the initial CSTNU is not DC, otherwise it is (as `ctrl` has a counter-strategy to react to any combination of `env`'s moves).

3 CSTNUs with Decisions

In this section we extend CSTNUs by injecting a new kind of time point: the *decision time point.* A decision time point $D!$ dualizes an observation one $P?$ as the truth value assignment to the associated proposition is *under control.* As a result, the controllable and uncontrollable part may now mutually influence one another. That is, deciding some truth value may restrict (or even exclude) some uncontrollable part and vice versa. Several interesting cases may arise depending on if a few truth values are decided before or after having full information on how

[1] We model $Y - X \leq k$ as $(\hat{\mathtt{c}} - \mathtt{cY}) - (\hat{\mathtt{c}} - \mathtt{cX}) \leq k$ simplifying to $\mathtt{cX} - \mathtt{cY} \leq k$. We might write $\mathtt{cX} - \mathtt{cY} \geq k$ as a short for $X - Y \leq -k$ and $\mathtt{cX} - \mathtt{cY} = k$ as a short for the pair $Y - X \leq k$ and $X - Y \leq -k$.



A_1 [1,6] C_1 [p] [p]
A_3 [3,5] C_3 [d] [d]
P! [⊡]
D? [⊡]
E [⊡]
A_2 [8,12] C_2 [¬p] [¬p]
A_4 [6,10] C_4 [¬d] [¬d]
[−∞, 20], d
[21, ∞], ¬d
P? [⊡]
D! [⊡]
P! [⊡]
D! [⊡]

(a) *A decision before any uncontrollable part.* **(b)** *A decision after all observation and some contingent.* **(c)** *A decision after another decision and a contingent.*

Figure 3 Possible cases of the CSTNU in Figure 1 when substituting decision time points for observation ones. Missing labels on requirement links $X \to Y$ are all $[1,1], L(X) \wedge L(Y)$ (Figure 1).

the uncontrollable part will or have behaved. We go ahead with this discussion by taking Figure 3 as an example. There, we took the initial CSTNU in Figure 1 and substituted decision time points for observation ones in all possible combinations. We discuss these examples focusing on the combinations of minimal and maximal durations of contingent links only. If it works for them, then it must work for any other combination of durations.

- In Figure 3a $P!$ is a decision time point. The resulting CSTNUD is uncontrollable. If we *decide* p (i.e., assign $\top$ to p), then *observe* $\neg d$ (i.e., $D?$ assigns $\bot$ to d) and C_1, C_4 take their maximal durations, then we will have to execute E at 20 violating $(P? - E \leq -21, \neg d)$ as $P?$ is executed at 0. Conversely, if we decide $\neg p$, then observe d and C_2 and C_3 take their maximal durations, then we will have to execute E at 21 (violating $E - P? \leq 20, d$).
- In Figure 3b $D!$ is a decision time point. The resulting CSTNUD is DC. Assume that we observe p. Regardless on what duration C_1 takes, we can only decide d. Indeed, if we decided $\neg d$, regardless of the duration of C_4 we would have to execute E before time 21 violating $(P? - E \leq -21, \neg d)$. Assume now that we observe $\neg p$. If C_2 takes its minimal duration, d is the only good decision. If we decided $\neg d$ and then C_4 took its minimal duration, we would execute E at 18 violating $(P? - E \leq -21, \neg d)$. On the contrary, if C_2 takes its maximal duration then we can only decide $\neg d$. If we decided d and C_3 took its maximal duration, we would have to execute E at 21 violating $(E - P? \leq 20, d)$.
- In Figure 3c $P!$ and $D!$ are both decision time points. The resulting CSTNUD is of course[2] dynamically controllable. If we decide p, then deciding d is always going to be fine. If we decide $\neg p$, then we will decide either d or $\neg d$ depending on how long C_2 lasts. If C_2 takes its minimal duration, then we will decide d (but not $\neg d$ since C_4 could then take its minimal duration). If C_2 takes its maximal duration, then we will decide $\neg d$ (but not d since if C_3 could then take its maximal duration).

  Hence, *decisions are dynamic.*

▶ **Definition 3** (CSTNUD). A *Conditional Simple Temporal Network with Uncertainty and Decisions* (*CSTNUD*) is a tuple $\langle \mathcal{T}, \mathcal{OT}, \mathcal{DT}, \mathcal{P}, O, L, \mathcal{L}, \mathcal{C} \rangle$, where:

- $\mathcal{T}, \mathcal{OT}, \mathcal{P}, L, \mathcal{L}, \mathcal{C}$ are exactly the same of those given for CSTNUs. Furthermore, we denote the set of contingent time points as *Contingent* $= \{C \mid (A, x, y, C) \in \mathcal{L}\}$.
- $\mathcal{DT} \subseteq \mathcal{T} = \{D!, E!, \dots\}$ is a set of *decision time points* such that $\mathcal{OT} \cap \mathcal{DT} = \emptyset$.
- $O \colon \mathcal{P} \to \mathcal{DT} \cup \mathcal{OT}$ is a bijection associating a unique observation or decision time point to each proposition. If $O(p) \in \mathcal{OT}$, then p is called *observable*, whereas if $O(d) \in \mathcal{DT}$,

[2] If a network is DC (e.g., Figure 3b), then turning controllable some uncontrollable part (e.g., Figure 3c) *cannot worsen* the situation turning the network uncontrollable. The vice versa does not hold.

then d is called *decidable.* $\mathcal{OP} \subseteq \mathcal{P} = \{p \mid O(p) \in \mathcal{OT}\}$ and $\mathcal{DP} \subseteq \mathcal{P} = \{d \mid O(d) \in \mathcal{DT}\}$ shorten the sets of all observable and decidable propositions, where $\mathcal{OP} \cap \mathcal{DP} = \emptyset$.

A CSTNUD is *well-defined* if and only if the underlying CSTNU is well-defined and time point label honesty extends to decidable propositions as follows: For each $X \in \mathcal{T}$, if $\lambda \in L(X)$, where $\lambda = \{d, \neg d\}$ and $d \in \mathcal{DP}$, then $L(X) \Rightarrow L(O(d))$ and $(O(d) - X \leq 0, L(X)) \in \mathcal{C}$). That is, X can be executed at the same time of $D!$ (but instantaneously after $D!$ since time points executed at the same instant must in general follow an *order of execution*).

We model the execution semantics of a CSTNUD as a two-player game in which `Player1` models the controller and `Player2` models the environment. We employ *execution sequences* [16] to model the state of the game and define players' strategies as mappings from execution sequences considered at specific time instants to *moves*.

A *sequence* $\{x_1, x_2, \dots, x_n\}$ is a totally ordered collection of elements such that for any pair of elements x_i, x_j, if $i < j$ (resp., $i > j$), then it means that x_i is before (resp., after) x_j. We abuse notation and write $\{x_1, x_2, \dots, x_n\} \cup \{x_p\}$ to mean the appending operation resulting in $\{x_1, x_2, \dots, x_n, x_p\}$ where $n < p$. We write $x_i \in \{x_1, x_2, \dots, x_n\}$ iff there exists $j \in \mathbb{N}$, $1 \leq j \leq n$ such that $x_i = x_j$ (membership), and $|\{x_1, x_2, \dots, x_n\}| = n$ (cardinality). A *partial schedule* for a subset of time points $\mathcal{T}' \subseteq \mathcal{T}$ is a mapping $S_{\mathcal{T}'} \colon \mathcal{T}' \to \mathbb{R}$ assigning a real value to each $X \in \mathcal{T}'$. A *partial schedule* for a subset of Boolean propositions $\mathcal{P}' \subseteq \mathcal{P}$ is a mapping $S_{\mathcal{P}'} \colon \mathcal{P}' \to \{\top, \bot\}$ assigning either $\top$ or $\bot$ to each $p \in \mathcal{P}'$. We write $\mathtt{b}$ for a generic Boolean value (i.e., $\mathtt{b} \in \{\top, \bot\}$). We write $S_{\mathcal{T}'} \cup \{S_{\mathcal{T}'}(Y) = k\}$ to shorten that the domain of $S_{\mathcal{T}'}$ extends by adding time point Y such that $S_{\mathcal{T}'}(Y) = k$. Similarly, we write $S_{\mathcal{P}'} \cup \{S_{\mathcal{P}'}(p) = \mathtt{b}\}$ for Boolean propositions.

▶ **Definition 4** (Instantiation sequence)**.** An *instantiation sequence* is a quadruple $\langle E, K, S_E, S_K \rangle$, where E is a finite sequence of distinct time points in $\mathcal{T}$, K is a finite sequence of distinct propositions in $\mathcal{P}$, and S_E, S_K are partial schedules whose domains are E and K, respectively.

▶ **Definition 5** (Execution sequence)**.** An *execution sequence* $Z = \langle E, K, S_E, S_K \rangle$ is an instantiation sequence satisfying the following properties:

S_E Monotonicity For any pair $X_i, X_j \in E$ if $i < j$, then $S_E(X_i) \leq S_E(X_j)$.

(Time Point Label) Honesty For each $X \in E$ and each literal $\lambda \in L(X)$ where $\lambda \in \{p, \neg p\}$, then $O(p) \in E$ and $O(p)$ is before X, $p \in K$, $S_K(p) = \top$ (if $\lambda = p$) and $S_K(p) = \bot$ (if $\lambda = \neg p$). Also, $S_E(O(p)) < S_E(X)$ (if $p \in \mathcal{OP}$) and $S_E(O(p)) \leq S_E(X)$ (if $p \in \mathcal{DP}$).

Z^* represents the set of all execution sequences. $t_{last}(Z) = \max\{S_E(X) \mid X \in E\}$ represents the last time instant in which a time point was executed in Z. $last(Z) = \{X \mid X \in E \wedge S_E(X) = t_{last}\}$ represents the set of the last executed time points.

Therefore, an execution sequence models the ordered sequence of executed time points and assigned propositions according to the well-definedness of a CSTNUD. As an example, consider again Figure 3b. Assume that we execute $P?$ at 0 and observe $\neg p$. Assume then that we execute A_2 at 1 and observe C_2 to occur at 13 (i.e., at its maximal duration). The execution sequence is $Z = \langle \{P?, A_2, C_2\}, \{p\}, \{S_E(P?) = 0, S_E(A_2) = 1, S_E(C_2) = 13\}, \{S_K(p) = \bot\} \rangle$. We can now compute the current partial scenario as the conjunction of all positive and negative literals arising from all propositions in K according to S_K and define local consistency.

▶ **Definition 6** (Current partial scenario)**.** Given any $Z = \langle E, K, S_E, S_K \rangle$, the *current partial scenario* is given by $\ell_{cps} = \lambda_1 \wedge \dots \wedge \lambda_k$, where for each $p_i \in K$, $\lambda_i = p_i$ (if $S_K(p_i) = \top$) and $\lambda_i = \neg p_i$ (if $S_K(p_i) = \bot$).

For Z we have that $\ell_{cps} = \neg p$ since $p \in K$ and $S_K(p) = \bot$.

▶ **Definition 7** (Local consistency)**.** An execution sequence $E = \langle E, K, S_E, S_K\rangle$, is *locally consistent* if and only if for each $(Y - X \le k, \ell) \in \mathcal{C}$ where $X, Y \in E$ and $\ell_{cps} \Rightarrow \ell$, $S_E(Y) - S_E(X) \le k$ holds.

Z is locally consistent since the schedule S_E satisfies $(A_2 - P? \le 1, \neg p)$ and $(P? - A_2 \le -1, \neg p)$. An execution sequence evolves over time according to the evolution of the game that `Player1` (the controller) plays against `Player2` (the environment). Each player follows a strategy saying what moves to make and when. Moreover, many moves can be made at the same time instant (provided that they respect an order) and sometimes moves are mandatory.

▶ **Definition 8** (Move)**.** A *move m is either X meaning "execute time point X" or* $(p, \mathtt{b})$ *meaning "assign $\mathtt{b} \in \{\top, \bot\}$ to proposition p".* A move for `Player1` requires that X is a *non-contingent* time point and p is a *decidable* proposition. A move for `Player2` requires that X is a *contingent* time point and p is an *observable* proposition. M_1^* and M_2^* represent the sets of all moves for `Player1` and `Player2`, respectively.

A *move-based strategy* is a mapping from execution sequences considered at particular time instants to moves augmented with a `wait` condition modeling the absence of move. A strategy tells a player to make a move at a particular time instant only if the move is applicable at that particular time. Therefore, a strategy specifies *applicability conditions* saying when a move *can* be made, *obligations* saying when a move *has to* be made and *postconditions* saying how the execution sequence evolves accordingly.

▶ **Definition 9** (Move-based strategy)**.** A *move-based strategy* for `Player1` is a mapping $\sigma_1 \colon Z^* \times \mathbb{R} \to M_1^* \cup \{\mathtt{wait}\}$ such that its *applicability conditions are*:

1. For any execution sequence Z and any time instant t, $\sigma_1(Z, t)$ is *applicable* iff $t \sim t_{last}(Z)$, where $\sim$ is $>$ if $last(Z)$ contains a contingent time point C or an observation time point $P?$ such that K contains its related proposition p (reaction time enforcement), $\ge$ otherwise.
2. For any execution sequence Z and any time instant t, $\sigma_1(Z, t) = X$ is applicable if (1) holds and X is an unexecuted non-contingent time point such that the current partial scenario entails $L(X)$ (i.e., $X \notin E \wedge X \notin \mathit{Contingent} \wedge \ell_{cps} \Rightarrow L(X)$).
3. For any execution sequence Z and any time instant t, $\sigma_1(Z, t) = \mathtt{wait}$ is applicable if (1) holds and there is no obligation at time t.

The unique *obligation* involves decidable propositions requiring that whenever a decision time point $D!$ has been executed and its related proposition d has not been assigned yet, then the strategy must issue a move to assign d a truth value instantaneously. In symbols: $D! \in E \wedge d \notin K \implies \sigma_1(Z, S_E(D!)) = (d, \mathtt{b})$.

A *move-based strategy* for `Player2` is a mapping $\sigma_2 \colon Z^* \times \mathbb{R} \to M_2^* \cup \{\mathtt{wait}\}$ such that its *applicability conditions* are:

1. For any execution sequence Z and any time instant t, $\sigma_2(Z, t)$ is *applicable* iff $t \ge t_{last}(Z)$.
2. For any execution sequence Z, any time instant t and any contingent link $(A, x, y, C) \in \mathcal{L}$, $\sigma_2(Z, t) = C$ is applicable if (1) holds, A has already been executed, C has not, and executing C at this time satisfies $C - A \in [x, y]$ (i.e., $A \in E \wedge C \in \mathit{Contingent} \wedge C \notin E \wedge t - S_E(A) \in [x, y]$).
3. For any execution sequence Z and any time instant t, $\sigma_2(Z, t) = \mathtt{wait}$ is applicable if (1) holds and there is no obligation at time t.

Obligations are of two kinds. The first obligation involves observable propositions requiring that whenever an observation time point $P?$ has been executed and its related proposition p has not been assigned yet, then the strategy must issue a move to assign p a truth value instantaneously. In symbols: $(P? \in E \wedge p \notin K) \implies \sigma_2(Z, S_E(P?)) = (p, \mathtt{b})$. The second obligation involves contingent links (A, x, y, C) requiring that if A has already been executed, C has not and the current time t is the last instant in which C can be executed, then the strategy must issue a move to execute C at t. In symbols: $\forall (A, x, y, C) \in \mathcal{L}, \forall t \in \mathbb{R}, (A \in E \wedge C \notin E \wedge t - S_E(A) = y) \implies \sigma_2(Z, t) = C$.

Postconditions for both σ_1 and σ_2 are the same. If the strategy tells the player to execute a time point X at time t then Z updates by appending X to E and extending S_E such that $S_E(X) = t$. If the strategy tells the player to assign the truth value $\mathtt{b}$ to the proposition p, then Z updates by appending p to K and extending S_K such that $S_K(p) = \mathtt{b}$. In symbols:

- If $\sigma_i(Z, t) = X$, then $\mathit{Post}(Z, \sigma_i, t) = \langle E \cup \{X\}, K, S_E \cup \{S_E(X) = t\}, S_K \rangle$.
- If $\sigma_i(Z, t) = (p, \mathtt{b})$, then $\mathit{Post}(Z, \sigma_i, t) = \langle E, K \cup \{p\}, S_E, S_K \cup \{S_K(p) = \mathtt{b}\} \rangle$.

Getting back to our example we have that $t_{last}(Z) = 13$ and $\mathit{last}(Z) = \{C_2\}$. Suppose that current time is $t = 14$. $\sigma_1(Z, 14) = D!$ is applicable since $t > t_{last}$ and $D!$ has not been executed yet, whereas $\sigma_1(Z, 14) = (d, \top)$ is not since $D! \notin E$. If $\sigma_1(Z, 14) = D!$ is taken into consideration (i.e., $Z' = \mathit{Post}(Z, \sigma_1, t)$), then $\sigma_1(Z', 14) = (d, \top)$ instantaneously after.

We now model `Player2` as the *most powerful player possible*. If `Player1` can beat this (worst-case of) environment, then `Player1` must be able to beat any other less powerful environment playing the same game. To achieve this purpose we model the game in *turns*. That is, at any time instant t, there exist two turns: $T_1(t)$ (occurring first) and $T_2(t)$ (occurring last). `Player1` makes his moves in $T_1(t)$, whereas `Player2` makes his in $T_2(t)$. If player i does not make any move in $T_i(t)$, then he loses forever the possibility to play at time t. As a result, `Player2`, making his moves in $T_2(t)$, is guaranteed to always have full information on what `Player1` has done in $T_1(t)$ (worst-case scenario). In what remains of this section we define the concept of *snapshot* modeling an execution sequence a particular time instant t (after the players are done in $T_1(t)$ and $T_2(t)$), *continuous game evolution* modeling how the execution sequence evolves and *winning conditions* for each player.

▶ **Definition 10** (Snapshot). Let $Z = \langle E, K, S_E, S_K \rangle$ be any execution sequence. $Z(t) = \langle E', K', S'_E, S'_K \rangle$ models the *snapshot* of Z at time t, where $E' = \{X \mid X \in E \wedge S_E(X) \leq t\}$, $K' = \{p \mid p \in K \wedge O(p) \in E'\}$, $\forall X \in E', S'_E(X) = S_E(X)$, and $\forall p \in K', S'_K(p) = S_K(p)$.

To give an example, let us get back to the execution sequence we have discussed before. At $t = 11$, we have $Z(11) = \langle \{P?, A_2\}, \{p\}, \{S_E(P?) = 0, S_E(A_2) = 1\}, \{S_K(p) = \bot\} \rangle$.

▶ **Definition 11** (Continuous game evolution). Let $t \in \mathbb{R}^{\geq 0}$ be the global time. The continuous game evolution is modeled by an infinite sequence of snapshots $Z(t)$ defined as:

$$Z(t) = \begin{cases} T_2(T_1(\langle \emptyset, \emptyset, \emptyset, \emptyset \rangle, t), t) & \text{if } t = 0 \\ T_2(T_1(Z(t - \epsilon), t), t) & \text{if } t > 0 \end{cases} \quad T_i(Z, t) = \begin{cases} Z & \text{if } \sigma_i(Z, t) = \mathtt{wait} \\ T_i(\mathit{Post}(Z, \sigma_i, t), t) & \text{otherwise} \end{cases}$$

where $T_i(t)$ models the evolution of Z during turn i at time t according to σ_i, whereas $\epsilon > 0$.

▶ **Definition 12** (Winning conditions). `Player1` wins the game if and only if the game evolution leads to a snapshot $Z(t)$ such that for each unexecuted time point X, ℓ_{cps} falsifies $L(X)$ and for each constraints $(Y - X \leq k, \ell)$ where $X, Y \in E$ and $\ell_{cps} \Rightarrow \ell$, $S_E(Y) - S_E(X) \leq k$ holds. `Player2` wins otherwise. σ_i is a *winning strategy* if player i wins the game by following σ_i.

▶ **Definition 13** (Dynamic controllability). A CSTNUD is dynamically controllable if `Player1` has a winning strategy such that for any $t > 0$ and any pair of execution sequences Z_1, Z_2, if $\sigma_2(Z_1, t') = \sigma_2(Z_2, t')$ for $0 \leq t' < t$, then $\sigma_1(Z_1, t) = \sigma_1(Z_2, t)$.

In other words, whenever `Player2` has made the same (infinite) sequence of moves up to time $t - \epsilon$, then `Player1` will make the same move(s) at time t.

4 Dynamic Controllability of CSTNUDs via TGAs

In this section we extend the encoding given for CSTNUs in Section 2. As an example, we consider Figure 4 depicting the encoding of the CSTNUD in Figure 3b.

Once again, we have three core locations but this time we borrow a few names from Section 3 and rename them to $\mathtt{T}_1$ (ex L_1), $\mathtt{T}_2$ (ex L_0) and $\mathtt{win}$ (ex $\mathtt{goal}$). $\mathtt{T}_1$ and $\mathtt{T}_2$ model the two turns $T_1(t)$ and $T_2(t)$ when global time is > 0. $\mathtt{T}_2$ is the initial location. The winning path is computed the same way only renaming each L_{ℓ_i} to $\mathtt{w}_{\ell_i}$. $\mathtt{gain}$ and $\mathtt{pass}$ regulate the turns at any time instant t. We still have a clock $\mathtt{cX}$ for each $X \in \mathcal{T}$ (considering decision time points too) and a clock $\mathtt{bP}$ for each $p \in \mathcal{P}$ (considering decidable propositions too).

We optimize the guard of each uncontrollable self-loop at $\mathtt{T}_1$ by exploiting what we know of the CSTNUD. That is, we extend the guards so that they enforce time point label honesty as well as the partial order among the time points when not ambiguous. This optimization was first discussed in [8] but there it dealt with disjunctive constraints and exploited internal data structures provided by the UPPAAL-TIGA software. Here, we propose a more formal definition avoiding such data structures. Moreover, [8] does not address decisions.

To give an example of this optimization, consider time points A_1 and A_4 of the CSTNUD in Figure 3b. $L(A_1) = p$ and $L(A_4) = \neg d$. Recall that the encoding models p and d as two dedicated clocks $\mathtt{bP}$ and $\mathtt{bD}$ such that if each of these clocks is equal to (resp., less than) $\hat{\mathtt{c}}$, once its related observation or decision time point has been executed, then the related proposition is $\top$ (resp., $\bot$). Moreover, time point label honesty also requires that $P? - A_1 \le -\epsilon$ (observation) and $D! - A_4 \le 0$ (decision).

Therefore, considering the time point label honesty property for CSTNUDs, it is possible to extend the guards of $\mathtt{exA}_1$ and $\mathtt{exA}_4$ by appending $\mathtt{bP} = \hat{\mathtt{c}} \wedge \mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{cP} > 0$ and $\mathtt{bD} < \hat{\mathtt{c}} \wedge \mathtt{cD} < \hat{\mathtt{c}} \wedge \mathtt{cD} \ge 0$, respectively. The former models the fact that A_1 must be executed if only if $p = \top$ (i.e., $\mathtt{bP} = \hat{\mathtt{c}}$), which also implies that A_1 must be executed after $P?$ (i.e., $P?$ have been executed ($\mathtt{cP} < \hat{\mathtt{c}}$)) and a positive amount of time ϵ has elapsed ($\mathtt{cP} > 0$). The latter models the fact that A_4 must be executed if only if $d = \bot$ (i.e., $\mathtt{bD} < \hat{\mathtt{c}}$), which also implies that A_4 must be executed after $D!$ (i.e., $D!$ have been executed ($\mathtt{cD} < \hat{\mathtt{c}}$)) and possibly immediately or after a positive amount of time has elapsed ($\mathtt{cD} \ge 0$).

▶ **Definition 14** (Encoding time point label honesty). A *label encoder* is a mapping $L_{enc} \colon \mathcal{T} \to \mathcal{H}(\mathcal{X})$ translating the label of a time point into the equivalent clock constraint $L_{enc}(X) = L^{\mathcal{OP}}_{enc}(X) \wedge L^{\mathcal{DP}}_{enc}(X)$, where $L^{\mathcal{OP}}_{enc}(X)$ and $L^{\mathcal{DP}}_{enc}(X)$ encode all literals containing observable and decidable propositions, respectively.

- $L^{\mathcal{OP}}_{enc}(X)$: $\bigwedge_{p \in L(X)} (\mathtt{bP} = \hat{\mathtt{c}} \wedge \mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{cP} > 0) \bigwedge_{\neg q \in L(X)} (\mathtt{bQ} < \hat{\mathtt{c}} \wedge \mathtt{cQ} < \hat{\mathtt{c}} \wedge \mathtt{cQ} > 0)$
- $L^{\mathcal{DP}}_{enc}(X)$: $\bigwedge_{d \in L(X)} (\mathtt{bD} = \hat{\mathtt{c}} \wedge \mathtt{cD} < \hat{\mathtt{c}} \wedge \mathtt{cD} \ge 0) \bigwedge_{\neg f \in L(X)} (\mathtt{bF} < \hat{\mathtt{c}} \wedge \mathtt{cF} < \hat{\mathtt{c}} \wedge \mathtt{cF} \ge 0)$

We now focus on constraints. Consider the requirement link $P? \to A_1$ labeled by $[1, 1], p$ in the CSTNUD that we are discussing. Such a constraint says that A_1 must be executed after 1 and within 1 since $P?$ (thus, exactly after 1 since $P?$). This requirement link has also an important characteristic: $L(A_1)$ coincides with the label of the link. Therefore, whenever A_1 is executed, the constraint must hold. Thus, we extend the original guard of $\mathtt{exA}_1$ (formerly $\mathtt{cA}_1 = \hat{\mathtt{c}}$) to $\mathtt{cA}_1 = \hat{\mathtt{c}} \wedge \mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{cP} = 1$, where the new conjuncts say that $P?$ has already been executed ($\mathtt{cP} < \hat{\mathtt{c}}$) and $A_1 - P? \in [1, 1]$ ($\mathtt{cP} = 1$). More formally:

▶ **Definition 15** (Encoding predecessors). Given a CSTNUD, a *predecessor* of a time point $Y \in \mathcal{T}$ is a time point $X \in \mathcal{T}$ such that there exists a constraint $(X - Y \le -x, L(Y)) \in \mathcal{C}$ where $x > 0$. $\Pi : \mathcal{T} \to 2^{\mathcal{T}}$ returns the predecessors of a given time point and it is formalized as $\Pi(Y) = \{X \mid (X - Y \le -x, \ell) \in \mathcal{C} \wedge x > 0 \wedge \ell = L(Y)\}$. A *predecessor encoder* is a

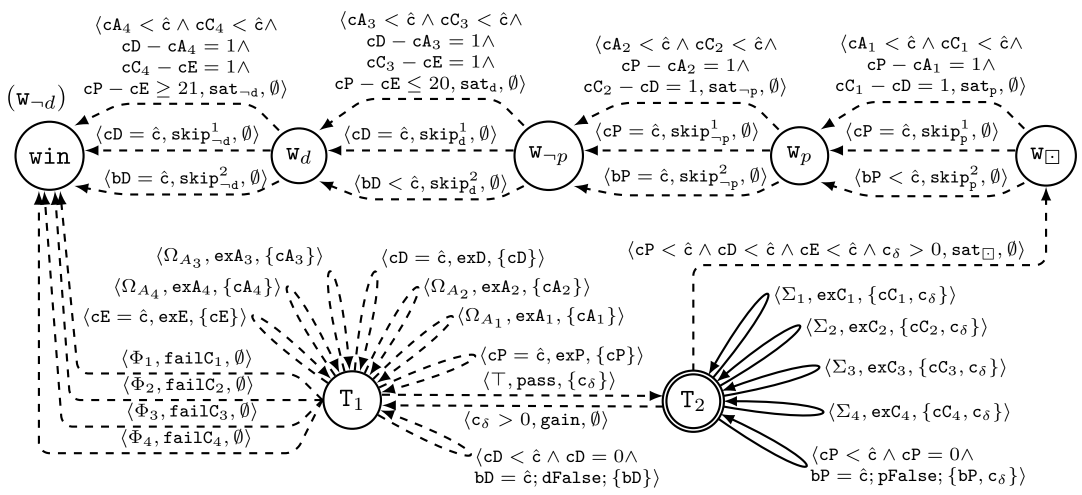


Figure 4 TGA encoding the CSTNUD in Figure 3b. $\mathtt{T}_2$ (ex L_0) is the initial location (modeling $T_2(t)$ for $t > 0$). $\mathtt{T}_1$ (ex L_1) models $T_1(t)$ for $t > 0$). $\mathtt{w}_\boxdot$, $\mathtt{w}_p$, $\mathtt{w}_{\neg p}$, $\mathtt{w}_d$, $\mathtt{win}$ model the winning path. $\Omega_{A_1}\colon \mathtt{cA}_1 = \hat{\mathtt{c}} \wedge \mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{bP} = \hat{\mathtt{c}} \wedge \mathtt{cP} > 0 \wedge \mathtt{cP} = 1$. $\Omega_{A_2}\colon \mathtt{cA}_2 = \hat{\mathtt{c}} \wedge \mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{bP} < \hat{\mathtt{c}} \wedge \mathtt{cP} > 0 \wedge \mathtt{cP} = 1$. $\Omega_{A_3}\colon \mathtt{cA}_3 = \hat{\mathtt{c}} \wedge \mathtt{cD} < \hat{\mathtt{c}} \wedge \mathtt{bD} = \hat{\mathtt{c}} \wedge \mathtt{cD} \geq 0 \wedge \mathtt{cD} = 1$. $\Omega_{A_4}\colon \mathtt{cA}_4 = \hat{\mathtt{c}} \wedge \mathtt{cD} < \hat{\mathtt{c}} \wedge \mathtt{bD} < \hat{\mathtt{c}} \wedge \mathtt{cD} \geq 0 \wedge \mathtt{cD} = 1$.

mapping $\Pi_{enc}\colon \mathcal{T} \to \mathcal{H}(\mathcal{X})$ translating each $X \in \Pi(Y)$ (along with its temporal bounds) into an equivalent clock constraint as follows. $\Pi_{enc}(Y) = \bigwedge_{X\in\Pi(Y)} \mathtt{cX} < \hat{\mathtt{c}} \wedge \mathtt{cX} \geq x \wedge \mathtt{cX} \leq y$, where $\mathtt{cX} \geq x$ models $(X - Y \leq -x, L(Y))$ and $\mathtt{cX} \leq y$ models $(Y - X \leq y, L(Y))$ (if any).

Therefore, for each non-contingent time point X, the guard of $\mathtt{exX}$ becomes $\Omega_X\colon \mathtt{cX} = \hat{\mathtt{c}} \wedge L_{enc}(X) \wedge \Pi_{enc}(X)$. In Figure 4 we shortened the guards of $\mathtt{exA}_1$, $\mathtt{exA}_2$, $\mathtt{exA}_3$ and $\mathtt{exA}_4$ as Ω_{A_1}, Ω_{A_2}, Ω_{A_3} and Ω_{A_4} and detailed them in the caption.

After optimizing the guard of each $\mathtt{exX}$ transition we now discuss how to model the truth value assignment to the decidable propositions. Dually to observable propositions, for each decidable proposition $d \in \mathcal{DP}$ we generate an uncontrollable self-loop transition $\langle \mathtt{T}_1, \mathtt{cD} < \hat{\mathtt{c}} \wedge \mathtt{cD} = 0 \wedge \mathtt{bD} = \hat{\mathtt{c}}, \mathtt{dFalse}, \{\mathtt{bD}\}, \mathtt{T}_1\rangle$ at $\mathtt{T}_1$. If we take this transition, it means that we decide $\neg d$. If we do not, it means that we decide (actually confirm) d. In the former case, such a transition has to be taken at the same instant in which $D!$ was executed but after $\mathtt{exD}$ was taken. In this way we model "how" to decide the truth values of the propositions in $\mathcal{DP}$. All other transitions remain the same of those given for CSTNUs.

5 Automated Planning: A Tool for the Experimental Evaluation

We made a tool[3] for CSTNUDs which takes as input a CSTNUD specification and allows for the automated encoding into the corresponding UPPAAL-TIGA specification as well as execution simulation. To get the UPPAAL-TIGA specification we run `./Cstnud Network.cstnud --encode TGA.xml`, where `Network.cstnud` is the CSTNUD specification and `TGA.xml` the encoding into a TGA the tool returns in output. To synthesize a strategy we use UPPAAL-TIGA by querying the TGA with `verifytga -s -q -w0 TGA.xml dc.q > strategy`, where `dc.q` contains the TCTL query `control: A[] not tga.win` and

[3] Available at `http://regis.di.univr.it/TIME2017.tar.gz` along with the case studies of this paper and further 1000 randomly generated CSTNUDs as an initial set of benchmarks.

```
$ ./Cstnud ...
P = 0.1
p = true
A1 = 1.1
C1 = 6.0
D = 7.0
d = true
A3 = 8.0
C3 = 11.7
E = 12.7
Verifying ... SAT!
```

(a) `Player1` observes p and $C_1 = 6$, therefore decides d.

```
$ ./Cstnud ...
P = 0.1
p = false
A2 = 1.1
C2 = 9.2
D = 10.2
d = true
A3 = 11.2
C3 = 14.7
E = 15.7
Verifying ... SAT!
```

(b) `Player1` observes $\neg p$ and $C_2 = 9.2$, therefore decides d.

```
$ ./Cstnud ...
P = 0.1
p = false
A2 = 1.1
C2 = 12.7
D = 13.7
d = false
A4 = 14.7
C4 = 21.5
E = 22.5
Verifying ... SAT!
```

(c) `Player1` observes $\neg p$ and $C_2 = 12.7$, therefore decides $\neg d$.

Figure 5 Execution simulations for Figure 3b (`./Cstnud Fig3b.cstnud --execute Fig3b.s`).

`strategy` is the memoryless execution strategy that UPPAAL-TIGA spits out. To execute a controllable CSTNUD we run `./Cstnud Network.cstnud --execute strategy`.

We encoded the CSTNUDs in Figure 3a, Figure 3b and Figure 3c to get the UPPAAL-TIGA specifications. We ran UPPAAL-TIGA on such specifications. We used a Linux virtual machine run on top of a VMWare ESXi hypervisor using a physical machine equipped with an Intel i7 2.80GHz and 20GB of RAM for the experimental evaluation. The VM was assigned 5GB of RAM and full CPU power. For Figure 3a the analysis took 2 minutes and 4 seconds answering `Property is satisfied` and spitting out an execution strategy of 68K for `Player2`. For both Figure 3b and Figure 3c the analysis took 1 minute and 53 second spitting out a strategy of 44K for `Player1`. Finally, we executed the latter two controllable cases. The simulator correctly scheduled all non contingent time points satisfying all constraints. We show the output of a few simulations for Figure 3b in Figure 5.

Furthermore, we randomly generated 1000 CSTNUDs as an initial set of benchmarks and ran the analysis on those networks imposing a time out of 900 seconds each. The analysis proved that 169 networks were DC and 14 non-DC. The remaining networks reached the timeout limit. Each CSTNUD proved DC was correctly executed.

6 Correctness of the Encoding

We prove the correctness and discuss the complexity of the encoding provided in Section 4.

▶ **Theorem 16.** *The encoding in Section 4 is correct.*

Proof. To prove that we start by showing that the encoding in Section 4 correctly models the move-based semantics of Section 3. A state of the TGA is given by a pair $(L, \vec{c})$, where L is a locations and $\vec{c}$ represents the values of all clocks. The state of a CSTNUD during execution is given by its execution sequence Z. We show that the game interplay correctly models the continuous game evolution given in Definition 11 for all $t > 0$. We exclude the case for $t = 0$, so `Player1` does not play in $T_1(0)$ and `Player2` does not play in $T_2(0)$.

(Invariant) At any instant $t > 0$ the snapshot $Z(t) = \langle E, K, S_E, S_K \rangle$ corresponds to a state of the TGA $(L, \vec{c})$ in which $L = \mathtt{T}_2$ and $\vec{c}$ is as follows: $\hat{\mathtt{c}} = t$, $\mathtt{c}_\delta = 0$, for each $X \in \mathcal{T}$, $\mathtt{cX} < \hat{\mathtt{c}}$ and $\hat{\mathtt{c}} - \mathtt{cX} = k$ (if $X \in E \wedge S_E(X) = k$), $\mathtt{cX} = \hat{\mathtt{c}}$ otherwise. For each $p \in \mathcal{P}$, $\mathtt{cP} < \hat{\mathtt{c}} \wedge \mathtt{bP} = \hat{\mathtt{c}}$

(if $p \in K$ and $S_K(p) = \top$) and $\texttt{cP} < \hat{\texttt{c}} \wedge \texttt{bP} < \hat{\texttt{c}}$ (if $p \in K$ and $S_K(p) = \top$), $\texttt{cP} = \texttt{bP} = \hat{\texttt{c}}$ otherwise. Finally, `Player2` has finished taking controllable transitions at t.

When $t = 0$ (i.e., $\hat{\texttt{c}} = 0$) `Player2` cannot play in $\texttt{T}_2$ as no controllable transition is enabled. `Player1` cannot play either because the current location is not $\texttt{T}_1$ and he can only got there after a positive amount of time has elapsed. Therefore, at $t = 0$, $Z(0) = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$.

When $t > 0$ (i.e., $\hat{\texttt{c}} > 0$) both `Player1` and `Player2` can play in their respective turns $T_1(t)$ and $T_2(t)$. `Player1` can take `gain` to enter $\texttt{T}_1$ at time t. `Player2` cannot prevent him from doing so because `gain`, being urgent, has priority over any other controllable transition that `Player2` could take at that time. So, `Player1` plays first. Once got in $\texttt{T}_1$, `Player1` can take (in general) a non-empty sequence of transitions to execute a few non contingent time points and decide the truth values of some decidable propositions if he has executed some decision time points. Such a sequence is finite since there is a finite number of time points to execute and a finite number of propositions to assign. Furthermore, each time point (resp., proposition) can be executed (resp., assigned a value) only once. When this sequence of transitions is over, `Player1` ends his turn by taking `pass` to lead the run back to $\texttt{T}_2$. Since $\texttt{T}_1$ is urgent, time has not elapsed. Therefore, the sequence of transitions taken at $\texttt{T}_1$ corresponds to the sequence of moves made by `Player1` in $T_1(t)$. Instead, if `Player1` wants to wait at time t, he can either take `gain` and `pass` immediately after or just avoid taking `gain`. Now, at $\texttt{T}_2$, `Player2` does the same for contingent time points and observable propositions if some observation time points have been executed by `Player1` in $T_1(t)$. When `Player2` is done, the sequence of transitions taken, models the sequence of moves made in $T_2(t)$. Since `Player2` does not make any other move in $T_2(t)$, $Z(t)$ can no longer be modified.

`Player1` and `Player2` are driven by their strategies σ_1 and σ_2 which say what moves to make (i.e., transitions to take) in $T_1(t)$ and $T_2(t)$ at any time t depending on the current Z. The purpose of σ_1 is to keep $Z(t)$ locally consistent, whereas that of σ_2 is the opposite.

The strategies also satisfy their applicability conditions as `Player1` can make his moves in $T_1(t)$ according to σ_1 iff `Player2` has not played yet in $T_2(t)$, whereas `Player2` can make his moves in $T_2(t)$ according to σ_2 iff either `Player1` has not played in $T_1(t)$ or `Player2` is not done in $T_2(t)$. We have already proved that for any $t > 0$, `Player1` plays first.

The strategies satisfy their obligations as each time `Player1` executes a decision time point $D!$, he also assigns the associated decidable proposition d a truth value as well. This occurs at the same time but sequentially after the execution of $D!$. `Player1` assigns $\top$ to d by not taking `dFalse` and assigns $\bot$ to d by taking `pFalse`. If `Player1` takes the transition then he cannot take it again in the same turn (as the guard of `pFalse` invalidates). If he does not, then he will never be able to take `dFalse` in any $T_1(t')$ where $t' > t$. Likewise, σ_2 satisfies its similar obligation for observable propositions. Furthermore, σ_2 also satisfies the obligation regarding contingent time points as the encoding generates a `failC` transition for each contingent time point C (belonging to a $(A, x, y, C) \in \mathcal{L}$) allowing `Player1` to move to `win` if `Player2` does not take `exC` within its maximum upper bound y. Since `Player2` wants to prevent `Player1` from getting to `win`, σ_2 is obliged to schedule C such that $C - A \in [x, y]$.

Both σ_1 and σ_2 satisfy their postconditions: the reset of `cX` clocks says when the time points were executed, whereas the values of `bP` clocks say what truth values the propositions have been assigned. Finally, winning conditions are modeled differently with respect to the player. For `Player1` they are abstracted as a winning path checking that all time points and constraints whose labels are not falsified by ℓ_{cps} have been executed and satisfied, respectively. For `Player2` winning conditions correspond to schedule a contingent time point at a particular time or decide a truth value for an observable propositions (or any combination of these moves) such that `Player1` is unable to satisfy at least one constraint and ends up blocked somewhere while going through the winning path before entering `win`. ◀

▶ **Theorem 17.** *Any CSTNUD can be encoded into a TGA in polynomial time.*

Proof Sketch. Our encoding subsumes that for CSTNUs which runs in polynomial time [4, 5]. We "worsen" that encoding by adding a `dFalse` transition for each $d \in \mathcal{DP}$. For each $X \in \mathcal{T}$, $L_{enc}(X)$ and $\Pi_{enc}(X)$ are computed in polynomial time by analyzing $L(X)$ and $\mathcal{C}$. ◀

7 Related Work

STNs [10] and Drake [9] differ from CSTNUDs since they do not specify any uncontrollable part. Therefore, they are incomparable with CSTNUDs.

STNUs [18] specify contingent durations as the unique uncontrollable part. The execution of non-contingent time points cannot influence any contingent duration. Instead, contingent durations do influence the real-value assignment to the non-contingent time points. However, such durations never prevent any non-contingent time point from being executed. This work also addresses the influence of the controllable part over the uncontrollable one.

CSTNs [14, 20] specify conditional constraints as the unique uncontrollable part. Again, the execution of non-contingent time points cannot prevent any truth value assignment from happening. Instead, depending on what truth value a propositional variable is assigned some time point might be excluded, runtime, from the execution of the network. Similar explanations hold for CSTNUs [7, 13] which merge CSTNs and STNUs. CSTNUDs are also able to prevent uncontrollable truth value assignments and durations from happening.

In [3] CSTNs are extended with decision nodes regulating the truth value assignments to some propositions under control. That work focuses on the complexity analysis of the DC-checking problem and provides constraint-propagation algorithms for special cases in which either the network specifies only decisions and no observations or all decisions are made before any observation. Moreover, contingent durations are not addressed. This work follows a complete different direction starting from CSTNUs and it is based on TGAs.

In temporal workflow management, the difference between controllable and uncontrollable XOR splits is introduced in [11] and a technique based on PERT-nets computes internal activity deadlines in order to meet the global ones. Some missed deadlines require human interaction for recovery. We rely on DC, which guarantees that we never miss any deadline.

In [19] UPPAAL-TIGA is used to synthesize a controller for timeline-based plans which consider multivalued state variables and networks of TGAs. Apart from time points, our variables are Boolean and our encoding involves one TGA only.

8 Conclusions and Future Work

We defined *conditional simple temporal networks with uncertainty and decisions* (CSTNUDs) as a unified formalism. CSTNUDs implicitly embed all minor temporal network formalisms such as STNs (if $\mathcal{L} = \mathcal{OT} = \mathcal{DT} = \emptyset$), CSTNs (if $\mathcal{L} = \mathcal{DT} = \emptyset$), STNUs (if $\mathcal{OT} = \mathcal{DT} = \emptyset$), CSTNUs (if $\mathcal{DT} = \emptyset$), STNDs (if $\mathcal{L} = \mathcal{OT} = \emptyset$), CSTNDs (if $\mathcal{L} = \emptyset$), and STNUDs (if $\mathcal{OT} = \emptyset$). We modeled the DC-checking of a CSTNU as a two-player game where `Player1` models the controller and `Player2` models the environment and gave the execution semantics in move-based strategies. We provided an encoding from CSTNUDs into TGAs as an optimized extension of that given for CSTNUs and discussed the correctness and complexity of such an encoding. We automated the approach by making a tool we used to analyze and simulate the execution of the examples discussed in this paper. We also provided a preliminary experimental evaluation of the approach against a set of 1000 randomly generated CSTNUDs. As future work, we plan to address weak and strong controllability of CSTNUDs.

References

1 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994. doi:10.1016/0304-3975(94)90010-8.

2 Gerd Behrmann, Agnès Cougnard, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Uppaal-tiga: Time for playing games! In *CAV 2007*. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-73368-3_14.

3 Massimo Cairo, Carlo Combi, Carlo Comin, Luke Hunsberger, Roberto Posenato, Romeo Rizzi, and Matteo Zavatteri. Incorporating decision nodes into conditional simple temporal networks. In S. Schewe, T. Schneider, and J. Wijsen, editors, *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*, volume 91 of *LIPIcs*, pages 9:1–9:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.TIME.2017.9.

4 Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Sound and complete algorithms for checking the dynamic controllability of temporal networks with uncertainty, disjunction and observation. In *21st International Symposium on Temporal Representation and Reasoning (TIME 2014)*, pages 27–36, 2014. doi:10.1109/TIME.2014.21.

5 Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Dynamic controllability via timed game automata. *Acta Informatica*, 53(6-8):681–722, 2016. doi:10.1007/s00236-016-0257-2.

6 Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, and Marco Roveri. Using timed game automata to synthesize execution strategies for simple temporal networks with uncertainty. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27-31, 2014, Québec City, Québec, Canada.*, pages 2242–2249, 2014. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8512.

7 Carlo Combi, Luke Hunsberger, and Roberto Posenato. An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty. In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence – Volume 2: ICAART*, pages 144–156. INSTICC, ScitePress, 2013. doi:10.5220/0004256101440156.

8 Carlo Combi, Roberto Posenato, Luca Viganò, and Matteo Zavatteri. Access controlled temporal networks. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence – Volume 2: ICAART*, pages 118–131. INSTICC, ScitePress, 2017. doi:10.5220/0006185701180131.

9 Patrick R. Conrad and Brian C. Williams. Drake: An efficient executive for temporal plans with choice. *J. Artif. Int. Res.*, 42(1):607–659, September 2011.

10 Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. doi:10.1016/0004-3702(91)90006-6.

11 Johann Eder, Euthimios Panagos, Heinz Pozewaunig, and Michael Rabinovich. *Time Management in Workflow Systems*, pages 265–280. Springer London, 1999. doi:10.1007/978-1-4471-0875-7_22.

12 John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

13 Luke Hunsberger, Roberto Posenato, and Carlo Combi. The Dynamic Controllability of Conditional STNs with Uncertainty. In *Workshop on Planning and Plan Execution for Real-World Systems (PlanEx) at ICAPS-2012*, pages 1–8, Atibaia, June 2012. URL: http://arxiv.org/abs/1212.2005.

14 Luke Hunsberger, Roberto Posenato, and Carlo Combi. A sound-and-complete propagation-based algorithm for checking the dynamic consistency of conditional simple temporal networks. In *22st International Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 4–18, 2015. `doi:10.1109/TIME.2015.26`.

15 Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems. In Ernst W. Mayr and Claude Puech, editors, *STACS'95: 12th Annual Symposium on Theoretical Aspects of Computer Science Munich, Germany, March 2–4, 1995 Proceedings*, pages 229–242, Berlin, Heidelberg, 1995. Springer. `doi:10.1007/3-540-59042-0_76`.

16 Paul Morris. The mathematics of dispatchability revisited. In *Proceedings of the Twenty-Sixth International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'16, pages 244–252. AAAI Press, 2016.

17 Paul Morris and Nicola Muscettola. Temporal Dynamic Controllability Revisited. In *Proceedings of the Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, pages 1193–1198. AAAI Pr., 2005.

18 Paul H. Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal uncertainty. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 494–502, 2001.

19 Andrea Orlandini, Alberto Finzi, Amedeo Cesta, and Simone Fratini. TGA-Based Controllers for Flexible Plan Execution. In Joscha Bach and Stefan Edelkamp, editors, *KI 2011: Advances in Artificial Intelligence: 34th Annual German Conference on AI, Berlin, Germany, October 4-7,2011. Proceedings*, pages 233–245. Springer Berlin Heidelberg, 2011. `doi:10.1007/978-3-642-24455-1_22`.

20 Ioannis Tsamardinos, Thierry Vidal, and Martha E. Pollack. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints*, 8(4):365–388, 2003. `doi:10.1023/A:1025894003623`.