Incorporating Decision Nodes into Conditional Simple Temporal Networks

Massimo Cairo[1], Carlo Combi[2], Carlo Comin[2], Luke Hunsberger[3], Roberto Posenato[2], Romeo Rizzi[2], and Matteo Zavatteri[2]

1 **Department of Mathematics, University of Trento, Trento, Italy**
massimo.cairo@unitn.it
2 **Department of Computer Science, University of Verona, Verona, Italy**
{carlo.combi,carlo.comin,roberto.posenato,
romeo.rizzi,matteo.zavatteri}@univr.it
3 **Department of Computer Science, Vassar College, Poughkeepsie, NY, USA**
hunsberger@vassar.edu

Abstract

A Conditional Simple Temporal Network (CSTN) augments a Simple Temporal Network (STN) to include special time-points, called observation time-points. In a CSTN, the agent executing the network controls the execution of every time-point. However, each observation time-point has a unique propositional letter associated with it and, when the agent executes that time-point, the environment assigns a truth value to the corresponding letter. Thus, the agent observes, but does not control the assignment of truth values. A CSTN is dynamically consistent (DC) if there exists a strategy for executing its time-points such that all relevant constraints will be satisfied no matter which truth values the environment assigns to the propositional letters.

Alternatively, in a Labeled Simple Temporal Network (Labeled STN)—also called a Temporal Plan with Choice—the agent executing the network controls the assignment of values to the so-called choice variables. Furthermore, the agent can make those assignments at any time. For this reason, a Labeled STN is equivalent to a Disjunctive Temporal Network.

This paper incorporates both of the above extensions by augmenting a CSTN to include not only observation time-points but also decision time-points. A decision time-point is like an observation time-point in that it has an associated propositional letter whose value is determined when the decision time-point is executed. It differs in that the agent—not the environment—selects that value. The resulting network is called a CSTN with Decisions (CSTND). This paper shows that a CSTND generalizes both CSTNs and Labeled STNs, and proves that the problem of determining whether any given CSTND is dynamically consistent is PSPACE-complete. It also presents algorithms that address two sub-classes of CSTNDs: (1) those that contain only decision time-points; and (2) those in which all decisions are made before execution begins.

1998 ACM Subject Classification G.2.2 Graph Theory, I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases Conditional Simple Temporal Networks with Decisions, Dynamic Consistency, SAT Solver, Hyper Temporal Networks, PSPACE

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.9

1 Introduction

Temporal networks have long been employed for the representation, validation, and execution of plans affected by temporal constraints [1, 5, 8, 10, 11, 17, 23]. A temporal network constains *time-points* and *temporal constraints*. Time-points are real-valued variables; temporal

constraints are binary difference constraints that specify lower or upper bounds on the temporal distance between pairs of time-points [16]. The *execution* of a time-point (i.e., the assignment of a real value to it) models the (instantaneous) occurrence of an event. An agent executing a temporal network aims to execute its time-points so that all relevant temporal constraints are satisfied.

A *Simple Temporal Network* (STN) is the most studied and used kind of temporal network due to its simplicity, efficiency, and general applicability [16]. An STN is typically used in planning applications where all time-points must be executed (i.e., must play their role in the plan) and where the agent controls their execution. An STN is *consistent* if the network can be executed in such a way that all of its constraints are satisfied.

Since STNs were proposed, several authors have introduced extensions to STNs to augment their expressiveness. Among them, we mention here: (i) *Simple Temporal Networks with Uncertainty* (STNUs) [24], (ii) *Conditional Simple Temporal Networks* (CSTNs) [22, 25], and (iii) *Conditional Simple Temporal Networks with Uncertainty* (CSTNUs) [21].

A *Simple Temporal Network with Uncertainty* extends an STN by incorporating *contingent links* to model uncontrollable, but bounded temporal durations. A contingent link has the form (A, x, y, C), where A is the *activation* time-point, x and y are real numbers such that $0 < x < y < \infty$, and C is the *contingent* time-point. Typically, the agent controls the execution of A, but once A has been executed, the execution of C is beyond the agent's control because the time of its execution is decided by the environment. Indeed, C may be executed at any time such that $x \leq C - A \leq y$. Thus, the agent merely observes the execution of C when it occurs. The agent aims to execute the time-points under its control such that all constraints will be satisfied no matter how the contingent durations turn out. Crucially, the agent may *react* to the contingent durations it observes, adapting its execution of the remaining unexecuted time-points (i.e., it may use a *dynamic* execution strategy [24]).

A *Conditional Simple Temporal Network* extends an STN in a different direction by specifying the time-points/constraints must be executed/satisfied in various *scenarios*. Each scenario is represented by a conjunction of (positive or negative) propositional literals, where each proposition represents some condition. The agent executing a CSTN aims to execute the time-points relevant to the unfolding scenario, while satisfying all of the relevant constraints. Each propositional letter p has a corresponding *observation time-point* $P?$ When the agent executes $P?$, the environment sets the value of p. Since these values are only observed in real time, the agent aims to satisfy all relevant constraints no matter how the conditions turn out during execution [21, 25]. Hence, for a CSTN, the agent typically uses a *dynamic* execution strategy with respect to the uncontrollable condition values [22].

A *Conditional Simple Temporal Network with Uncertainty* generalizes both STNUs and CSTNs in order to deal with both kinds of uncertainties simultaneously [21].

For temporal networks with uncontrollable features (e.g., STNUs, CSTNs and CSTNUs), dynamic properties, named *dynamic controllability* or *dynamic consistency*, have been studied and some related algorithms have been proposed. These properties specify whether it is possible to execute all time-points under the agent's control while satisfying all relevant constraints, by dynamically reacting to the uncertainties (whether the duration of a contingent link or the setting of a boolean value) as they are revealed in real time.

Conrad and Williams [14] present *Drake,* a dynamic executive for temporal plans that include discrete choices. In Drake, the executive sets the values for the propositional letters—hence the name *choice*—and the goal of the system is to both schedule events and make discrete choices as the execution unfolds. The ability to make discrete choices enriches an executive by offering it the ability to order activities, and choose between alternate methods

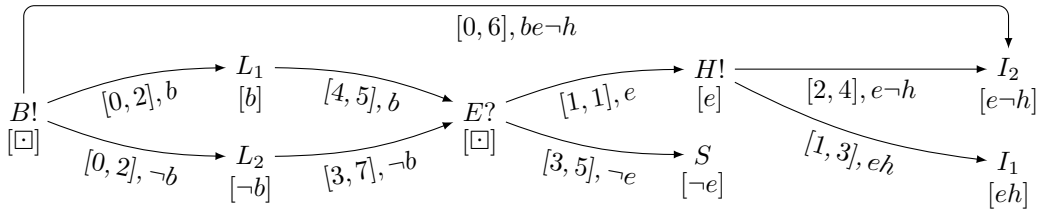

Figure 1 A CSTN with Decisions modeling the example discussed in the text. The propositions b, e and h correspond to the time-points $B!, E?$ and $H!$, respectively.

(sub-plans) for achieving goals. Consistency analysis aims to determine a set of choices that will enable the executive to satisfy the relevant constraints.

Thus, STNUs, CSTNs and CSTNUs address uncontrollable parts, whereas Drake addresses controllable parts only. No temporal network discussed so far has addressed the arising interplay that occurs when controllable and uncontrollable conditions may mutually influence one another. This paper focuses on this issue and makes the following contributions.

1. A new model, *Conditional Simple Temporal Network with Decision* (CSTND), that accommodates contingent propositional variables (conditions) and *controllable propositional variables* (decisions). During execution, the decisions made by the agent, together with the conditions specified by the environment, determine the unfolding scenario.
2. A proof that the decision problem of establishing whether or not any CSTND is dynamically consistent is PSPACE-complete.
3. Algorithms addressing two sub-classes of CSTNDs: (i) those containing decisions only, and (ii) those in which all decisions must be set before starting to execute the network.

2 Motivating Example

Fig. 1 depicts a simplification of an example from the healthcare domain. The nodes in the graph represent time-points; the edges represent constraints. Time-points and constraints are relevant whenever their propositional labels are consistent with the unfolding scenario. Temporal ranges are in minutes. For instance, the annotation $[0, 2], b$ on the edge from $B!$ to L_1 represents that in scenarios where b is true, the difference $L_1 - B!$ must be in the interval $[0, 2]$ (i.e., L_1 must be executed between zero and two minutes after $B!$).

The plan applies to patients suffering from hematological diseases. It starts by having a patient's blood tested. There are two labs in the hospital, $\texttt{Lab}_1$ and $\texttt{Lab}_2$, but only one of them will analyze the blood sample, to be determined by the value the executing agent assigns to b, when $B!$ is executed. If b is $\top$ (resp., $\bot$), then $\texttt{Lab}_1$ (resp., $\texttt{Lab}_2$) will analyze the sample. The execution of either L_1 or L_2 models this task. Depending on the result of the blood test, a physician (who represents the environment in this example) evaluates whether the patient needs urgent care. This evaluation is represented by the time-point $E?$. The environment setting $e = \bot$ represents that the physician determined that the patient does not need urgent care. In that case, the plan concludes with a standard treatment, represented by the execution of S. However, if $e = \top$, then a hospitalization process is engaged, represented by the time-points $H!, I_1$ and I_2. There are two Intensive Care Units in the hospital, $\texttt{ICU}_1$ and $\texttt{ICU}_2$, but only one of them will be used, depending on the value for h, which is determined when $H!$ is executed. If $h = \top$ (resp., $h = \bot$), then the patient is hospitalized in $\texttt{ICU}_1$ (resp., $\texttt{ICU}_2$). Note that if $b = e = \top$, and $h = \bot$, then I_2 must be executed no more than 6 minutes after $B!$, the time-point modeling the start of the plan.

Our goal is to determine whether such networks are dynamically consistent. Note that if the values of b and h were set by the environment, then the network in Fig. 1 would be inconsistent. For example, in the scenario $be\neg h$, the lower bound of I_2 is 7, which is inconsistent with the constraint $B!\xrightarrow{[0,6],\, be\neg h} I_2$. However, if b and h are set by the agent, then the plan is dynamically consistent. Indeed, there are two significant possibilities: if $b = \bot$ and $e = \top$, then h can be $\top$ or $\bot$ without any problem; but if $b = e = \top$, then h must be $\top$, because the scenario $be\neg h$ would introduce the same inconsistency discussed above.

3 Conditional Simple Temporal Network with Decisions

This section introduces the formal definitions of a *Conditional Simple Temporal Network with Decisions* (CSTND) and the corresponding *dynamic consistency* property. We begin by recalling *Simple Temporal Networks* (STNs) [16], a well-known model for representing and reasoning about temporal constraints. An STN is a pair $(\mathcal{T}, \mathcal{C})$, where $\mathcal{T}$ is a set of real-valued variables, called *time-points,* and $\mathcal{C}$ is a set of binary constraints on those variables, each having the form, $(Y - X \leq \delta)$, where $X, Y \in \mathcal{T}$ and $\delta \in \mathbb{R}$. A constraint $(Y - X = \delta)$ can be represented by the constraints, $(Y - X \leq \delta)$ and $(X - Y \leq -\delta)$. The *Simple Temporal Problem* (STP) is that of determining whether an STN is consistent (i.e., has a solution).

Tsamardinos et al. [25] extended STNs to include time-points and temporal constraints that apply only in certain *scenarios*, where each scenario is represented by a conjunction of propositional literals. In their work, each time-point has a *label* that concisely specifies the scenarios in which that time-point must be executed. During execution, the execution of so-called *observation time-points* non-deterministically generates truth values for the corresponding propositional variables. Thus, the scenario is incrementally revealed. Later, Hunsberger et al. [21, 22] augmented their model to also allow constraints to have labels that specify the scenarios in which they must be satisfied. The result was a *Conditional STN* (CSTN). A CSTN is called *dynamically consistent* if there is a strategy for executing its time-points such that all relevant constraints will be satisfied no matter which scenario is incrementally revealed. They also formalized several well-definedness properties that had been only informally expressed in the earlier work.

This paper generalizes a CSTN by allowing some of the propositional variables to be assigned values not by the environment, but by the agent executing the network.

▶ **Definition 1** (Label)**.** Let $\mathcal{P}$ be a set of propositional letters. A *label* ℓ over $\mathcal{P}$ is a (possibly empty) conjunction, $\ell = l_1 \wedge \cdots \wedge l_k$, of (positive or negative) literals $l_i \in \{p_i, \neg p_i\}$ on distinct variables $p_i \in \mathcal{P}$. The empty label is denoted by $\boxdot$. $\mathcal{P}^*$ denotes the set of all labels over $\mathcal{P}$.

▶ **Definition 2** (CSTND)**.** A *Conditional Simple Temporal Network with Decisions* (CSTND) is a tuple $\Gamma = \langle \mathcal{T}, \mathcal{P}, \mathcal{CP}, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ where:

- $\mathcal{T}$ is a finite set of *temporal variables* or *time-points*;
- $\mathcal{P}$ is a finite set of *propositional letters/variables*;
- $(\mathcal{CP}, \mathcal{DP})$ is a partition of $\mathcal{P}$ into *contingent propositional variables (conditions)* $\mathcal{CP}$ and *controllable propositional variables (decisions)* $\mathcal{DP}$;
- $\mathcal{C}$ is a finite set of *labeled constraints,* each of the form, $(Y - X \leq \delta, \ell)$, where $X, Y \in \mathcal{T}$, $\delta \in \mathbb{R}$, and $\ell \in \mathcal{P}^*$;
- $\mathcal{OT} \subseteq \mathcal{T}$ is the set of *disclosing time-points;* and
- $\mathcal{O} \colon \mathcal{P} \to \mathcal{OT}$ is a bijection that associates each propositional variable $p \in \mathcal{P}$ to a disclosing time-point $\mathcal{O}(p) \in \mathcal{OT}$. If $p \in \mathcal{CP}$, then its disclosing time-point is called an *observation time-point;* but if $p \in \mathcal{DP}$, its disclosing time-point is called a *decision time-point.*

When an observation time-point is executed, the corresponding propositional variable (condition) is assigned a truth value by the environment; however, when a decision time-point is executed, the corresponding variable (decision) is assigned a truth value by the agent. To highlight the correspondence between propositional variables and their disclosing time-points, a decision time-point for any variable p is notated as $P!$, while an observation time-point for any variable q is notated as $Q?$, as illustrated in Fig. 1.

It is worth pointing out that in the definition of CSTND only constraints are labeled, not time-points. This restriction does not limit the expressivity of CSTNDs, as it has been recently shown that labeling constraints is sufficient to represent any possible CSTN [3]. Moreover, since time-points in CSTNDs do not have labels, the well-definedness properties formalized by Hunsberger et al. [22] become vacuous, which simplifies subsequent definitions.

To facilitate defining the dynamic consistency property for CSTNDs, we refine the supporting definitions of scenarios, projections and schedules to distinguish condition variables from decision variables.

▶ **Definition 3** (Scenario)**.** A (combined) *scenario* over $\mathcal{P}$ is a total assignment $s\colon \mathcal{P} \to \{\bot, \top\}$ of truth values to propositional variables. Each scenario s also determines a truth value for each label $\ell \in \mathcal{P}^*$. If $s(\ell) = \top$, we may write $s \vDash \ell$. The set of all scenarios over $\mathcal{P}$ is denoted by $\Sigma_{\mathcal{P}}$. When a scenario is restricted to the subset $\mathcal{CP} \subseteq \mathcal{P}$ of conditions, then it may be called a *condition scenario;* similarly, when a scenario is restricted to the subset $\mathcal{DP} \subseteq \mathcal{P}$ of decisions, it may be called a *decision scenario.* The sets of all condition and decision scenarios are denoted by $\Sigma_{\mathcal{CP}}$ and $\Sigma_{\mathcal{DP}}$, respectively.

A CSTND projection is an STN that contains the constraints applicable in a given scenario.

▶ **Definition 4** (Projection)**.** Let $\Gamma = \langle \mathcal{T}, \mathcal{P}, \mathcal{CP}, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ be any CSTND, and let s be any (combined) scenario. The *projection* of Γ over s is the STN $\Gamma_s = (\mathcal{T}, \mathcal{C}_s)$ where $\mathcal{C}_s = \{Y - X \leq \delta \mid (Y - X \leq \delta, \ell) \in \mathcal{C} \text{ and } s \vDash \ell\}$.

▶ **Definition 5** (Schedule)**.** A *schedule* over a set $\mathcal{T}$ of time-points is a total assignment $\psi\colon \mathcal{T} \to \mathbb{R}$ of real values to those time-points. Hereinafter, the value $\psi(X)$ will be notated as $[\psi]_X$. A schedule ψ over $\mathcal{T}$ is said to be *feasible* for an STN $(\mathcal{T}, \mathcal{C})$ if ψ satisfies all of the constraints in $\mathcal{C}$. The set of schedules over $\mathcal{T}$ is denoted by $\Psi_{\mathcal{T}}$.

The agent executing the network must (1) schedule time-points, and (2) assign values to decision variables. In addition, the agent is able to *react* in real time to conditions set by the environment. Therefore, we define a *two-part* execution *strategy,* as follows.

▶ **Definition 6** (Execution Strategy)**.** A *temporal strategy* for a CSTND Γ is a function $\sigma^{\mathrm{t}}\colon \Sigma_{\mathcal{CP}} \to \Psi_{\mathcal{T}}$ that maps each *condition scenario* $cs \in \Sigma_{\mathcal{CP}}$ to a (complete) schedule $\sigma^{\mathrm{t}}(cs)$ over $\mathcal{T}$. A *decision strategy* for Γ is a function $\sigma^{\mathrm{d}}\colon \Sigma_{\mathcal{CP}} \to \Sigma_{\mathcal{DP}}$ that maps each condition scenario $cs \in \Sigma_{\mathcal{CP}}$ to a decision scenario $ds = \sigma^{\mathrm{d}}(cs) \in \Sigma_{\mathcal{DP}}$. An *execution strategy* is a pair $\sigma = (\sigma^{\mathrm{t}}, \sigma^{\mathrm{d}})$ where σ^{t} is a temporal strategy and σ^{d} is a decision strategy. An *execution strategy* $\sigma = (\sigma^{\mathrm{t}}, \sigma^{\mathrm{d}})$ is *viable* if, for every condition scenario $cs \in \Sigma_{\mathcal{P}}$, letting $ds = \sigma^{\mathrm{d}}(cs)$ and $s = cs \cup ds$, the schedule $\sigma^{\mathrm{t}}(cs)$ is feasible for the projection Γ_s.

To ensure that the schedules and decisions generated by an execution strategy only depend on *past* observations, a *dynamic* execution strategy is subject to restrictions expressed in terms of *condition scenario histories,* as follows.

▶ **Definition 7** (Condition Scenario History)**.** Given a temporal strategy σ^{t}, a condition scenario $cs \in \Sigma_{\mathcal{CP}}$, and a time value $t \in \mathbb{R}$, the *condition scenario history* at t in the

condition scenario cs for the temporal strategy σ^{t}—notated as $scHst(t, cs, \sigma^{\mathrm{t}})$—is the set of contingent variable assignments made by the environment *before* time t according to the schedule $\sigma^{\mathrm{t}}(cs)$: $scHst(t, cs, \sigma^{\mathrm{t}}) = \{(p, cs(p)) \mid p \in \mathcal{CP} \text{ and } [\sigma^{\mathrm{t}}(cs)]_{\mathcal{O}(p)} < t\}$.

▶ **Definition 8** (Dynamic Execution Strategy)**.** A temporal strategy σ^{t} is *dynamic* if for any pair of condition scenarios $cs \in \Sigma_{\mathcal{CP}}$ and $cs' \in \Sigma_{\mathcal{CP}}$, and any time-point $X \in \mathcal{T}$:

let: $t = [\sigma^{\mathrm{t}}(cs)]_X$,
if: $scHst(t, cs, \sigma^{\mathrm{t}}) = scHst(t, cs', \sigma^{\mathrm{t}})$,
then: $[\sigma^{\mathrm{t}}(cs')]_X = t$.

Similarly, a decision strategy σ^{d} is *dynamic* if for any condition scenarios $cs \in \Sigma_{\mathcal{CP}}$ and $cs' \in \Sigma_{\mathcal{CP}}$, and any decision variable $p \in \mathcal{DP}$:

let: $t = [\sigma^{\mathrm{t}}(cs)]_{\mathcal{O}(p)}$,
if: $scHst(t, cs, \sigma^{\mathrm{t}}) = scHst(t, cs', \sigma^{\mathrm{t}})$,
then: $\sigma^{\mathrm{d}}(cs)(p) = \sigma^{\mathrm{d}}(cs')(p)$.

An execution strategy is *dynamic* if its temporal and decision strategies are both dynamic.

Now, it is possible to formally introduce the concept of dynamic consistency for CSTNDs.

▶ **Definition 9** (Dynamic Consistency)**.** A CSTND is *dynamically consistent* (DC) if it admits a dynamic and viable execution strategy. The *CSTND-DC problem* is that of checking whether any given CSTND is dynamically consistent.

4 Computational Complexity of the CSTND-DC Problem

This section shows that the CSTND-DC problem is PSPACE-complete.

4.1 PSPACE-hardness

Cairo and Rizzi [4] showed that the DC-checking problem for CSTNs is PSPACE-hard. This section presents a simpler proof of that result by providing a direct reduction from the Quantified Boolean Formula (QBF) problem to the CSTND-DC problem.

Consider any quantified boolean formula of the form, $\Phi = \exists x_1 \forall y_1 \cdots \exists x_n \forall y_n \; \varphi$, where φ is a formula in conjunctive normal form, each clause of which is limited to at most three literals over a finite set of propositional variables $x_1, y_1, \ldots, x_n, y_n$ (i.e., φ is a 3SAT formula). More specifically, φ is a conjunction of the form $\bigwedge_{j=1}^{m}(l_{j,1} \vee l_{j,2} \vee l_{j,3})$, where each literal $l_{j,k}$ is either a positive or negative instance of one of the quantified variables.

The reduction involves the construction of a corresponding CSTND instance Γ such that Γ is DC if and only if Φ is true. To begin, define the sets of decision and condition variables to be $\mathcal{DP} = \{x_1, \ldots, x_n\}$ and $\mathcal{CP} = \{y_1, \ldots, y_n\}$, respectively. Thus, the disclosing time-points of the network are given by $\mathcal{OT} = \{X_1!, \ldots, X_n!, Y_1?, \ldots, Y_n?\}$. The only other time-point, W, is discussed below. Thus, $\mathcal{T} = \mathcal{OT} \cup \{W\}$. Finally, the set of constraints is given by:

$$\mathcal{C} = \Big(\bigcup_{i=1,\ldots,n} \{(X_i! - Y_i? = -1, \boxdot), (Y_i? - X_{i+1}! = -1, \boxdot)\}\Big) \cup$$
$$\Big(\bigcup_{i=1,\ldots,n} \{(W - W \leq -1, \neg l_{i,1} \wedge \neg l_{i,2} \wedge \neg l_{i,3})\}\Big) \cup \{X_0! = 1, W = 2n + 1\}$$

The constraints, $(X_i! - Y_i? = -1, \boxdot)$ and $(Y_i? - X_{i+1}! = -1, \boxdot)$ for $i \in \{1, \ldots, n\}$, impose the order $X_1! < Y_1? < \ldots < X_n! < Y_n?$ on the disclosing time-points, which mirrors the order of the alternating quantifiers $x_1, y_1, \ldots, x_n, y_n$ in Φ. Furthermore, together with the constraint, $X_0! = 1$, they ensure that any viable temporal strategy must make the *fixed*

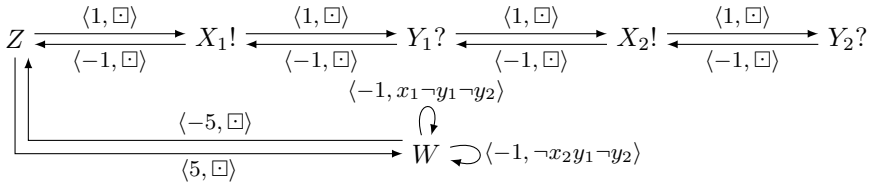

Figure 2 The CSTND obtained from the quantified boolean 3SAT formula $\Phi \equiv \exists x_1 \forall y_1 \exists x_2 \forall y_2 (\neg x_1 \vee y_1 \vee y_2) \wedge (x_2 \vee \neg y_1 \vee y_2)$

assignments, $X_0! = 1, Y_0? = 2, \ldots, X_n! = 2n-1, Y_n? = 2n$, across all (combined) scenarios. Note that any such temporal strategy is trivially dynamic.

The constraints of the form, $(W - W \leq -1, \neg l_{i,1} \wedge \neg l_{i,2} \wedge \neg l_{i,3})$ are negative self-loops on the extra time-point W, each of which has a label that is the logical negation of one of the clauses in the boolean formula φ.

Fig. 2 depicts a simple example of the reduction from a QBF to the corresponding CSTND, where $\Phi \equiv \exists x_1 \forall y_1 \exists x_2 \forall y_2 (\neg x_1 \vee y_1 \vee y_2) \wedge (x_2 \vee \neg y_1 \vee y_2)$.

In any (combined) scenario cs, the labeled negative self-loops at W are satisfied if and only if cs assigns the value $\bot$ to each of their labels. That happens if and only if the formula φ evaluates to $\top$ in the scenario cs. Therefore, Γ admits a viable execution strategy if and only if Φ is true.

Given the fixed ordering of the disclosing time-points, a decision strategy is dynamic if and only if the assignment of each decision variable x_i depends only on the *preceding* condition variables, $y_1, \ldots, y_{i-1}$. This mirrors the semantics of the nested quantifiers, where each x_i is existentially quantified, and each preceding y_j is universally quantified. Thus, Γ has a dynamic execution strategy if and only if Φ is true.

4.2 A Polynomial-Space Algorithm for the CSTND-DC Problem

This section presents a polynomial-space algorithm for the CSTND-DC problem, which extends the algorithm for CSTNs from prior work [4], assuming that time is discretized. As already discussed in [4], such assumption does not limit the generality of the algorithm. Together with the PSPACE-hard result from the preceding section, this proves that the CSTND-DC problem is PSPACE-complete.

We begin by showing that, under certain conditions, a DC CSTND admits a *discrete* execution strategy (i.e., one that only schedules time-points at rational multiples of some *fixed* real number ϵ, where the granularity of the rational factors is bounded). This result, whose proof is in the Appendix, adapts a similar result for CSTNs from prior work [4].

▶ **Lemma 10.** *Suppose that Γ is a CSTND such that, for some $\epsilon \in \mathbb{R}^+$ and $W \in \mathbb{Z}^+$, each constraint $(Y - X \leq w)$ in Γ satisfies $w = k\epsilon$ for some $k \in \mathbb{Z}$, where $-W < k < W$. If Γ is dynamically consistent, then Γ admits a viable and dynamic execution strategy $\sigma = (\sigma^{\mathrm{t}}, \sigma^{\mathrm{d}})$ such that for each scenario cs and each $X \in \mathcal{T}$, $[\sigma^{\mathrm{t}}(cs)]_X = k'\epsilon/K$, for some $k' \in \{0, 1, \ldots, 2K^2W\}$, where $K = |\mathcal{T}| \cdot 2^{|\mathcal{P}|}$.*

The execution of a CSTND can be viewed as a two-player game between the agent and the environment. The agent aims to make decisions and schedule time-points to satisfy all relevant constraints; the environment aims to assign values to conditions that will thwart the agent. Our DC-checking algorithm, whose pseudo-code is given in Algorithm 1, recursively explores all possible configurations of the game. Each configuration is a tuple $c = (t, \psi, h)$, where t is

Algorithm 1: CSTND Dynamic Consistency checking in polynomial space

```
1  Function DC(Γ)
       Input  : Γ is a CSTND satisfying the conditions of Theorem 10 for some values ε > 0 and
                W ∈ ℤ⁺
       Returns: true if Γ is dynamic consistent, false otherwise
2      c_0 ← (0, ∅, ∅)                                              ▷ Initial configuration
3      return DC-From(Γ, c_0)

4  Recursive Function DC-From(Γ, c)
       Input  : Γ is a CSTND satisfying the conditions of Theorem 10 for some values ε > 0 and
                W ∈ ℤ⁺, c = (kε/K, ψ, h) is a configuration with k ∈ {0, 1, ..., 2K²W} and
                K = 2^|𝒫| · |𝒯|.
       Returns: true if Γ is dynamically consistent from c, false otherwise
5      if Dom(ψ) = 𝒯 then return (true if ψ is feasible for Γ_h^+ else false)   ▷ Base case
6      foreach 𝒯_next ⊆ 𝒯 \ Dom(ψ) not empty do   ▷ Recursive case. Enumerate all next actions (∃)
7          𝒞𝒫_next ← {p ∈ 𝒞𝒫 | 𝒪(p) ∈ 𝒯_next}
8          𝒟𝒫_next ← {p ∈ 𝒟𝒫 | 𝒪(p) ∈ 𝒯_next}
9          foreach k_next ∈ {k + 1, ..., 2K²W} do                  ▷ Enumerate all discrete times (∃)
10             t_next ← k_next ε
11             ψ' ← ψ[t_next/𝒯_next]
12             foreach d: 𝒟𝒫_next → {⊤, ⊥} do                      ▷ Enumerate all decisions (∃)
13                 AllObsDC ← true
14                 foreach o: 𝒞𝒫_next → {⊤, ⊥} do                  ▷ Enumerate all observations (∀)
15                     h' ← h ∪ o ∪ d
16                     c' ← (t_next, ψ', h')
17                     if not DC-From(Γ, c') then AllObsDC ← false   ▷ Recursion
18                 if AllObsDC then return true
19     return false
```

the current time, $\psi\colon \mathcal{T}' \to [0,t)$ (with $\mathcal{T}' \subseteq \mathcal{T}$) is a partial schedule, and $h\colon \mathcal{P}' \to \{\top,\bot\}$ (with $\mathcal{P}' = \{p \in \mathcal{P} \mid \mathcal{O}(p) \in \mathrm{Dom}(\psi)\}$) is the partial (combined) scenario known at time t.

By Lemma 10, we may assume that each t satisfies $t = k{\cdot}\epsilon/K$ for some $k \in \{0,1,\dots,2K^2W\}$ and $[\psi]_X = k_X \cdot \epsilon/K$, where $k_X \in \{0,\dots,k-1\}$ for every $X \in \mathcal{T}$.

The base case of our recursive procedure corresponds to a final configuration of the game, which occurs when all the time-points have been executed (i.e., $\mathcal{T}' = \mathcal{T}$) and, hence, all propositional variables have been assigned. The agent wins if and only if the total schedule ψ is feasible for the projected network Γ_h^+ over the total combined scenario h.

In the recursive step, the algorithm enumerates all possible moves of the agent and all possible counter-moves of the environment. A move of the agent consists of:

1. the set $\mathcal{T}_{next} \subseteq \mathcal{T} \setminus \mathrm{Dom}(\psi)$ of time-points to execute next,
2. the time $t' > t$ at which these time-points are executed, and
3. the decisions to take at time t' (i.e., an assignment $d\colon \mathcal{DP}_{next} \to \{\top,\bot\}$ of the variables in $\mathcal{DP}_{next} = \{p \in \mathcal{DP} \mid \mathcal{O}(p) \in \mathcal{T}_{next}\}$, which must be decided at time t'.

Thanks to Lemma 10, we can assume $t' = k' \cdot \epsilon/K$ for $k' \in \{k+1,\dots,2K^2W\}$. This ensures that there are a finite number of possible moves, and that each move can be described with a polynomial number of bits.

The counter-move of the environment sets the values for the conditions to be revealed to the agent at time t' (i.e., an assignment $o\colon \mathcal{CP}_{next} \to \{\top,\bot\}$ of the propositional variables in $\mathcal{CP}_{next} = \{p \in \mathcal{CP} \mid \mathcal{O}(p) \in \mathcal{T}_{next}\}$ observed at time t'.

Checking the dynamic consistency of a network amounts to determining the winner of the game from the initial configuration $c_0 = (0,\emptyset,\emptyset)$, where the current time is 0 and no

time-points have yet been executed.

5 An Algorithm for CSTNDs having no Condition

In this section we propose an algorithm for solving the DC problem for CSTNDs having no contingent propositional variable, i.e., when $\mathcal{CP}$ is empty. Hereinafter we refer to this subclass of CSTND as the class of Simple Temporal Network with Decisions (STND).

▶ **Definition 11** (STND). A *Simple Temporal Network with Decisions* (STND) is a CSTND $\Gamma = \langle \mathcal{T}, \mathcal{P}, \emptyset, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ having $\mathcal{CP} = \emptyset$. An STND Γ is *consistent* if and only if there exists a decision scenario $ds \in \Sigma_{\mathcal{DP}}$ for which the projection STN Γ_{ds} is consistent.

Since contingent propositional variables are not present, it is possible to check the consistency of the network in a static way. The approach we are about to discuss synthesizes such an assignment offline, i.e., before the execution of the STND starts.

Basically, the proposed algorithm (Algorithm 2) checks whether an STND Γ is consistent by looking for one $ds \in \Sigma_{\mathcal{DP}}$ for which the projection STN Γ_{ds} is consistent. This search is performed by maintaining a support CNF ϕ, which, roughly speaking, represents the complementary of the space of all decision scenarios for which Γ is already known to be inconsistent; to satisfy ϕ means to find a decision scenario outside that space.

Therefore, Algorithm 2 (`STND-CC`) takes as input an STND Γ and it employs an approach working in rounds. Throughout the rounds, `STND-CC` maintains a formula ϕ in CNF representing all decision scenarios for which Γ is already known to be inconsistent. `STND-CC` keeps trying to guess a decision scenario $ds \in \Sigma_{\mathcal{DP}}$ until either $ds \not\models \phi$, or $ds \models \phi$ and Γ_{ds} is consistent—in such case it returns YES. If Γ_{ds} is inconsistent for all possible decision scenario, `STND-CC` returns NO.

Initially, ϕ contains no clauses and ds is a random decision scenario; thus, $ds \models \phi$ holds trivially. If Γ_{ds} is consistent, then `STND-CC` outputs `YES` and halts. Otherwise, ds contains at least a "bad decision" making Γ_{ds} inconsistent, i.e., Γ_{ds} contains a negative cycle. Let ρ be such a negative cycle and let ψ be the conjunction of the labels associated to the values making the negative cycle (see Algorithm 3). Then, `STND-CC` (i) augments ϕ adding $\neg(\psi)$ (which is a disjunction of literals) as a new clause (line 9), (ii) determines a new ds that satisfies ϕ using an external SAT solver, and (iii) proceeds to the next round if a suitable ds has been found, outputs `NO` and halts otherwise.

`STND-CC` uses a SAT solver as MiniSat [18] to find a decision scenario ds that satisfies (the augmented) ϕ. We underline that a decision scenario cannot be found when ϕ is unsatisfiable, i.e., when Γ is inconsistent. In this case, the algorithm can stop saying `NO`.

It is not difficult to see that `STND-CC` is sound and complete. As regards time complexity, the following facts hold. Checking the consistency of Γ_{ds} (line 5 of `STN-CC`) requires time $O(|\mathcal{T}| \cdot |\mathcal{C}|)$ using Bellman-Ford algorithm [16]. The time for determining a negative cycle ρ of Γ_{ds} (`CYCLE-CUT`) amounts to that of applying De Morgan's law to $\neg(\psi)$, which is a linear time in $|\mathcal{DP}|$. Finally, the number of invocations to the SAT Solver (line 11 of Algorithm 2) is at most that of all possible considered decision scenarios s, i.e., $2^{|\mathcal{DP}|}$; each of such invocations costs $O(2^{|\mathcal{DP}|})$ time. Therefore, the worst-case time complexity of `STND-CC` is $O(2^{2|\mathcal{DP}|} \cdot (|\mathcal{T}| \cdot |\mathcal{C}| + |\mathcal{DP}|))$.

▶ **Theorem 12.** *The problem of checking whether a given STND $\Gamma = \langle \mathcal{T}, \mathcal{P}, \emptyset, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ is consistent can be solved in singly exponential (w.r.t. $|\mathcal{DP}|$) deterministic time. Moreover, when Γ is consistent, a positive certificate (ds, ρ) (where ds is a decision scenario over $\mathcal{DP}$ and ρ is a feasible schedule of Γ_{ds}) is computable within the same time bound.*

Algorithm 2: Consistency Checking Algorithm with Decisions

```
Procedure STND-CC(Γ)
    input  : An STND Γ = ⟨T, P, ∅, DP, C, OT, O⟩.
    output : A feasible schedule of Γ or NO.
1   φ ← an empty set of clauses on DP;
2   ds ← any assignment on DP;                    ▷ Initialize the assignment ds arbitrarily
3   while TRUE do
4       Γ_ds ← the projection of Γ over ds;
5       ρ ← STN-CC(Γ_ds);                         ▷ Check the consistency of Γ_ds
6       if ρ is a feasible schedule of Γ_ds then
7           return (YES, ρ);
8       if ρ is a negative cycle of Γ_ds then
9           ψ ← CYCLE-CUT(Γ, ρ);                  ▷ Derive a clause ψ expressing cut of ρ in Γ
10          φ ← φ ∪ {ψ};                          ▷ Add the clause ψ to the CNF φ
11          ds ← SAT-SOLVE(φ);                    ▷ Invoke a SAT-Solver on input φ
12          if ds ⊭ φ then
13              return NO;
```

Algorithm 3: Cutting a Cycle with Decisions

```
Procedure CYCLE-CUT(Γ, ρ)
    input  : An STND Γ = ⟨T, P, ∅, DP, C, OT, O⟩ and one of its cycles ρ.
    output : A clause ψ on DP expressing the cut of ρ in Γ.
1   ψ ← ⊤;
2   foreach constraint C of ρ do
3       ℓ_C ← the label of C in Γ;                ▷ ℓ_C is a conjunction of literals
4       ψ ← ψ ∧ ℓ_C;                              ▷ ψ is also a conjunction of literals
5   ψ ← DeMorgan(¬ψ);                             ▷ apply De Morgan's law to ¬ψ
6   return ψ;
```

To complete the result of Theorem 12, we prove that STND consistency checking problem is NP-complete. For lack of space, we sketch the proof of a polynomial reduction from 3-SAT to the checking problem. Let $\psi(x_1, \ldots, x_n) = \bigwedge_{j=1}^{m} C_j$ be a 3-CNF formula with n boolean variables $\{x_i\}_{i=1}^{n}$ and with m clauses $\{C_j\}_{j=1}^{m}$. An STND instance can be represented by a triplet $\langle \mathcal{T}, \mathcal{C}, \mathcal{DP} \rangle$, i.e., dropping out $\mathcal{OT}, \mathcal{O}$ and $\mathcal{P}$ $(= \mathcal{DP})$; this compact form already allows a correct consistency checking. Consider STND $\Gamma_\psi = (\mathcal{T} = \{Z\}, \mathcal{C} = \{\mathcal{C}_j\}_{j=1}^{m}, \mathcal{DP} = \{x_i\}_{i=1}^{n})$ having only one time-point Z and m negative self-loop constraints, i.e., $\mathcal{C}_j = \big(Z - Z \leq -1, \neg(C_j)\big)$ where $\neg(C_j)$ can be turned into a label by De Morgan's law. It is not difficult to verify that ψ is SAT if and only if Γ is consistent.

5.1 Hyper Temporal Networks with Decisions

In this subsection we consider the *Hyper Temporal Network with Decisions* (HyTND) model and we prove that Algorithm 2 and 3 can be easily extend to it. *Hyper Temporal Networks* (HyTNs) are a strict generalization of STNs, introduced to partially overcome the limitation of allowing only conjunctions of constraints [12]. Compared to STN distance graphs, HyTNs allow for a greater flexibility in the definition of the temporal constraints meanwhile offering a pseudo-polynomial tractability in the consistency checking of the instances. In turn, HyTNDs extend the HyTN model by labeling each hyper-constraint with a conjunction of literals drawn from the set $\mathcal{DP}$ of controllable propositional variables; then, the consistency problem of HyTNDs asks for the existence of a decision scenario for which the corresponding projection network is consistent.

In order to formally define the model, let us firstly recall (multi-head) hypergraphs.

▶ **Definition 13.** (Hypergraphs) A (multi-head, weighted) *hypergraph* $\mathcal{H}$ is a pair $(\mathcal{T}, \mathcal{HC})$, where $\mathcal{T}$ is a set of nodes and $\mathcal{HC}$ is a set of *hyper-edges*. Each hyper-edge $A = (t_A, H_A, w_A) \in \mathcal{HC}$ has a distinguished node t_A, called the *tail* of A, and a non-empty set $H_A \subseteq \mathcal{T} \setminus \{t_A\}$ comprising the *heads* of A; to each head $v \in H_A$, it is associated a *weight* $w_A(v) \in \mathbb{Z}$. Let W be the maximum absolute weight in $\mathcal{H}$ and $|A| = |H_A \cup \{t_A\}|$. The *size* of $\mathcal{H}$ is $m_{\mathcal{H}} = \sum_{A \in \mathcal{HC}} |A|$, and it is a measure for the encoding length of $\mathcal{H}$. If $|A| = 2$, then $A = (u, v, w)$ is a standard edge; in this way hypergraphs generalize graphs.

We are now in the position to define HyTNDs and related concepts.

▶ **Definition 14.** (HyTND) A *Hyper Temporal Network with Decisions* (HyTND) is a triplet $\Gamma = (\mathcal{T}, \mathcal{HC}, \mathcal{DP})$ where $\mathcal{H} = (\mathcal{T}, \mathcal{HC})$ is a hypergraph and $\mathcal{DP}$ is a set of controllable propositional variables. Nodes $T \in \mathcal{T}$ represent temporal variables (time-points), and each hyper-edge $A = (t_A, H_A, w_A, \ell_A) \in \mathcal{HC}$, where $w_A : H_A \to \mathbb{Z}$ and $\ell_A \in \mathcal{DP}^*$, represents a temporal distance constraint between the tail and the heads (*labeled hyper-constraint* (LHC))

In general, given any $ds \in \Sigma_{\mathcal{DP}}$ and $\psi : \mathcal{T} \to \mathbb{R}$, we say that A is satisfied by (ds, ψ) if and only if the following implication holds: $ds \models \ell_A \Rightarrow \psi(t_A) \geq \min_{v \in H_A} \{\psi(v) - w_A(v)\}$. Note that, when $\mathcal{DP} = \emptyset$, $(\mathcal{T}, \mathcal{HC})$ becomes an *Hyper Temporal Network* (HyTN), and each hyper-edge $A \in \mathcal{HC}$ represents an (unlabeled) temporal distance which is satisfied by any given $\psi : \mathcal{T} \to \mathbb{R}$ if and only if $\psi(t_A) \geq \min_{v \in H_A} \{\psi(v) - w_A(v)\}$. We recall that the HyTN-Consistency problem asks, given any HyTN $\mathcal{H} = (\mathcal{T}, \mathcal{HC})$, to decide whether there exists a schedule $\psi : \mathcal{T} \to \mathbb{R}$ that satisfies every hyper-constraint $A \in \mathcal{HC}$; if so, $\mathcal{H}$ is said *consistent*. HyTND-Consistency is also a static notion of consistency, i.e., all decisions can be taken offline; so, for ease of notation, it is fine to omit decision time-points in Definition 14.

▶ **Definition 15.** (HyTND Projection, HyTND-Consistency) The *projection* of a HyTND $\Gamma = (\mathcal{T}, \mathcal{HC}, \mathcal{DP})$ over a decision scenario $ds \in \Sigma_{\mathcal{DP}}$ is the HyTN $\Gamma_{ds} = (\mathcal{T}, \mathcal{HC}_{ds})$, where:

$$\mathcal{HC}_{ds} = \Big\{ (t_A, H_A, w_A) \mid \exists \ell_A \in \mathcal{DP}^* \text{ s.t. } (t_A, H_A, w_A, \ell_A) \in \mathcal{HC} \text{ and } ds \models \ell_A \Big\}.$$

The HyTND-Consistency problem asks, given any HyTND $\Gamma = (\mathcal{T}, \mathcal{HC}, \mathcal{DP})$, to decide whether there exists a decision scenario $ds \in \Sigma_{\mathcal{DP}}$ such that the projection Γ_{ds} is consistent; if so, Γ is said *consistent* as well.

As a negative cycle is a negative certificate for consistency check in STN, the *generalized negative cycle* is a negative certificate for HyTN and HyTND [12].

▶ **Definition 16** (Generalized (negative) cycle)**.** Given a HyTN $\mathcal{H} = (\mathcal{T}, \mathcal{HC})$, a *cycle* is a pair $(S, \mathcal{C})$ with $S \subseteq \mathcal{T}$ and $\mathcal{C} \subseteq \mathcal{HC}$ such that:

1. $S = \bigcup_{A \in \mathcal{C}} \big(H_A \cup \{t_A\}\big)$ and $S \neq \emptyset$;
2. $\forall v \in S$ there exists an unique $A \in \mathcal{C}$ such that $t_A = v$.

Moreover, we let $a(v)$ denote the unique edge $A \in \mathcal{C}$ with $t_A = v$, as required in above item 2. Every infinite path in a cycle $(S, \mathcal{C})$ contains, at least, one *finite cyclic sequence* $v_i, v_{i+1}, \ldots, v_{i+p}$, where $v_{i+p} = v_i$ is the only repeated node in the sequence. A cycle $(S, \mathcal{C})$ is *negative* if and only if $\sum_{t=1}^{p-1} w_{a(v_t)}(v_{t+1}) < 0$, for *any* finite cyclic sequence $v_1, v_2, \ldots, v_p$.

▶ **Example 17.** An example of a cycle $(S, \mathcal{C})$ is shown in Fig. 3; where $S = \{v_0, \ldots, v_6\}$ and $\mathcal{C} = \{A_0, \ldots, A_6\}$, provided $t_{A_i} = v_i$ for every $i \in \{0, \ldots, 6\}$; and $H_{A_0} = \{v_1, v_2, v_3\}$, $H_{A_1} = \{v_4, v_5\}$, $H_{A_2} = \{v_5, v_6\}$, $H_{A_3} = \{v_0, v_6\}$, $H_{A_4} = \{v_0, v_5\}$, $H_{A_5} = \{v_2, v_6\}$, $H_{A_6} = \{v_3, v_5\}$. Moreover, a finite cyclic sequence, $(v_0, v_2, v_5, v_6, v_3, v_0)$, is highlighted with thickened edges.

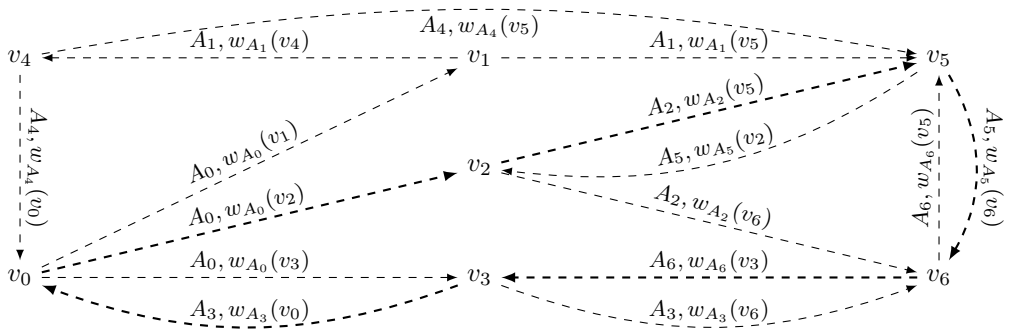

Figure 3 A (generalized) cycle $(S, \mathcal{C})$, where $S = \{v_0, \ldots, v_6\}$ and $\mathcal{C} = \{A_0, \ldots, A_6\}$.

As shown in [12], checking HyTN-Consistency can be done in pseudo-polynomial time.

▶ **Theorem 18** ([12]). *Let $\mathcal{H} = (\mathcal{T}, \mathcal{HC})$ be a HyTN. The following propositions hold:*

1. *There exists an $O((|\mathcal{T}| + |\mathcal{HC}|)m_{\mathcal{H}}W)$ pseudo-polynomial time algorithm deciding* HyTN-Consistency *for $\mathcal{H}$;*
2. *There exists an $O((|\mathcal{T}| + |\mathcal{HC}|)m_{\mathcal{H}}W)$ pseudo-polynomial time algorithm such that, given as input any consistent HyTN $\mathcal{H}$, it returns a feasible schedule $s : \mathcal{T}_{\mathcal{H}} \to \mathbb{Z}$ of $\mathcal{H}$;*
3. *There exists an $O((|\mathcal{T}| + |\mathcal{HC}|)m_{\mathcal{H}}W)$ pseudo-polynomial time algorithm such that, given as input any inconsistent HyTN $\mathcal{H}$, it returns a negative cycle $(S, \mathcal{C})$ of $\mathcal{H}$.*

To solve HyTND-Consistency, one may as well apply Algorithm 2 and Algorithm 3, subject to the following simple modifications:

1. at line 5 of Algorithm 2, replace `STN-CC` by the HyTN-Consistency checking algorithm mentioned in Theorem 18 (see [12]);
2. at line 8 of Algorithm 2, replace "*cycle*" with "*generalized cycle*" and observe that checking whether a generalized cycle $(S, \mathcal{C})$ is negative can be done in polynomial time (see e.g., Lemma 3 in [12] for an algorithm);
3. at line 2 of Algorithm 3, replace "*constraint*" with "*hyper-constraint*" and notice that, since by Definition 14 each $A \in \mathcal{HC}$ has a unique label $\ell_A \in \mathcal{DP}^*$, it is still possible to apply De Morgan's law at line 5 of Algorithm 3 for obtaining a clause.

Considering such modification, it is not difficult to verify that the checking algorithm remains correct. Concerning time complexity, the only noticeable overhead is now due to Theorem 18 results. Considering such complexity results, the new worst-case time complexity for the checking algorithm is $O(2^{2|\mathcal{DP}|} \cdot ((|\mathcal{T}| + |\mathcal{HC}|)m_{\mathcal{H}}W + |\mathcal{DP}|))$.

▶ **Theorem 19.** *The problem of checking whether or not a given HyTND $\Gamma = (\mathcal{T}, \mathcal{HC}, \mathcal{DP})$ is consistent can be solved in (pseudo) singly exponential (w.r.t. $|\mathcal{DP}|$) deterministic time. Moreover, when Γ is consistent, a positive certificate (ds, ρ) (where ds is a decision scenario over $\mathcal{DP}$ and ρ is a feasible schedule of Γ_{ds}) is computable within the same time bound.*

As STNDs are special cases of HyTNDs, checking HyTND-Consistency is NP-complete.

6 An Algorithm for CSTNDs having Offline Decisions

In this section we consider a special case of CSTNDs where decisions are made before the execution of the network starts (offline decisions). This allows us to apply the techniques

developed in the previous Section 5. In this special case, dynamic consistency of CSTNDs is equivalent to the existence of a decision scenario such that the (partial) projection of the network over this scenario (now, a traditional CSTN) is in turn dynamically consistent.

▶ **Definition 20.** [Offline Decision DC] A CSTND $\Gamma = \langle \mathcal{T}, \mathcal{P}, \mathcal{CP}, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ has the property of *Offline Decision Dynamic Consistency* (CSTND-OD-DC) when there exists some decision scenario $ds \in \Sigma_{\mathcal{DP}}$ such that the projection CSTN Γ_{ds} is dynamically consistent.

We will also also say that a CSTND is CSTND-OD-DC if it has CSTND-OD-DC property.

To check if a CSTND instance is CSTND-OD-DC, it is possible to re-use and adapt the techniques presented in Section 5. There, the problem is reduced to check the consistency of a STN instance (projected network), here the projected network is an CSTN. Therefore, it is necessary to consider a CSTN dynamic consistency checking algorithm as the constraint propagation algorithm for CSTNs proposed in [22] for testing the CSTND-OD-DC property. An alternative way to check the CSTN dynamic consistency is to reduce an CSTN instance to an appropriate HyTN one and check the dynamic consistency of the last one [13].

An improvement of such approach, is to reduce the problem of CSTND-OD-DC checking to the problem of HyTND-Consistency consistency check. We now present such reduction.

Firstly, we argue that in this prospect any CSTND can be viewed as a succinct representation which can be expanded into an exponentially sized HyTND. The following definition introduce the concept of *expansion* of a CSTND.

▶ **Definition 21** (Expansion $\langle \mathcal{T}_\Gamma^{\mathsf{Ex}}, \Lambda_\Gamma^{\mathsf{Ex}} \rangle$)**.** For any $\ell \in \mathcal{P}^*$, let us denote by $\ell_{\mathcal{DP}} \in \mathcal{DP}^*$ ($\ell_{\mathcal{CP}} \in \mathcal{CP}^*$) the label comprising all and only those literals of ℓ whose propositional variable lies in $\mathcal{DP}$ ($\mathcal{CP}$, respectively): $\ell = \ell_{\mathcal{DP}} \wedge \ell_{\mathcal{CP}}$ where $\ell_{\mathcal{DP}} \in \mathcal{DP}^*$ and $\ell_{\mathcal{CP}} \in \mathcal{CP}^*$.

For any $s \in \Sigma_{\mathcal{CP}}$, let us consider the *(partial) projection* $\mathcal{C}_s^+$ of $\mathcal{C}$ over s defined as $\mathcal{C}_s^+ = \Big\{ \big(X, Y, \delta, \ell_{\mathcal{DP}}\big) \mid (Y - X \leq \delta, \ell) \in \mathcal{C} \text{ and } s \models \ell_{\mathcal{CP}} \Big\}$.

Next, let us consider the family of distinct and disjoint STNDs $(\mathcal{T}_s, \mathcal{C}_s, \mathcal{DP})$, where $\mathcal{T}_s = \{v_s \mid v \in \mathcal{T}, s \in \Sigma_{\mathcal{CP}}\}$, $\mathcal{C}_s = \Big\{ \big(X_s, Y_s, \delta, \ell_{\mathcal{DP}}\big) \;\Big|\; \big(X, Y, \delta, \ell_{\mathcal{DP}}\big) \in \mathcal{C}_s^+ \Big\}$, and $v_s = (v, s)$ for every $v \in \mathcal{T}, s \in \Sigma_{\mathcal{CP}}$. For each condition scenario $s \in \Sigma_{\mathcal{CP}}$ there is one such STND.

Now, the *expansion STND* of the CSTND Γ is defined as

$$(\mathcal{T}_\Gamma^{\mathrm{Ex}}, \Lambda_\Gamma^{\mathrm{Ex}}, \mathcal{DP}), \text{ where } \mathcal{T}_\Gamma^{\mathrm{Ex}} = \bigcup_{s \in \Sigma_{\mathcal{CP}}} \mathcal{T}_s \text{ and } \Lambda_\Gamma^{\mathrm{Ex}} = \bigcup_{s \in \Sigma_{\mathcal{CP}}} \mathcal{C}_s.$$

Note that $\mathcal{T}_{s_1} \cap \mathcal{T}_{s_2} = \emptyset$ whenever $s_1 \neq s_2$ and that $(\mathcal{T}_\Gamma^{\mathrm{Ex}}, \Lambda_\Gamma^{\mathrm{Ex}}, \mathcal{DP})$ is an STND with $|\mathcal{T}_\Gamma^{\mathrm{Ex}}| \leq |\Sigma_{\mathcal{CP}}| \cdot |\mathcal{T}|$ time-points and size at most $|\Lambda_\Gamma^{\mathrm{Ex}}| \leq |\Sigma_{\mathcal{CP}}| \cdot |\mathcal{C}|$.

We show next that the expansion of a CSTND can be enriched with some (extra) multi-head hyper-edges in order to model dynamic consistency, by means of a particular HyTND $\mathcal{H}_\epsilon^\Gamma$ for some small $\epsilon \in \mathbb{R}_{>0}$. As it was in [13], the actual value of ϵ will turn out to be singly exponentially small in the number of contingent propositional variables (i.e., $\epsilon = \frac{1}{|\Sigma_{\mathcal{CP}}| \cdot |\mathcal{T}|}$).

▶ **Definition 22** (HyTND $\mathcal{H}_\epsilon^\Gamma$)**.** For any two condition scenarios $s_1, s_2 \in \Sigma_{\mathcal{CP}}$, let us denote by $\Delta(s_1; s_2) = \big\{(\mathcal{O}_p?)_{s_1} \in \mathcal{T}_{s_1} \mid \mathcal{O}_p? \in \mathcal{OT}, p \in \mathcal{CP}, s_1(p) \neq s_2(p)\big\}$ the set of all nodes $(\mathcal{O}_p?)_{s_1} \in \mathcal{T}_{s_1}$ such that $\mathcal{O}_p? \in \mathcal{OT}$ is an observation time-point of Γ that is executed in s_1 and, considered in pair with respect to s_2, the value of the variable p differs, i.e., $s_1(p) \neq s_2(p)$.

Given any $\epsilon \in \mathbb{R}_{>0}$, HyTND $\mathcal{H}_\epsilon^\Gamma$ is defined as follows:

- For every two condition scenarios $s_1, s_2 \in \Sigma_{\mathcal{CP}}$ and for every time-point $u \in \mathcal{T}$, define a hyper-edge $\alpha_\epsilon(s_1; s_2; u) = \big(t_\alpha, H_\alpha, w_\alpha, \boxdot\big)$, $\forall\, s_1, s_2 \in \Sigma_{\mathcal{CP}}$ and $u \in \mathcal{T}$, where:
  - $t_\alpha = u_{s_1}$ is the tail of the (multi-head) hyper-edge $\alpha_\epsilon(s_1; s_2; u)$;

- $H_\alpha = \{u_{s_2}\} \cup \Delta(s_1; s_2)$ is the set of the heads of $\alpha_\epsilon(s_1; s_2; u)$;
- $w_\alpha(u_{s_2}) = 0$, and $w_\alpha(v) = -\epsilon$ for each $v \in \Delta(s_1; s_2)$.

- Consider the expansion STND $(\mathcal{T}_\Gamma^{\text{Ex}}, \Lambda_\Gamma^{\text{Ex}}, \mathcal{DP})$ of Γ, the HyTND $\mathcal{H}_\epsilon^\Gamma$ is the tuple $\left(\mathcal{T}_\Gamma^{\text{Ex}}, \mathcal{HC}_\epsilon, \mathcal{DP}\right)$, where $\mathcal{HC}_\epsilon = \Lambda_\Gamma^{\text{Ex}} \cup \bigcup_{\substack{s_1, s_2 \in \Sigma_{\mathcal{CP}} \\ u \in \mathcal{T}}} \alpha_\epsilon(s_1; s_2; u)$.

  Notice that each $\alpha_\epsilon(s_1; s_2; u)$ has size $|\alpha_\epsilon(s_1; s_2; u)| = 1 + |\Delta(s_1; s_2)| \leq 1 + |\mathcal{CP}|$.
  Now, based on the results given in [13], the following result can be shown.

▶ **Theorem 23.** *Let $\Gamma = \langle \mathcal{T}, \mathcal{P}, \mathcal{CP}, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ be a CSTND and let $\hat{\epsilon} = \frac{1}{|\Sigma_{\mathcal{CP}}| \cdot |\mathcal{T}|}$. It holds that Γ is* CSTND-OD-DC *if and only if the HyTND $\mathcal{H}_{\hat{\epsilon}}^\Gamma$ is consistent.*

We give the proof of Theorem 23 in the appendix. Given a CSTND Γ, CSTND-OD-DC can be checked by firstly constructing the HyTND $\mathcal{H}_{\hat{\epsilon}}^\Gamma$ and then by relying on Theorem 19 for checking its consistency. Notice that, even though the size of $\mathcal{H}_{\hat{\epsilon}}^\Gamma$ is singly exponential in $|\mathcal{CP}|$ and also a possible negative generalized cycle can be of an exponential size, the corresponding clause ψ that is eventually returned by Algorithm 3 has size at most $|\mathcal{DP}|$. The obtained results of this section are summarized in the following theorem.

▶ **Theorem 24.** *Deciding* CSTND-OD-DC *on a given CSTND can be done in (pseudo) singly exponential (w.r.t. $|\mathcal{P}|$) deterministic time. When the input CSTND is* CSTND-OD-DC*, a viable and dynamic execution strategy is computable within the same time bound.*

7 Related Work

There are many proposals in the literature for ways of extending the expressiveness of the STN model. Below, we summarize the main results about CSTNs and related models.

Tsamardinos et al. [25] defined the Conditional Simple Temporal Problem (CTP) as that of determining whether a given CSTN admits a viable and dynamic execution strategy. (The CSTN acronym was introduced later.) In their work, propositional labels are associated only with time-points, not constraints. They also informally specified some reasonableness properties that any CSTN ought to satisfy. Although they showed how to solve the CTP by encoding it as a meta-level Disjunctive Temporal Problem (DTP) and feeding it to an off-the-shelf solver, that approach is not practical because the CTP-to-DTP encoding has exponential size and, on top of that, the DTP solver runs in exponential time. To our knowledge, this approach has never been implemented or empirically evaluated.

Later, Hunsberger et al. [21, 22] defined CSTNs (separate from the CTP) and formalized the well-definedness properties for CSTNs. In their work, both time-points (nodes) and constraints (edges) of a CSTN can have propositional labels that specify the scenarios in which they are applicable. (Allowing constraints to be labeled was inspired by the work of Conrad et al. [14], discussed below.) They showed that the labels must satisfy the well-definedness properties in order to guarantee the existence of a dynamic execution strategy. They also presented a sound-and-complete DC-checking algorithm for solving the CTP, and empirically demonstrated its practical performance.

Conrad et al. [14] considered a variant of CSTNs, proposing Drake, a dynamic executive for temporal plans with choice. In their work, the constraints of a temporal plan are labeled as in CSTNs, but the values of propositions (choices) are decided by the executive during run-time, not by the environment.

Cimatti et al. [6, 7] presented a different approach to solving a variety of temporal problems (CSTNs included) in which a temporal network is first translated into an equivalent

Timed Game Automaton (TGA) and, then, solved by an off-the-shelf TGA solver. Although this approach is interesting because it shows the relationships between TGAs and a variety of temporal networks—including CSTNs—it has not yet been shown to be practical for solving the CTP.

Comin and Rizzi [13] solved the CTP by converting it into a Mean Payoff Game (MPG). They also introduced a variant of dynamic consistency, called ε-DC, where $\varepsilon > 0$ represents the minimum reaction time of the executive in response to observations. They presented (1) a sharp lower-bounding analysis on the critical value of the reaction time where the CSTN changes from being DC to non-DC, (2) a proof that the CTP is coNP-hard, and (3) the first singly-exponential-time algorithm for solving the CTP.

Hunsberger and Posenato [19] showed how their DC-checking algorithm from earlier work [22] can be extended to check the ε-DC property without incurring any performance degradation. They also introduced four benchmarks for testing DC-checking algorithms.

Hunsberger and Posenato [20] presented another optimization of the approach presented by Cimatti et al. in which the CTP is viewed as a two-player game. Its solution is determined by exploring an abstract game tree to find a "winning" strategy, using Monte Carlo Tree Search and Limited Discrepancy Search to guide its search. An empirical evaluation shows that the new algorithm is competitive with the propagation-based algorithm.

Cairo et al. [2] improved the analysis of the ε-DC property. They showed that if $\varepsilon = 0$ (i.e., if the system can react instantaneously), it is necessary to impose a further condition to avoid a form of instantaneous circularity.

Cui and Haslum [15] extend STNU by conditioning temporal constraints on the assignment of controllable discrete variables (decisions) that can be done at any time. In CSTNDs we connect decisions to time-points and thus provide greater expressiveness because we allow a designer to constraint when decisions have to been taken.

Zavatteri [26] defined CSTNUDs (i.e., CSTNUs [9] augmented with decision nodes), using an approach based on Timed Game Automata (TGAs) for both checking the dynamic controllability of CSTNUDs and the synthesizing memoryless execution strategies. Although CSTNUDs are more general than CSTNDs, this paper focused on analyzing the complexity of the CSTND-DC problem and presenting DC-checking algorithms for two special cases of CSTNDs.

8 Conclusions and Future Work

This paper introduced a new kind of temporal network, called a *Conditional Simple Temporal Network with Decisions*, that accommodates both *conditions* that are not under the control of the executing agent, and *decisions* that are under the agent's control. The agent aims to make decisions and schedule time-points so that all relevant constraints are satisfied no matter how the environment assigns values to the conditions in real time. After defining a notion of dynamic consistency for CSTNDs, the paper proved that the CSTND-DC problem is PSPACE-complete. Finally, it introduced some algorithms to deal with two special cases: (1) CSTNDs that contain decisions, but not conditions; and (2) CSTNDs for which all decisions are made prior to executing the network.

As for future work, among the many possible research directions, we mention here the application of our approach to the design of business process models, where not all of the decisions (represented as gateway variables in business process models) are under the control of the process engine. Another potential topic concerns the identification of other special cases of CSTNDs that might yield corresponding DC-checking algorithms.

A Appendix: Proofs of Lemma 10 and Theorem 23.

Lemma 10. Let $\sigma = (\sigma^{\mathrm{t}}, \sigma^{\mathrm{d}})$ be any dynamic and viable execution strategy. We know from [4] that the statement of the lemma is valid for CSTNs. Therefore, if we do not consider all the controllable propositional variables, and transform the corresponding observation time-points into standard time-points, we can apply the result given in [4] for transforming σ^{t} to a strategy that satisfies the statement of the lemma. In the transformation showed in the proof [4], only the numerical values determined by the strategy for time-points are modified while the relative execution order of them is preserved. Hence, the conditions for σ^{d} to be dynamic are not changed by such transformation. Therefore, the resulting strategy is indeed dynamic and viable. ◀

Theorem 23. By Definition 20, the CSTND Γ is CSTND-OD-DC if and only if there exists some $ds \in \Sigma_{\mathcal{DP}}$ such that the CSTN Γ_{ds} is dynamically consistent. By Theorem 4 and 6 in [13], for any $ds \in \Sigma_{\mathcal{DP}}$, the CSTN Γ_{ds} is dynamically consistent if and only if the HyTN $\mathcal{H}_{\hat{\epsilon}}^{\Gamma_{ds}}$ is consistent provided that $\hat{\epsilon} = \frac{1}{|\Sigma_{\mathcal{CP}}|\cdot|\mathcal{T}|}$. Then, considering the definition of (i) CSTND projection (Definition 4), (ii) HyTN projection (Definition 15), and (iii) that of $\mathcal{H}_{\epsilon}^{\Gamma}$ (Definition 22), it holds that $\mathcal{H}_{\hat{\epsilon}}^{\Gamma_{ds}}$ is HyTN $(\mathcal{H}_{\hat{\epsilon}}^{\Gamma})_{ds}$.

Finally, by Definition 15, there exists some $ds \in \Sigma_{\mathcal{DP}}$ such that the projection HyTN $(\mathcal{H}_{\hat{\epsilon}}^{\Gamma})_{ds}$ is consistent if and only if the HyTND $\mathcal{H}_{\hat{\epsilon}}^{\Gamma}$ is consistent. At this point, by composing all of these logical equivalences, the thesis follows. ◀

References

1 Claudio Bettini, Xiaoyang Sean Wang, and Sushil Jajodia. Temporal reasoning in workflow systems. *Distributed and Parallel Databases*, 11(3):269–306, 2002. doi:10.1023/A:1014048800604.

2 Massimo Cairo, Carlo Comin, and Romeo Rizzi. Instantaneous reaction-time in dynamic-consistency checking of conditional simple temporal networks. In *23rd International Symposium on Temporal Representation and Reasoning (TIME 2016)*, pages 80–89, 2016. doi:10.1109/TIME.2016.16.

3 Massimo Cairo, Luke Hunsberger, Roberto Posenato, and Romeo Rizzi. A streamlined model of conditional simple temporal networks. In *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*, volume 90 of *LIPIcs*, pages 10:1–10:19, 2017. doi:10.4230/LIPIcs.TIME.2017.10.

4 Massimo Cairo and Romeo Rizzi. Dynamic controllability of conditional simple temporal networks is PSPACE-complete. In *23rd International Symposium on Temporal Representation and Reasoning (TIME 2016)*, pages 90–99, 2016. doi:10.1109/TIME.2016.17.

5 Susan J. Chinn and Gregory R. Madey. Temporal representation and reasoning for workflow in engineering design change review. *IEEE Transactions on Engineering Management*, 47(4):485–492, 2000. doi:10.1109/17.895343.

6 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.

7 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.

8 Carlo Combi, Mauro Gambini, Sara Migliorini, and Roberto Posenato. Representing business processes through a temporal data-centric workflow modeling language: An application to the management of clinical pathways. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(9):1182–1203, 2014. doi:10.1109/TSMC.2014.2300055.

9 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 (ICAART 2013)*, volume 2, pages 144–156, 2013. doi:10.5220/0004256101440156.

10 Carlo Combi and Roberto Posenato. Controllability in temporal conceptual workflow schemata. In *Business Process Management (BPM 2009)*, volume 5701 of *LNCS*, pages 64–79, 2009. doi:10.1007/978-3-642-03848-8_6.

11 Carlo Combi and Roberto Posenato. Towards temporal controllabilities for workflow schemata. In *17th International Symposium on Temporal Representation and Reasoning (TIME 2010)*, pages 129–136, 2010. doi:10.1109/TIME.2010.17.

12 Carlo Comin, Roberto Posenato, and Romeo Rizzi. Hyper temporal networks - A tractable generalization of simple temporal networks and its relation to mean payoff games. *Constraints*, 22(2):152–190, 2017. doi:10.1007/s10601-016-9243-0.

13 Carlo Comin and Romeo Rizzi. Dynamic consistency of conditional simple temporal networks via mean payoff games: A singly-exponential time dc-checking. In *22nd International Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 19–28, 2015. doi:10.1109/TIME.2015.18.

14 Patrick R. Conrad and Brian C. Williams. Drake: An efficient executive for temporal plans with choice. *Journal of Artificial Intelligence Research*, 42(1):607–659, 2011. doi:10.1613/jair.3478.

15 Jing Cui and Patrik Haslum. Dynamic controllability of controllable conditional temporal problems with uncertainty. In *27th International Conference on Automated Planning and Scheduling (ICAPS 2017)*, 2017. URL: https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15738.

16 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.

17 Johann Eder, Wolfgang Gruber, and Euthimios Panagos. Temporal modeling of workflows with conditional execution paths. In *Database and Expert Systems Applications (DEXA 2000)*, volume 1873 of *LNCS*, pages 243–253, 2000. doi:10.1007/3-540-44469-6_23.

18 Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Theory and Applications of Satisfiability Testing: 6th International Conference (SAT 2003)*, pages 502–518, 2004. doi:10.1007/978-3-540-24605-3_37.

19 Luke Hunsberger and Roberto Posenato. Checking the dynamic consistency of conditional simple temporal networks with bounded reaction times. In *26th International Conference on Automated Planning and Scheduling (ICAPS 2016)*, pages 175–183, 2016. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/view/13108.

20 Luke Hunsberger and Roberto Posenato. A new approach to checking the dynamic consistency of conditional simple temporal networks. In *Principles and Practice of Constraint Programming (CP 2016)*, volume 9892 of *LNCS*, pages 268–286, 2016. doi:10.1007/978-3-319-44953-1_18.

21 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, 2012. URL: http://arxiv.org/abs/1212.2005.

22 Luke Hunsberger, Roberto Posenato, and Carlo Combi. A sound-and-complete propagation-based algorithm for checking the dynamic consistency of conditional simple temporal net-

works. In *22st International Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 4–18, 2015. doi:10.1109/TIME.2015.26.

23 Eleanna Kafeza and Kamalakar Karlapalem. Gaining control over time in workflow management applications. In *Database and Expert Systems Applications: 11th International Conference (DEXA 2000)*, volume 1873 of *LNCS*, pages 232–241, 2000. doi:10.1007/3-540-44469-6_22.

24 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 2001)*, pages 494–502, 2001.

25 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.

26 Matteo Zavatteri. Conditional simple temporal networks with uncertainty and decisions. In *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*, volume 90 of *LIPIcs*, pages 23:1–23:17, 2017. doi:10.4230/LIPIcs.TIME.2017.23.