Sound-and-Complete Algorithms for Checking the Dynamic Controllability of Conditional Simple Temporal Networks with Uncertainty

Luke Hunsberger
Department of Computer Science, Vassar College, NY, USA
hunsberger@vassar.edu

Roberto Posenato
Department of Computer Science, University of Verona, Verona, Italy
roberto.posenato@univr.it
https://orcid.org/0000-0003-0944-0419

Abstract

A Conditional Simple Temporal Network with Uncertainty (CSTNU) is a data structure for representing and reasoning about time. CSTNUs incorporate *observation time-points* from Conditional Simple Temporal Networks (CSTNs) and *contingent links* from Simple Temporal Networks with Uncertainty (STNUs). A CSTNU is *dynamically controllable* (DC) if there exists a strategy for executing its time-points that guarantees the satisfaction of all relevant constraints no matter how the uncertainty associated with its observation time-points and contingent links is resolved in real time. This paper presents the first sound-and-complete DC-checking algorithms for CSTNUs that are based on the propagation of labeled constraints and demonstrates their practicality.

2012 ACM Subject Classification Computing methodologies → Temporal reasoning, Theory of computation → Network optimization, Theory of computation → Dynamic graph algorithms, Mathematics of computing → Graph algorithms

Keywords and phrases Temporal Networks, Conditional Simple Temporal Problem with Uncertainty, Dynamic Controllability, Checking Algorithm

Digital Object Identifier 10.4230/LIPIcs.TIME.2018.14

1 Introduction

A Conditional Simple Temporal Network with Uncertainty (CSTNU) is a data structure for representing and reasoning about time in domains where some constraints may apply only in certain scenarios and some events may have uncontrollable, but bounded durations [13]. They were defined to represent important features of, for example, (1) *workflow systems* used to automate medical-treatment processes [16],[7]; and (2) *planning systems* when uncertain durations are present [17]. A CSTNU may include *observation time-points* and *contingent links*. An observation time-point represents a test action whose execution generates a truth value for a corresponding propositional letter. A contingent link represents an action with an uncertain, but bounded duration. Observation time-points and contingent links both involve uncertainty *and* uncontrollability, since the outcomes of tests and contingent durations are not known in advance and are not controlled by the scheduling agent; they are only *observed* during execution.

CSTNUs generalize Conditional Simple Temporal Networks (CSTNs) [23] and Simple Temporal Networks with Uncertainty (STNUs) [21]. The *dynamic controllability* (DC) property for CSTNUs generalizes the corresponding properties for CSTNs and STNUs. In brief, a CSTNU is DC if there exists a strategy for executing its time-points that guarantees

25th International Symposium on Temporal Representation and Reasoning (TIME 2018).
Editors: Natasha Alechina, Kjetil Nørvåg, and Wojciech Penczek; Article No. 14; pp. 14:1–14:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the satisfaction of all relevant constraints no matter how the uncertainty associated with its observation time-points and contingent links is resolved during execution. The *DC-checking problem* for CSTNUs is that of determining whether arbitrary CSTNUs are DC.

Combi et al. [5, 6] presented a variety of sound constraint-propagation rules for CSTNUs, but did not address completeness. Following their approach, but focusing on CSTNs, Hunsberger et al. [14] presented the first practical sound-and-complete DC-checking algorithm for CSTNs. Recently, they presented a faster version of their algorithm, called the π-DC-checking algorithm [12], that will play a role in this paper. In other approaches, Hunsberger and Posenato [11] presented an algorithm that views the DC-checking problem for CSTNs as a two-player game, searching an abstract game tree to find a "winning" strategy, guided by Monte-Carlo Tree Search and Limited Discrepancy Search; and Cimatti et al. [4] reduced the DC-checking problem for CSTNUs (and a broader class of networks) to a controller-synthesis problem for timed game automata, but have not shown whether that approach can be made practical for CSTNUs.

Contribution. This paper presents the first *practical sound-and-complete* DC-checking algorithms for CSTNUs. The first algorithm reduces the DC-checking problem for CSTNUs to the DC-checking problem for CSTNs; the second propagates constraints directly in the input CSTNU. The paper proves that both algorithms are correct and empirically evaluates their performance.

2 Conditional STNs with Uncertainty (CSTNUs)

This section recalls the definition of a well-defined CSTNU, which allows contingent links (as in an STNU) and observation time-points (as in a CSTN). The presentation combines and extends definitions from earlier work [18, 13, 6, 12]. The notion of a *streamlined* temporal network from recent work on CSTNs [2] is then applied to CSTNUs.

▶ **Definition 1** (P-Labels). For a set $\mathcal{P}$ of propositional letters, a *p-label* is a (possibly empty) conjunction of (positive or negative) literals from $\mathcal{P}$. The *empty p-label* is notated $\boxdot$. For any p-label ℓ, and any $p \in \mathcal{P}$, if $\ell \models p$ or $\ell \models \neg p$, we say that p *appears* in ℓ. For p-labels, ℓ_1 and ℓ_2, if $\ell_1 \models \ell_2$, we say that ℓ_1 *entails* ℓ_2. If $\ell_1 \wedge \ell_2$ is satisfiable, we say that ℓ_1 and ℓ_2 are *consistent*. $\mathcal{P}^*$ denotes the set of all *satisfiable* p-labels with literals from $\mathcal{P}$.

▶ **Definition 2** (CSTNU). A *Conditional Simple Temporal Network with Uncertainty* is a tuple, $\langle \mathcal{T}, \mathcal{P}, L, \mathcal{C}, \mathcal{OT}, \mathcal{O}, \mathcal{L} \rangle$, where:

- $\mathcal{T}$ is a finite set of real-valued variables, called *time-points*;
- $\mathcal{P}$ is a finite set of propositional letters;
- $L\colon \mathcal{T} \to \mathcal{P}^*$ assigns p-labels to time-points;
- $\mathcal{C}$ is a 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 a set of *observation* time-points;
- $\mathcal{O}\colon \mathcal{P} \to \mathcal{OT}$ is a bijection from propositional letters to observation time-points;
- $\mathcal{L}$ is a set of *contingent links* each of the form (A, x, y, C), where: $A \in \mathcal{T}, C \in \mathcal{T} \setminus \mathcal{OT}$, $A \not\equiv C$ are called the *activation* and *contingent* time-points, respectively; $L(A) = L(C)$; $0 < x < y < \infty$; and distinct contingent links have distinct contingent time-points.

By convention, for each $p \in \mathcal{P}$, $\mathcal{O}(p)$ (i.e., the observation time-point whose execution determines the truth value for p) may be denoted by $P?$.

Hunsberger *et al.* [14] called a CSTN *well-defined* if the p-labels on its time-points and constraints satisfied certain properties. CSTNs that are not well defined turn out to be useless. Definition 3 extends well-definedness to CSTNUs.

▶ **Definition 3** (Well-defined CSTNU). A CSTNU is *well defined* if:

1. for each $(Y - X \leq \delta, \ell) \in \mathcal{C}$, $\ell \models L(X) \wedge L(Y)$;

2. for each $T \in \mathcal{T}$, and each p appearing in $L(T)$:

  a. $\ell \models L(P?)$; and

  b. $(P? - T \leq -\epsilon, L(T)) \in \mathcal{C}$, for some $\epsilon > 0$;

3. for each $(Y - X \leq \delta, \ell) \in \mathcal{C}$, and each p appearing in ℓ, $\ell \models L(P?)$.

Cairo *et al.* [2] showed that if $\mathcal{S}$ is a well-defined CSTN (i.e., satisfies properties 1–3 above), then there is a CSTN $\mathcal{S}_{\boxdot}$ such that:

1. $\mathcal{S}_{\boxdot}$ does *not* have any labels on its time-points, and

2. $\mathcal{S}_{\boxdot}$ is DC if and only if $\mathcal{S}$ is DC.

$\mathcal{S}_{\boxdot}$ is called a *streamlined* CSTN. We say that $\mathcal{S}_{\boxdot}$ is *DC-equivalent* to $\mathcal{S}$.
This result extends easily to CSTNUs.

▶ **Definition 4** (Streamlined CSTNU). Given a well-defined CSTNU $\mathcal{S}$, its *streamlined* version $\mathcal{S}_{\boxdot}$ is the same as $\mathcal{S}$ except that its time-points have no p-labels.

It is straightforward to show that if $\mathcal{S}$ is a *well-defined* CSTNU, then $\mathcal{S}$ is DC if and only if its streamlined version $\mathcal{S}_{\boxdot}$ is DC. For this reason, this paper restricts attention to streamlined CSTNUs, which simplifies definitions and proofs of the main results with no loss of generality.

Each CSTNU has a corresponding graph whose nodes represent time-points, and whose edges represent various kinds of temporal constraints. To accommodate the propositional labels (p-labels) from CSTN graphs and the alphabetic labels (a-labels) from STNU graphs, each edge in a CSTNU graph may be annotated by both p-labels *and* a-labels. This paper treats a-labels more rigorously than in prior work in anticipation of how they are conjoined and modified during constraint propagation. In particular, an a-label is liberally defined to allow *conjunctions of upper-case letters* that may arise during constraint propagation.

▶ **Definition 5** (A-Letters, A-Labels). If $C_1, \ldots, C_k$ are the contingent time-points for a CSTNU $\mathcal{S}$, then $\mathcal{A} = \{c_1, \ldots, c_k, C_1, \ldots, C_k\}$ is the set of *alphabetic letters (a-letters)* for $\mathcal{S}$; $c_1, \ldots, c_k$ are the *lower-case* (LC) a-letters; and $C_1, \ldots, C_k$ are the *upper-case* (UC) a-letters. An *a-label*, $\aleph$, is a set of a-letters that: (1) is empty, notated as $\diamond$; (2) contains exactly one LC a-letter, notated as c_i; or (3) contains *one or more* UC a-letters, notated as $C_{i_1} \ldots C_{i_m}$. The set of all a-labels with letters from $\mathcal{A}$ is denoted by $\mathcal{A}^*$. The set of *UC a-labels* (that is, a-labels that contain zero or more UC a-letters) is denoted by $\mathcal{A}^*_u$. For any $\aleph, \aleph' \in \mathcal{A}^*_u$, their *conjunction* is given by their union (i.e., $\aleph\aleph' = \aleph \cup \aleph'$).

Edges in a CSTNU graph are annotated by triples, called a *labeled values,* that generalize: (1) the numerical weights that appear on edges in STN graphs; (2) the *lower-case* and *upper-case* a-letters on edges in STNU graphs; and (3) the p-labels on edges in CSTN graphs. Since any pair of time-points, X and Y, may participate in multiple constraints, an edge from X to Y may have multiple labeled values, each of the form, $\langle \delta_i, \aleph_i, \ell_i \rangle$.

▶ **Definition 6** (Labeled values). A *labeled value* is a triple, $\langle \delta, \aleph, \ell \rangle$, where: $\delta \in \mathbb{R}$, $\aleph \in \mathcal{A}^*$, and $\ell \in \mathcal{P}^*$.

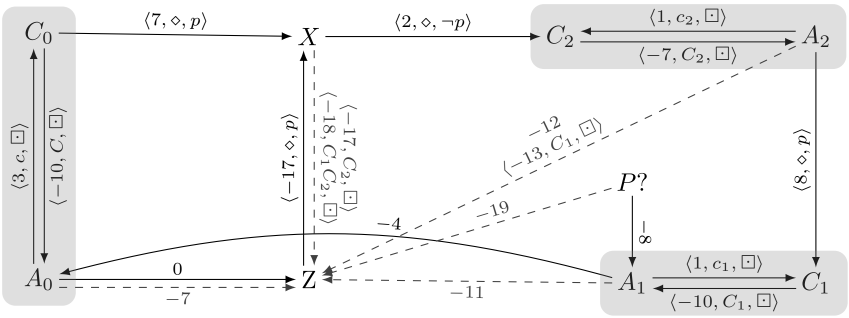

C_0
$\langle 7, \diamond, p\rangle$
X
$\langle 2, \diamond, \neg p\rangle$
C_2
$\langle 1, c_2, \boxdot\rangle$
A_2
$\langle -7, C_2, \boxdot\rangle$
$\langle 3, c, \boxdot\rangle$
$\langle -10, C, \boxdot\rangle$
$\langle -17, \diamond, p\rangle$
$\langle -17, C_2, \boxdot\rangle$
$\langle -18, C_1C_2, \boxdot\rangle$
-12
$\langle -13, C_1, \boxdot\rangle$
$P?$
-19
$\langle 8, \diamond, p\rangle$
-4
$-\infty$
0
-7
Z
-11
A_0
A_1
$\langle 1, c_1, \boxdot\rangle$
C_1
$\langle -10, C_1, \boxdot\rangle$

Figure 1 A CSTNU graph with 3 contingent links and one observation time-point $P?$

In a CSTNU graph, an *ordinary* STN constraint, $Y - X \leq \delta$, is represented by an edge $X\xrightarrow{\langle\delta,\diamond,\boxdot\rangle}Y$; and a conditional constraint, $(Y - X \leq \delta, \ell)$, is represented by an edge $X\xrightarrow{\langle\delta,\diamond,\ell\rangle}Y$. Edges associated with contingent links require further introduction.

In an *STNU* graph, each contingent link (A, x, y, C) gives rise to two edges: a *lower-case* edge from $A\xrightarrow{c:x}C$ that represents the possibility that the duration $C - A$ might take on its minimum value x; and an *upper-case* edge from $C\xrightarrow{C:-y}A$ that represents that $C - A$ might take on its maximum value y. In a CSTNU graph, the lower-case edge is $A\xrightarrow{\langle x,c,\boxdot\rangle}C$ and the upper-case edge is $C\xrightarrow{\langle -y,C,\boxdot\rangle}A$.

Figure 1 shows the graph for a sample CSTNU. Z is a (non-contingent, non-observation) time-point whose value is fixed at 0. Each time-point X is implicitly constrained to occur at or after Z. Labeled values such as $\langle -11, \diamond, \boxdot\rangle$ are abbreviated as simply -11. The dashed edges represent constraints obtained by Algorithm 2, described in section 6; it determines that this CSTNU is not dynamically controllable.

3 Dynamic Controllability for CSTNUs

The truth values of propositions in a CSTNU are not known in advance; they are incrementally revealed as observation time-points are executed. Similarly, the durations of contingent links are only observed as the contingent time-points happen to execute. However, a *dynamic strategy* for executing the time-points in a CSTNU can *react* to observations and contingent executions in real time. A *viable* strategy is one that guarantees that all relevant constraints will be satisfied no matter which truth values and durations are revealed over time. A CSTNU with a dynamic and viable strategy is *dynamically controllable* (DC).

Like much recent work on STNUs and CSTNs [18, 14, 1], this paper defines the DC property for CSTNUs to allow execution strategies to react *instantaneously* to observations, instead of requiring arbitrarily small delays. It generalizes the DC semantics for STNUs [18] and the π-DC semantics for CSTNs [1].

This paper focuses on CSTNUs whose sets of contingent and observation time-points are distinct – with no loss of generality because any contingent observation time-point could be represented by two time-points, a contingent time-point C and an observation time-point $P?$, constrained to occur simultaneously. An instantaneously reactive strategy could wait for C to execute and then execute $P?$ at the same time.

Preliminaries. A *scenario* s specifies a truth value for each proposition, and a *situation* ω specifies a duration for each contingent link. A *drama* is then a scenario-situation pair (s, ω). The projection of a CSTNU onto a drama (s, ω) is the *STN* obtained by restricting attention to the constraints whose labels are true under s and assigning each contingent duration to the value specified by ω.

▶ **Definition 7** (Scenario/Situation/Drama/Projection). A *scenario* is a function, $s\colon \mathcal{P} \to \{\top, \bot\}$, that assigns a truth value to each $p \in \mathcal{P}$. A scenario also determines the truth value, $s(\ell)$, for any p-label $\ell \in \mathcal{P}^*$. The set of all scenarios over $\mathcal{P}$ is denoted by $\mathcal{I}$. If (A_1, x_1, y_1, C_1), $\ldots$, (A_k, x_k, y_k, C_k) are the contingent links for a CSTNU $\mathcal{S}$, then $\Omega = [x_1, y_1] \times \ldots \times [x_k, y_k]$ is called the *space of situations* for $\mathcal{S}$, and any $\omega = (\omega_1, \ldots, \omega_k) \in \Omega$ is called a *situation.* A *drama* is any pair $(s, \omega) \in \mathcal{I} \times \Omega$, where $s \in \mathcal{I}$ is a scenario, and $\omega \in \Omega$ is a situation. Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O}, \mathcal{L} \rangle$ be any CSTNU, and (s, ω) any drama for $\mathcal{S}$, where $\omega = (\omega_1, \ldots, \omega_k)$. The *projection* of $\mathcal{S}$ onto (s, ω) – denoted by $\mathit{Prj}(\mathcal{S}, s, \omega)$ – is the STN, $(\mathcal{T}, \mathcal{C}_s)$, where:[1]

$$\begin{aligned}\mathcal{C}_s = \{(Y - X \le \delta) \mid\ &\text{for some } \ell, (Y - X \le \delta, \ell) \in \mathcal{C} \text{ and } s(\ell) = \top\} \\ \cup \{(C_i - A_i = \omega_i) \mid\ &(A_i, x_i, y_i, C_i) \in \mathcal{L}\}\end{aligned}$$

3.1 Execution strategies

Cairo et al. [1] introduced the π-DC semantics for CSTNs that, unlike prior versions [14, 8], does not permit a kind of circular dependency among simultaneous observations. A π-dynamic strategy must specify, for each scenario, both a *schedule* for the time-points, and an *order of dependency* among the observation time-points. *This section extends the π-DC semantics to cover CSTNUs.*

▶ **Definition 8** (Schedule). A *schedule* for a set $\mathcal{T}$ of time-points is a *complete* mapping, $\psi\colon \mathcal{T} \to \mathbb{R}$. For any $X \in \mathcal{T}$, and any schedule ψ, the execution time for X in ψ is denoted by $[\psi]_X$. The set of schedules for $\mathcal{T}$ is denoted by Ψ.

▶ **Definition 9** (Order of Dependency). Let $\mathcal{OT} = \{P_1?, \ldots, P_k?\}$ be a set of observation time-points. Any permutation π over $(1, 2, \ldots, k)$ effectively specifies an order for those observation time-points. For any $P? \in \mathcal{OT}$, let $\pi(P?) \in \{1, 2, ..., k\}$ denote the (integer) position of $P?$ in the order determined by π; and let Π_k denote the set of all permutations over $(1, 2, \ldots, k)$.

▶ **Definition 10** (π-Execution Strategy). A *π-execution strategy* for a CSTNU $\mathcal{S}$ with k contingent links is a mapping, $\sigma\colon (\mathcal{I} \times \Omega) \to (\Psi \times \Pi_k)$, where for each drama $r = (s, \omega) \in \mathcal{I} \times \Omega$, $\sigma(s, \omega)$ is a pair (ψ_r, π_r) such that $\psi_r\colon \mathcal{T} \to \mathbb{R}$ is a schedule, and $\pi_r \in \Pi_k$ determines an order of dependency among the observation time-points. For convenience, π_r is extended such that $\pi_r(C) = 0$ for each contingent time-point C, and $\pi_r(X) = \infty$ for each non-contingent, non-observation time-point X.

The strategy σ is *viable* if for each drama $r = (s, \omega)$, the schedule ψ_r is a solution to the projection $\mathit{Prj}(\mathcal{S}, s, \omega)$. And σ is *coherent* if for each drama $r = (s, \omega)$, and any $P?$ and $Q?$ in $\mathcal{OT}$, $[\psi_r]_{P?} < [\psi_r]_{Q?}$ implies $\pi_r(P?) < \pi_r(Q?)$ (i.e., if *ψ_r schedules* $P?$ before $Q?$, then *π_r orders* $P?$ before $Q?$).

▶ **Definition 11** (π-History). Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O}, \mathcal{L} \rangle$ be a CSTNU with k contingent links; σ a π-execution strategy for $\mathcal{S}$; $r = (s, \omega)$ a drama; $(\psi_r, \pi_r) = \sigma(s, \omega)$; $t \in \mathbb{R}$; and

[1] $C_i - A_i = \omega_i$ abbreviates the pair of constraints, $C_i - A_i \le \omega_i$ and $A_i - C_i \le -\omega_i$.

Table 1 Morris-Muscettola rules for DC-checking STNUs.

Rule	Conditions	Pre-existing and generated edges
No Case (NC):		$W \xleftarrow{v} Y \xleftarrow{u} X$ $W \xleftarrow{u+v} X$
Upper Case (UC):		$A \xleftarrow{C:v} Y \xleftarrow{u} X$ $A \xleftarrow{C:u+v} X$
Lower Case (LC):	$(v < 0)$	$X \xleftarrow{v} C \xleftarrow{c:u} A$ $X \xleftarrow{u+v} A$
Cross Case (CC):	$(D \not\equiv C$ and $v < 0)$	$X \xleftarrow{D:v} C \xleftarrow{c:u} A$ $X \xleftarrow{D:u+v} A$
Label Removal (LR):	$(v \geq -x)$	$C \xleftarrow{c:x} A \xleftarrow[v]{C:v} X$

$d \in \{1, 2, \ldots, k; \infty\}$. Then $\mathcal{H}(t, d, s, \omega, \sigma) = (\mathcal{H}_s, \mathcal{H}_\omega)$ is the π-*history* of (t, d) for the drama (s, ω) and strategy σ, where:

$$\mathcal{H}_s = \{(p, s(p)) \mid P? \in \mathcal{OT},\ [\psi_r]_{P?} \leq t,\ \text{and}\ \pi_r(P?) < d\}\,;$$
$$\mathcal{H}_\omega = \{(A, C, [\psi_r]_C - [\psi_r]_A) \mid \exists x, y \text{ such that } (A, x, y, C) \in \mathcal{L},\ \text{and}\ [\psi_r]_C \leq t\}\,.$$

$\mathcal{H}_s$ specifies the truth values of all propositions p observed *before* time t in the schedule ψ_r, as well as those observed *at* time t if $P?$ is ordered *before* position d by the permutation π_r. And $\mathcal{H}_\omega$ specifies the durations of all contingent links that completed *at or before* time t in the schedule ψ_r.

▶ **Definition 12** (π-Dynamic Execution Strategy). A π-execution strategy, σ, for a CSTNU $\mathcal{S}$, is called π-*dynamic* if for every pair of dramas, (s_1, ω_1) and (s_2, ω_2), and every *non-contingent* (but possibly observation) time-point X:

let: $(\psi_1, \pi_1) = \sigma(s_1, \omega_1)$ and $(\psi_2, \pi_2) = \sigma(s_2, \omega_2)$,
let: $t = [\psi_1]_X$, and $d = \pi_1(X) \in \{1, 2, \ldots, |\mathcal{OT}|; \infty\}$.
if: $\mathcal{H}(t, d, s_1, \omega_1, \sigma) = \mathcal{H}(t, d, s_2, \omega_2, \sigma)$
then: $[\psi_2]_X = t$ and $\pi_2(X) = d$.

Thus, if, in the drama (s_1, ω_1), σ executes X at time t and position d, and the relevant histories are the same then, in the drama (s_2, ω_2), σ must also execute X at t and d.

▶ **Definition 13** (π-DC). A CSTNU, $\mathcal{S}$, is π-*dynamically controllable* (π-DC) if there exists a π-execution strategy for $\mathcal{S}$ that is both viable and π-dynamic.

4 DC-Checking for STNUs and CSTNs

This section summarizes the DC-checking algorithms for STNUs and CSTNs, due to Morris and Muscettola [20] and Hunsberger and Posenato [12], respectively, that play important roles in our *new* CSTNU DC-checking algorithms.

4.1 DC checking for STNUs

Table 1 lists the five constraint-propagation rules for STNUs due to Morris and Muscettola [20].[2] The *No Case* rule captures standard constraint propagation for STNs. The *Upper Case* rule generates a *conditional* constraint that guards against the possibility that the contingent duration $C - A$ might take on its maximum value. It can be glossed as: "While

[2] Later STNU algorithms [18, 19] use techniques that do not readily transfer to CSTNUs.

C remains unexecuted, X must *wait* at least $-u-v$ after the execution of A."[3] The *Lower Case* rule generates a constraint that guards against $C-A$ taking on its minimum value. The *Cross Case* rule generates a conditional constraint that guards against one contingent duration $C-A$ taking on its minimum value, while another $D-X$ takes on its maximum value. The *Label Removal* rule specifies when a conditional constraint has the force of an unconditional constraint.

The Morris-Muscettola DC-checking algorithm applies the rules from Table 1 in at most $O(N^2)$ rounds, at a cost of $O(N^3)$ per round. Afterward, it computes the *AllMax* STN, which is the *STN* projection in which each contingent link is set to its maximum duration. The *AllMax* STN is computed from the *fully propagated* STNU by: (1) removing all lower-case edges; and (2) removing the upper-case letters from all (original or generated) upper-case edges. If the *AllMax* STN is consistent, then the STNU is declared to be DC.

4.2 π-DC checking for CSTNs

Hunsberger et al. [14] presented a 6-rule IR-DC-checking algorithm for CSTNs ("IR" for "instantaneous reaction") that is based on the propagation of labeled constraints. Hunsberger and Posenato [12] subsequently introduced a faster, 3-rule version of their algorithm, called the π-DC-checking algorithm, which is used in this paper. The π-DC-checking algorithm generates constraints whose labels may include *q-literals,* such as $?p$, that indicate that a constraint need only hold as long as the value of p is unknown.

▶ **Definition 14** (Q-literals, q-labels). If $p \in \mathcal{P}$, then $?p$ is a *q-literal,* a *q-label* is a (possibly empty) conjunction of literals and/or q-literals, and $\mathcal{Q}^*$ denotes the set of all q-labels. For example, $p(?q)\neg r$ and $(?q)(?r)t\neg u$ are both q-labels.

The $\star$ operator extends ordinary conjunction to q-labels. Intuitively, if constraint C_1 is labeled by p, and constraint C_2 is labeled by $\neg p$, then *both* C_1 and C_2 must hold as long as p is unknown, which is represented by $p \star \neg p \ = ?p$.

▶ **Definition 15** ($\star$). The operator, $\star$: $\mathcal{Q}^* \times \mathcal{Q}^* \to \mathcal{Q}^*$, is defined thusly. First, for any $p \in \mathcal{P}$, $p \star p = p$ and $\neg p \star \neg p = \neg p$; otherwise, for any $p_1, p_2 \in \{p, \neg p, ?p\}$, $p_1 \star p_2 = ?p$. Next, for any $\ell_1, \ell_2 \in \mathcal{Q}^*$, $\ell_1 \star \ell_2 \in \mathcal{Q}^*$ denotes the conjunction obtained by applying $\star$ in pairwise fashion to matching literals from ℓ_1 and ℓ_2, and conjoining any unmatched literals. For example: $(p\neg q(?r)t) \star (qr\neg s) = p(?q)(?r)\neg st$.

Table 2 lists the sound-and-complete propagation rules for the π-DC-checking algorithm for CSTNs. The LP rule implements ordinary STN constraint propagation except that the labels, α and β, from the parent edges are conjoined in the generated edge. The qR_0 rule stipulates that a lower-bound constraint on $P?$ cannot depend on the value of p determined by executing $P?$. The qR_3^* rule specifies when an occurrence of $p, \neg p$ or $?p$ can be removed from a propositional label. The qR_3^* rule can generate edges whose labels are q-labels.

The π-DC-checking algorithm applies the rules from Table 2 until either (Non-DC) a negative self-loop with a consistent label is found; or (DC) no new edges can be generated. The completeness proof for the π-DC-checking algorithm shows how, in positive instances, to construct the *earliest-first strategy,* whose execution decisions are based on tracking the *current partial scenario* and computing *effective lower bounds* for unexecuted time-points. The *spreading lemma* ensures that lower-bound execution constraints are already present

[3] Wait constraints are only relevant if the wait time, $-u-v$, is positive (i.e., if $u+v<0$).

Table 2 Constraint-propagation rules for π-DC-checking CSTNs.

Rule	Conditions	Pre-existing and Generated Edges
LP:	$u+v<0,\ \alpha\beta \in \mathcal{P}^*$	$$\begin{tikzcd} Z & Y \arrow[l, "\langle v,\beta\rangle"'] & X \arrow[l, "\langle u,\alpha\rangle"'] \arrow[ll, bend left, "\langle u+v,\alpha\beta\rangle"] \end{tikzcd}$$
$\mathbf{qR}_0$:	$w<0,\ \alpha \in \mathcal{Q}^*$	$$\begin{tikzcd} Z & P? \arrow[l, "\langle w,\alpha\tilde{p}\rangle"', "\langle w,\alpha\rangle"] \end{tikzcd}$$
$\mathbf{qR}_3^*$:	$w<0,\ \alpha,\beta \in \mathcal{Q}^*$	$$\begin{tikzcd} Y \arrow[r, "\langle v,\beta\tilde{p}\rangle", "\langle \max\{v,w\},\alpha\star\beta\rangle"'] & Z & P? \arrow[l, "\langle w,\alpha\rangle"'] \end{tikzcd}$$

In each rule, $X,Y \in \mathcal{T}$; $P? \in \mathcal{OT}$; and $Z = 0$. In qR_0 and qR_3^*, $\tilde{p} \in \{p, \neg p, ?p\}$; and p does not appear in α or β (in any form).

in the fully propagated network, courtesy of the qR_0 and qR_3^* rules. The proof also shows that upper-bound execution constraints cannot generate negative loops in the relevant STN projection. Termination is guaranteed by inserting a global upper bound (or *horizon*), whose value is $h = nM$, where $n = |\mathcal{T}|$ and M is the maximum absolute value of any negative edge in the network. The horizon constraints do not affect the DC property, assuming that all edge weights are rational [2].

An upper bound for the computational complexity of the algorithm can be obtained assuming that each possible labeled value of each edge heading to Z must be updated M times and propagated to all other edges: $O(|\mathcal{T}|(3^{|\mathcal{P}|}|\mathcal{T}|M)) = O(M|\mathcal{T}|^2 3^{|\mathcal{P}|})$. Although exponential in the worst case, it has been shown to be practical across a variety of networks.

5 Algorithm 1: Reducing CSTNU-DC to CSTN-DC

This section introduces a novel DC-checking algorithm for CSTNUs that first transforms its input CSTNU $\mathcal{S}$ into a *DC-equivalent* CSTN $\mathcal{S}'$, and then applies the π-DC-checking algorithm for CSTNs to $\mathcal{S}'$. The transformation for contingent links is illustrated below. For each contingent link, (A, x, y, C), a new observation time-point $P_c?$ is introduced that is constrained to occur exactly x after A. Executing $P_c?$ generates a value for p_c that determines whether the duration $C - A$ shall be x or y.[4] If $p_c = \top$, then C must co-occur with $P_c?$ (i.e., x after A); otherwise, C must execute exactly $y - x$ after $P_c?$ (i.e., y after A). Because the CSTNU has been transformed into a CSTN, the horizon value $h = nM$ can be applied to that CSTN without affecting the DC property. The computational cost of this CSTNU-to-CSTN transformation is $O(|\mathcal{L}|)$ (i.e., linear).

$$\begin{tikzcd} A \arrow[r, shift left, "\langle x,\boxdot\rangle"] & P_c? \arrow[l, shift left, "\langle -x,\boxdot\rangle"] \arrow[r, shift left, "{\langle 0,p_c\rangle,\ \langle y-x,\boxdot\rangle}"] & C \arrow[l, shift left, "{\langle 0,\boxdot\rangle,\ \langle x-y,\neg p_c\rangle}"] \end{tikzcd}$$

6 Algorithm 2: Propagating in the CSTNU

This section introduces a novel DC-checking algorithm for CSTNUs that propagates constraints in the CSTNU using the rules in Table 3. The names of the rules reflect the STNU and CSTN rules from Tables 1 and 2 that they generalize, except that z! is a new kind of rule that *forward propagates* upper-case a-labels. The z! rule is not needed for DC-checking STNUs,

[4] Cairo and Rizzi [3] proved that restricting contingent durations to be either the minimum or maximum value, but nothing in between, does not affect the DC property.

Table 3 Constraint-propagation rules for CSTNU Algorithm 2.

Rule	Conditions	Pre-existing and Generated Edges
(zLp/Nc/Uc)	$u+v<0,\ \alpha\beta\in\mathcal{P}^*$	$Z \xleftarrow{\langle v,\aleph,\beta\rangle} Y \xleftarrow{\langle u,\diamond,\alpha\rangle} X$ $Z \xleftarrow{\langle u+v,\aleph,\alpha\beta\rangle} X$
(zLc/Cc)	$x+v<0,\ C\notin\aleph,\ \beta\in\mathcal{P}^*$	$Z \xleftarrow{\langle v,\aleph,\beta\rangle} C \xleftarrow{\langle x,c,\boxdot\rangle} A$ $Z \xleftarrow{\langle x+v,\aleph,\beta\rangle} A$
(z!)	$-y+v<0,\ \beta\in\mathcal{P}^*$	$Z \xleftarrow{\langle v,\aleph,\beta\rangle} A \xleftarrow{\langle -y,C,\boxdot\rangle} C$ $Z \xleftarrow{\langle -y+v,C\aleph,\beta\rangle} C$
(zLr)	$m=\max\{v,w-x\},\ C\ \notin\ \aleph\aleph_1,$ $\beta,\gamma\in\mathcal{Q}^*$	$Y \xrightarrow{\langle v,C\aleph,\beta\rangle} Z \xleftarrow{\langle w,\aleph_1,\gamma\rangle} A \xrightarrow{\langle x,c,\boxdot\rangle} C$ $Y \xrightarrow{\langle m,\aleph\aleph_1,\beta\star\gamma\rangle} Z$
(zqR$_0$)	$w<0;\ \tilde{p}\in\{p,\neg p,?p\};\ \tilde{p}\beta\in\mathcal{Q}^*,$	$Z \xleftarrow{\langle w,\aleph,\beta\tilde{p}\rangle} P?$ $Z \xleftarrow{\langle w,\aleph,\beta\rangle} P?$
(zqR$_3^*$)	$w<0;\ \tilde{p}\in\{p,\neg p,?p\};\ \tilde{p}\beta,\gamma\in\mathcal{Q}^*,$	$Y \xrightarrow{\langle v,\aleph,\beta\tilde{p}\rangle} Z \xleftarrow{\langle w,\aleph_1,\gamma\rangle} P?$ $Y \xrightarrow{\langle \max\{v,w\},\aleph\aleph_1,\beta\star\gamma\rangle} Z$

$Z=0;\ A,C,X,Y\in\mathcal{T};\ C$ is contingent; $P?\in\mathcal{OT};\ \aleph,\aleph_1\in\mathcal{A}_u^*$.

```
Algorithm 2: CSTNU-DC-CH($\mathcal{S}$).
Input: $\mathcal{S}=\langle\mathcal{T},\mathcal{P},\mathcal{C},\mathcal{OT},\mathcal{O},\mathcal{L}\rangle$: a CSTNU instance
Output: the dynamic controllability status of $\mathcal{S}$.
$G$ = graph for $\mathcal{S}$
$h = M|\mathcal{T}|$, where $M$ is the maximum absolute value of any negative edge
foreach $X\in\mathcal{T}$ do
    Add the edges, $Z\xrightarrow{\langle h,\diamond,\boxdot\rangle}X$ and $X\xrightarrow{\langle 0,\diamond,\boxdot\rangle}Z$, to $G$.
do
    $G = \mathrm{zqR}_0(G)$                                  // Label Modification
    $G = \mathrm{zqR}_3^*(G)$
    $G = zLp/Nc/Uc(G)$                                       // Edge Generation
    $G = z!(G)$
    $G = zLc/Cc(G)$
    $G = zLr(G)$
    if (any negative self-loop with a p-label has been found) then return not DC
while (rules continue to generate new edges)
return DC
```

but is needed to ensure completeness for CSTNU DC-checking. Note that z! and zqR$_3^*$ can generate *conjunctions* of upper-case a-labels, which can be handled by all of the other rules.

To ensure termination, the algorithm inserts the same horizon constraints seen earlier, then it exhaustively applies the rules from Table 3. It outputs *not DC* if a negative self-loop with a consistent p-label is found; otherwise, *DC*. The pseudo-code for the algorithm is given in Algorithm 2.

We begin with relevant definitions, then prove soundness and completeness.

▶ **Definition 16** (Precedes). Let σ be a π-dynamic strategy, (s,ω) any drama, and $(\psi,\pi)=\sigma(s,\omega)$. For any $X,Y\in\mathcal{T}$, if either $[\psi]_X<[\psi]_Y$ or $([\psi]_X=[\psi]_Y$ and $\pi(X)<\pi(Y))$ then we say that X *precedes* Y in (ψ,π), notated $X\prec_\psi^\pi Y$.

If $X\prec_\psi^\pi P?$, then the decision to execute X cannot depend on the observation of p. In addition, if X and Y are distinct time-points, and at least one of them is an observation time-point, then $X \prec^{\pi}_{\psi} Y$ if and only if $\neg(Y \prec^{\pi}_{\psi} X)$.

▶ **Definition 17** (Satisfy a Labeled Constraint)**.** A π-execution strategy σ *satisfies the labeled constraint* $(Y - X \leq \delta, \ell)$, where $\ell \in \mathcal{P}^*$, if, for each drama (s, ω), either $s(\ell) = \bot$ or $[\psi]_Y - [\psi]_X \leq \delta$, where $(\psi, \pi) = \sigma(s, \omega)$.

▶ **Definition 18** (Satisfy a Contingent Link)**.** A π-execution strategy σ *satisfies the contingent link* (A_i, x_i, y_i, C_i) if for each drama (s, ω), $[\psi]_{C_i} - [\psi]_{A_i} = \omega_i$. In such a case, we also say that σ satisfies the lower-case and upper-case edges associated with that contingent link.

From Defns. 7 and 12, it follows that a viable π-execution strategy σ must satisfy all of the (original) labeled constraints in $\mathcal{S}$ (before any constraint propagation) and all of the (original) lower- and upper-case edges in $\mathcal{S}$.

A constraint-propagation rule is sound if whenever a viable and dynamic σ satisfies the pre-existing edge(s) in that rule, σ must also satisfy the edge generated by that rule. Now, the rules in Table 3 only generate edges pointing at Z, which represent lower-bound constraints; however, the generated edges may have a-labels with multiple UC letters and q-labels (e.g., see rules z!, zLr and zqR$_3^*$); and many of the rules can propagate such labeled values. Therefore, the semantics of satisfying a lower-bound edge must accommodate such a-labels and q-labels.

▶ **Definition 19** (Satisfy a Lower-Bound Constraint)**.** A π-execution strategy σ *satisfies the lower-bound constraint* $(Y \geq \delta, \langle\aleph, \beta\rangle)$ represented by the edge from Y to Z labeled by $\langle -\delta, \aleph, \beta\rangle$, where $\beta \in \mathcal{Q}^*$, and $\aleph \in \mathcal{A}_u^*$, if for each drama (s, ω), *any* of the following hold, where $(\psi, \pi) = \sigma(s, \omega)$:

(1) $[\psi]_Y \geq \delta$;

(2) for some $C_i \in \aleph$, where $(A_i, x_i, y_i, C_i) \in \mathcal{L}$, $\omega_i < y_i$ (i.e., the i^{th} contingent link does *not* take on its maximum duration);

(3) for some $p \in \beta$, $s(p) = \bot$;

(4) for some $\neg p \in \beta$, $s(p) = \top$; or

(5) for some $?p \in \beta$, $P? \prec^{\pi}_{\psi} Y$.

If $(\text{Z} - Y \leq -\delta, \beta)$ (i.e., $(Y \geq \delta, \beta)$) is a labeled constraint in a CSTNU (prior to any propagation), then $\beta \in \mathcal{P}^*$, and the corresponding edge in the graph has the labeled value $\langle -\delta, \diamond, \beta\rangle$. For this edge, clauses (2) and (5) in Defn. 19 are vacuous, whence satisfaction reduces to: $[\psi]_Y \geq \delta$ or $s(\beta) = \bot$. Thus, Defn. 19 reduces to Defn. 17 for original labeled edges that happen be lower-bound edges.

More generally, it will be useful to note that for any lower-bound edge labeled by $\langle -\delta, \aleph, \beta\rangle$, where $\beta \in \mathcal{P}^*$, satisfaction (i.e., Defn. 19) reduces to:

(i) $[\psi]_Y \geq \delta$;

(ii) for some $C_i \in \aleph$, where $(A_i, x_i, y_i, C_i) \in \mathcal{L}$, $\omega_i < y_i$; or

(iii) $s(\beta) = \bot$. (‡)

▶ **Definition 20** (Soundness)**.** A constraint-propagation rule is *sound* if whenever a *viable* and π-*dynamic* execution strategy σ satisfies the rule's pre-existing (parent) edges, it also satisfies the rule's generated (child) edge.

Note. In each of the soundness proofs below, σ is assumed to be a viable and π-dynamic strategy that satisfies the parent edges in the rule under consideration. Note, too, that soundness proofs for the (zqR$_0$), (zqR$_3^*$) and (zLp/Nc/Uc) rules are skipped to save space.

▶ **Lemma 21.** *The (zLc/Cc) rule from Table 3 is sound.*

Proof. Suppose σ does *not* satisfy the generated edge from A to Z in rule (zLc/Cc). Since the p-label $\alpha\beta$ on the generated edge is consistent, it follows from Defn. 19 that there is some drama (s, ω) such that *all* of the following hold:

(¬ i) $[\psi]_A < -x - v$;
(¬ ii) for each $C_i \in \aleph$, where $(A_i, x_i, y_i, C_i) \in \mathcal{L}$, $\omega_i = y_i$; and
(¬ iii) $s(\alpha\beta) = \top$.

First, (¬ iii) implies that $s(\alpha) = \top$ and $s(\beta) = \top$. Therefore, since σ satisfies the edge from C to Z, (¬ ii) implies that $[\psi]_C \geq -v$, by Defn. 19. Next, let ω' be the same as ω except that the contingent link AC takes on its *minimum* value x; and let $(\psi', \pi') = \sigma(s, \omega')$. Since σ is viable, $[\psi']_C - [\psi']_A = x$. However, since $C \notin \aleph$, (¬ ii) also holds for ω'; thus, $[\psi']_C \geq -v$ must hold. And, since the only difference between (s, ω) and (s, ω') is the duration of the contingent link AC, the first difference between ψ and ψ' must occur when C executes, which happens *after* A executes. Thus, $[\psi]_A = [\psi']_A = [\psi']_C - x \geq -v - x$, contradicting (¬ i). ◀

▶ **Lemma 22.** *The (z!) rule from Table 3 is sound.*

Proof. Let (s, ω) be any drama for which all $C_i \in C\aleph$ take on their maximum durations (i.e., $\omega_i = y_i$), and such that $s(\alpha\beta) = \top$; and let $(\psi, \pi) = \sigma(s, \omega)$. Then $s(\beta) = \top$ and all $C_i \in \aleph$ take on their maximum durations. Therefore, since σ satisfies the parent edge from A to Z, it follows that $[\psi]_A \geq -v$. Next, since $s(\alpha) = \top$, and C takes on its maximum duration, then $[\psi]_C = [\psi]_A - y \geq -v - y$. Thus, σ satisfies the generated edge from C to Z. ◀

▶ **Lemma 23.** *The (zLr) rule from Table 3 is sound.*

Proof. Suppose that σ does *not* satisfy the generated edge in rule (zLr). Then, by Defn. 19, there is a drama (s, ω) for which *all* of the following hold:

($1^\dagger$) $[\psi]_Y < -m$;
($2^\dagger$) for each $C_i \in \aleph\aleph_1$, where $(A_i, x_i, y_i, C_i) \in \mathcal{L}$, $[\psi]_{C_i} - [\psi]_{A_i} = y_i$;
($3^\dagger$) for each $p \in \beta \star \gamma$, $s(p) = \top$;
($4^\dagger$) for each $\neg p \in \beta \star \gamma, s(p) = \bot$; and
($5^\dagger$) for each $?p \in \beta \star \gamma$, $\neg(P? \prec^\pi_\psi Y)$.

where $(\psi, \pi) = \sigma(s, \omega)$. In addition, since σ is valid and satisfies the parent edge from Y to Z, one of the following must hold, by (‡), above:

(1) $[\psi]_Y \geq -v$;
(2) for some $C' \in C\aleph$, where $(A', x', y', C') \in \mathcal{L}$, $[\psi]_{C'} - [\psi]_{A'} < y'$;
(3) for some $p \in \beta$, $s(p) = \bot$;
(4) for some $\neg p \in \beta$, $s(p) = \top$; or
(5) for some $?p \in \beta$, $P? \prec^\pi_\psi Y$.

Now, (1) contradicts ($1^\dagger$), since $-v \geq -m$. (2) holding for some $C' \in \aleph$ would contradict ($2^\dagger$). And (5) contradicts ($5^\dagger$), since $?p \in \beta$ implies $?p \in \beta \star \gamma$. Therefore, either (2) holds for $C' = C$ (i.e., $[\psi]_C - [\psi]_A < y$) or some instance(s) of (3) or (4) hold(s).

Suppose some instance(s) of (3) or (4) hold(s). For (3), if $p \in \beta$ and $s(p) = \bot$, then to avoid contradicting ($3^\dagger$), we must have $?p \in \beta \star \gamma$, which, by ($5^\dagger$), implies that $\neg(P? \prec^\pi_\psi Y)$. We can assume Y and $P?$ are distinct because any occurrence of p in β could be removed by (zqR$_0$). Therefore, $Y \prec^\pi_\psi P?$ must hold. A similar argument applies to any occurrence of $\neg p \in \beta$ that makes (4) hold. Thus, Y must precede any $P?$ for which p or $\neg p$ makes (3) or (4) hold, respectively.

Let s' equal s, except that: if $p \in \beta$ makes (3) hold, then $s'(p) = \top$; and if $\neg p \in \beta$ makes (4) hold, then $s'(p) = \bot$. Since σ satisfies the parent edge from Y to Z, one or more clauses from Defn. 19 must hold for $(\psi', \pi') = \sigma(s', \omega)$. By construction, (3) and (4) do *not* hold for s'; thus, one of the following must hold:

(1′) $[\psi']_Y \geq -v$;
(2′) for some $C' \in C\aleph, [\psi']_{C'} - [\psi']_{A'} < y'$; or
(5′) for some $?p \in \beta,\ \ P? \prec^{\pi'}_{\psi'} Y$.

Now (ψ, π) and (ψ', π') each determine a sequence of events that can be ordered, first by execution time and, second, for simultaneous events, by order of dependence. Let t' be the *earliest* time at which the two sequences differ. By construction, it must be where some $[\psi]_{R?} = [\psi']_{R?} = t'$, but $s(r) \neq s'(r)$. Furthermore, $Y \prec^{\pi}_{\psi} R?$ and, thus, by the definition of t', $[\psi']_Y = [\psi]_Y$. But then $(1^\dagger)$ implies that $[\psi']_Y = [\psi]_Y < -m \leq -v$, whence $(1')$ is false. As for $(5')$, if $?q \in \beta$ (and hence $?q \in \beta \star \gamma$) and $Q? \prec^{\pi'}_{\psi'} Y$, then $[\psi']_{Q?} \leq t'$, whence $[\psi']_{Q?} = [\psi]_{Q?}$ and, thus, $Q? \prec^{\pi}_{\psi} Y$, which contradicts $(5^\dagger)$. Thus, $(2')$ must hold for some $C' \in C\aleph$. Since σ is valid, and the contingent durations in ω did not change from (s, ω) to (s', ω), $(2')$ and (2) are equivalent. Thus, the only possibility is that (2) holds for $C' = C$ (i.e., $[\psi]_C - [\psi]_A < y$).

Next, suppose that $[\psi]_Y < [\psi]_C$. Let ω^+ be the same as ω except that $C - A = y$. It is not hard to check that in the drama (s', ω^+), conditions $(1^\dagger)$–$(5^\dagger)$ all hold, but that *none* of the conditions $(1')$–$(5')$ can hold. (Changing to the situation ω^+ removed the last possibility (i.e., that $C - A < y$).) But that contradicts that σ satisfies the parent edge from Y to Z. Thus, $[\psi]_Y \geq [\psi]_C$.

Next, since σ satisfies the edge from A to Z, by Defn. 19, one of these must hold:

(1_A) $[\psi]_A \geq -w$;
(2_A) for some $C' \in \aleph_1,\ \ [\psi]_{C'} - [\psi]_{A'} < y'$;
(3_A) for some $p \in \gamma,\ \ s(p) = \bot$;
(4_A) for some $\neg p \in \gamma,\ \ s(p) = \top$; or
(5_A) for some $?p \in \gamma,\ \ P? \prec^{\pi}_{\psi} A$.

Now, $(2^\dagger)$ contradicts (2_A). And $(5^\dagger)$ contradicts (5_A). (We can assume that A and $P?$ are distinct since, otherwise, rule (zqR$_0$) could have been used to remove any occurrence of $P?$ from γ.) And $[\psi]_A \leq [\psi]_C - x \leq [\psi]_Y - x < -m - x \leq -w$ implies (1_A) is false. (The inequalities follow from σ being viable, from $[\psi]_C \leq [\psi]_Y$, from $(1^\dagger)$, and $m = \max\{v, w - x\}$.) Finally, any $?p$ making (5_A) true yields $[\psi]_{P?} \leq [\psi]_A < [\psi]_C \leq [\psi]_Y$, whence $P? \prec^{\pi}_{\psi} Y$, contradicting $(5^\dagger)$. Thus, (3_A) or (4_A) must hold.

Now, suppose that some p makes both (3) and (3_A) true. Then $p \in \beta \star \gamma$ and $s(p) = \bot$, contradicting $(3^\dagger)$. Thus, the letters that make (3) true, if any, must be distinct from the letters that make (3_A) true, if any. Similarly, the letters that make (4) true must be distinct from those that make (4_A) true. In addition, any p that makes (3) true requires $s(p) = \bot$, which implies that p cannot simultaneously make (4_A) true; and any p that makes (4) true cannot simultaneously make (3_A) true. In short, the letters that make (3) or (4) true are *distinct* from those that make (3_A) or (4_A) true. And, to avoid contradicting $(3^\dagger)$ or $(4^\dagger)$, for any $\pm p$ that makes (3), (4), (3_A) or (4_A) true, $?p$ must be in $\beta \star \gamma$, whence $(5^\dagger)$ yields that Y must precede $P?$ (i.e., $Y \prec^{\pi}_{\psi} P?$).

So, let s'' be the same as s' except that if p makes either (3_A) or (4_A) true, then $s''(p) \neq s'(p) = s(p)$. (Since s and s' only differ on letters that make (3) or (4) true, and since those letters are distinct from the letters making (3_A) or (4_A) true, s and s' must agree on all letters that make (3_A) or (4_A) true.) Let $(\psi'', \pi'') = \sigma(s'', \omega)$. Let t'' be the first time when the events in (ψ'', π'') and (ψ, π) differ. Then for some $U?$, $[\psi'']_{U?} = [\psi]_{U?} = t''$ and $s''(u) \neq s(u)$; and Y precedes $U?$ in (ψ'', π'') and (ψ, π). Therefore, $[\psi'']_Y = [\psi']_Y = [\psi]_Y \leq t'' \leq t'$.

Since σ satisfies the edge from A to Z, one or more clauses from Defn. 19 must hold for that edge. By construction, the only candidates are:
($1''_A$) $[\psi'']_A \geq -w$;
($2''_A$) for some $C' \in \aleph_1$, $[\psi'']_{C'} - [\psi'']_{A'} < y'$; or
($5''_A$) for some $?p \in \gamma$, $P? \prec^{\pi''}_{\psi''} A$.
Since all events occurring before time t'' are executed identically by (ψ, π) and (ψ'', π''), (1_A) being false implies that $(1''_A)$ must also be false, since $[\psi]_A < [\psi]_Y \leq t''$. Similarly, (2_A) being false implies that $(2''_A)$ must also be false. Finally, if $?g \in \gamma$ makes $(5''_A)$ true, that contradicts $(5^\dagger)$, since $?g \in \beta \star \gamma$. Therefore, all cases lead to a contradiction. ◀

▶ **Theorem 24.** *The rules from Table 3 are complete for π-DC checking for CSTNUs.*

Proof. Let $\mathcal{S}$ be any CSTNU; let $\mathcal{S}^*$ be the CSTNU obtained by fully propagating $\mathcal{S}$ using the rules from Table 3; and suppose that no negative loop with a consistent p-label was found and, thus, the DC-checking algorithm returned *DC*. Let $\mathcal{S}^*_x$ be the *AllMax* CSTN obtained by deleting all LC edges from $\mathcal{S}^*$ and removing all UC a-labels from labeled values in $\mathcal{S}^*$. By construction, the *AllMax* CSTN must already be fully propagated. To see this, note that ignoring the a-labels in the (zLp/Nc/Uc), zqR$_0$ and zqR$^*_3$ rules for CSTNUs from Table 3 reduces them to the LP, qR$_0$ and qR$^*_3$ rules for CSTNs from Table 2, respectively. Since no negative loop with consistent p-label was found by the CSTNU DC-checking algorithm, none exist in the *AllMax* CSTN; hence it too must be DC.

Construct the *earliest-first strategy* σ for $\mathcal{S}$, as follows. Let α be the *current partial scenario* (CPS), initially $\boxdot$; and let $\mathcal{T}_u$ be the unexecuted time-points, initially $\mathcal{T}\backslash\{\text{Z}\}$. For each $X \in \mathcal{T}_u$, compute its *effective lower bound:* $\text{ELB}(X) = \max\{\delta \mid \exists (X \geq \delta, \ell) \in \mathcal{S}^*_x, \mathit{appl}(\ell, \alpha)\}$.[5] Let (Λ, χ) be the first execution decision: "if nothing happens before time Λ, then execute the time-points in χ", where $\Lambda = \min\{\text{ELB}(X) \mid X \in \mathcal{T}_u\}$; and $\chi = \{X \in \mathcal{T}_u \mid \text{ELB}(X) = \Lambda\}$ [9].

Case 1: No contingent time-point executes before time Λ. For each active contingent link, (A_i, x_i, y_i, C_i), raise its lower bound to $\Lambda - a_i$, where $a_i = [\sigma(s)]_{A_i}$. This cannot introduce any new constraints into $\mathcal{S}^*$ or $\mathcal{S}^*_x$; thus, both are still DC. And, since no ELB values have changed, (Λ, χ) is the earliest-first decision for the CSTN $\mathcal{S}^*_x$. Thus, inserting the relevant execution constraints cannot introduce any negative loops into any relevant STN projection [12]. Remove any executed time-points from $\mathcal{T}_u$; update the CPS α to include any new observations; and delete any labeled values that are inconsistent with those observations.

Case 2: A contingent time-point C executes at some time $t \leq \Lambda$. Update $\mathcal{S}^*$, as follows. First, replace the labeled value $\langle -y, C, \boxdot\rangle$ on the original UC edge from C to A with $\langle -\delta, \diamond, \boxdot\rangle$, where $\delta = t - [\sigma(s)]_A$ is the observed duration for the link (A, x, y, C); and replace the labeled value $\langle x, c, \boxdot\rangle$ on the original lower-case edge from A to C by $\langle \delta, \diamond, \boxdot\rangle$. The execution semantics ensures that $\delta \in [x, y]$. Second, *remove* any labeled value $\langle w, \aleph, \beta\rangle$ from $\mathcal{S}^*$ for which $C \in \aleph$. Third, for each $X \in \mathcal{T}_u$, insert a lower-bound constraint, $(X \geq t, \alpha)$. (Although $\mathit{ELB}(X, \alpha) \geq \Lambda \geq t$, the ELB value could have been due to a C-labeled edge which has since been removed.) Finally, fully propagate the modified $\mathcal{S}^*$ CSTNU.

Suppose a negative loop with consistent p-label is discovered in the modified $\mathcal{S}^*$. Any such loop must include the edge from A to C since, otherwise, nothing could prevent the

[5] $\mathit{appl}(\ell, \alpha)$ holds if ℓ is *applicable* given the CPS α [14]. Formally, $\mathit{appl}(\ell, \alpha)$ holds if each p that appears in *both* ℓ and α appears as p in both or as $\neg p$ in both.

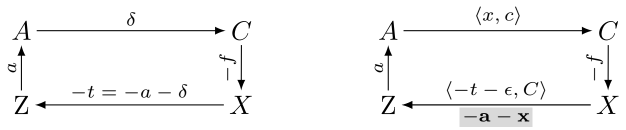


Figure 2 A negative loop in the modified (left) and original (right) CSTNU $\mathcal{S}^*$.

corresponding loop in the original $\mathcal{S}^*$ from being generated, which would be even more negative. First, consider the graphs shown in Fig. 2, where irrelevant details (e.g., p-labels) have been omitted to improve clarity. The lower-bound $X \geq t = a + \delta$ in the modified $\mathcal{S}^*$ generates a negative loop only if $-f < 0$. But that lower bound on X can only be new/relevant if X's original ELB value arose from a C-labeled edge as illustrated on the righthand side, where $-t - \epsilon < -t$. But then the (zLr) rule would have generated the shaded labeled value in the figure, whence propagating backward from X to C to A to Z, courtesy of the (zLp/Nc/Uc), (z!) and (zLc/Cc) rules, would have generated a loop of length $(-a - x) + (-f) + x + a = -f < 0$ in the original, a contradiction. The only other possibility is if the path from C to Z in Fig. 2 included an occurrence of the (formerly UC) edge from C to A. This case would require a negative path from C to C, which would have made the original $\mathcal{S}^*$ non-DC, a contradiction. Thus, the modified $\mathcal{S}^*$ is necessarily DC. Compute ELB values based on the updated and propagated $\mathcal{S}^*_x$ graph and continue recursively.

The construction of the strategy will be complete (and the network still consistent) once all time-points have been executed. ◀

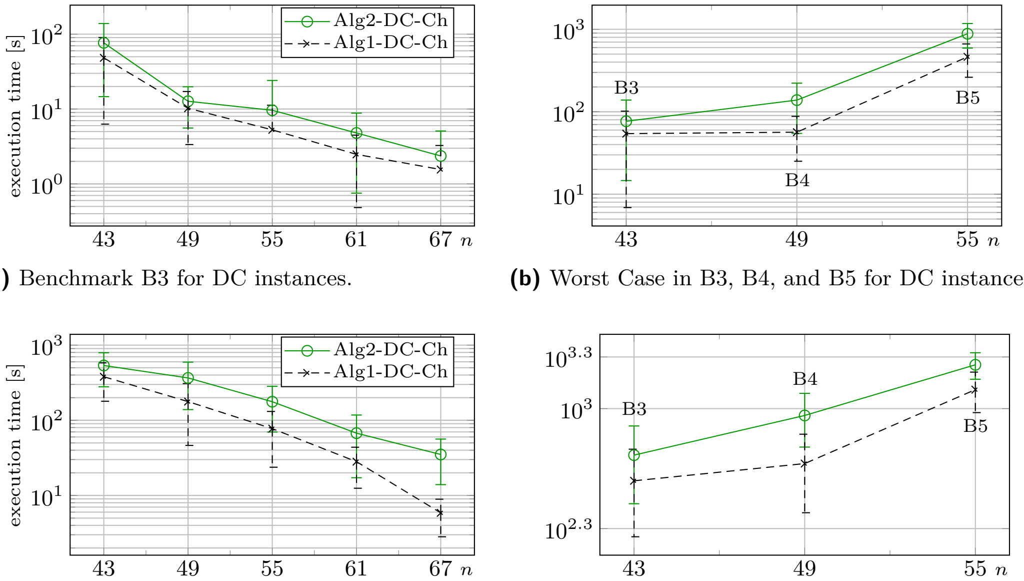
6.1 Computational complexity

An upper bound for the computational complexity of Algorithm 2 can be derived by adapting the original CSTN DC-checking algorithm complexity, while also considering the presence of a-labels. The three CSTN rules may have to consider all possible combinations of a-labels and q-labels. It is a matter of combinatoric operation to show that the complexity of such rules dominates the final upper bound, $O(M|\mathcal{T}|^2 3^{|\mathcal{P}|} 2^{|\mathcal{L}|})$, where M is the maximum absolute value of any negative weight in the graph.

7 Empirical Evaluation

This section presents an empirical comparison of the performance of the two DC-checking algorithms introduced in this paper. `Alg1-DC-Ch` is our implementation of Algorithm 1 (cf. Sect. 5), which converts CSTNUs to CSTNs; `Alg2-DC-Ch` is our implementation of Algorithm 2 (cf. Sect. 6) which directly propagates CSTNU constraints. Both algorithms were implemented in Java and executed on a JVM 8 on a Linux machine with an Intel(R) Xeon(R) E5-2637 @ 3.5 GHz and 128GB of RAM. The source code is freely available [22].

For testing, we built benchmarks with a structure similar to those proposed in earlier work [10] where each benchmark had DC CSTNs and non-DC CSTNs obtained from random workflow schemata generated by the ATAPIS toolset [15], parametrized by the number of activities, N, and the number of observations, $|\mathcal{P}|$. Here, we created three benchmarks, called B3, B4, and B5, each having 250 DC CSTNUs and 250 non-DC CSTNUs, obtained from random workflow schemata with (1) $N = 10$; (2) $|\mathcal{P}|$ equal to 3, 4 and 5, for B3, B4, and B5, respectively; and (3) representing activities as contingents links. Each benchmark is a composite of five sub-benchmarks, each having 50 instances generated by fixing also the number of parallel components in the generated workflow schemata. In particular, the



(a) Benchmark B3 for DC instances.

(b) Worst Case in B3, B4, and B5 for DC instances.

(c) Benchmark B3 for non-DC instances.

(d) Worst Case in B3, B4, and B5 for non-DC instances.

Figure 3 Experimental evaluation, $n = |\mathcal{T}|$.

maximum number of parallel components in a workflow was limited by the numbers of tasks and observations. Since the maximum number of observations was 5, we decided to consider, for each benchmark, 5 sub-benchmarks, each composed of CSTNU instances representing workflows where the number of parallel components was fixed at $0, 1, 2, 3$ or 4, respectively.

To conserve space, Figure 3a displays the average execution times of the two algorithms over all five sub-benchmarks of B3 considering DC instances. Each data point has a vertical error bar that represents a 95% confidence interval for the average execution time. With more parallel components, a random workflow can have more observations in parallel and, therefore, scenarios that are more independent of each other than scenarios that are nested. The corresponding CSTNU instance may have more nodes, but it can be easier to solve, as shown by the average execution times of the sub-benchmark in Figure 3a.

From the data in Figure 3a, and the results for benchmarks B4 and B5, it emerges that the most difficult instances are related to workflow schemata having no parallel connectors (i.e., instances belonging to the first sub-benchmark of each main benchmark). Therefore, Figure 3b reports only the average execution times obtained from instances of first sub-benchmark of B3, B4, and B5, respectively.

Figure 3c and Figure 3d show the results obtained when instances are non-DC. The two algorithms exhibit a worse performance checking non-DC instances than checking DC ones. In details, each algorithm require an average execution time that, at most, can be 8 times greater than the average execution time required for checking positive instances having the same order. We verified that such worse behavior is due to the fact that some instances have a similar configuration where a negative cycle emerge only after many iterations between lower bound and the global upper bound. We are investigating if it is possible to detect such configuration in advance in order to speed up the convergence.

However, Figure 3 demonstrates that Algorithm `Alg1-DC-Ch` tends to perform better, but the difference is not statistically significant.

8 Conclusions

This paper presented the first *practical, sound-and-complete* DC-checking algorithms for CSTNUs. The first algorithm converts an input CSTNU into a DC-equivalent CSTN, then runs an existing CSTN algorithm. The second directly propagates CSTNU constraints, using new rules that ensure completeness. An empirical evaluation demonstrated their practicality across a variety of benchmarks. Future work aims to determine whether adding new rules can speed up the algorithms.

References

1 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`.

2 Massimo Cairo, Luke Hunsberger, Roberto Posenato, and Romeo Rizzi. A Streamlined Model of Conditional Simple Temporal Networks - Semantics and Equivalence Results. 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`.

3 Massimo Cairo and Romeo Rizzi. Dynamic Controllability Made Simple. In *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*, volume 90 of *LIPIcs*, pages 8:1–8:16, 2017. `doi:10.4230/LIPIcs.TIME.2017.8`.

4 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`.

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

6 Carlo Combi, Luke Hunsberger, and Roberto Posenato. An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty - revisited. In *Agents and Artificial Intelligence*, volume 449 of *CCIS*, pages 314–331. Springer, 2014. `doi:10.1007/978-3-662-44440-5_19`.

7 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. IEEE Comp. Soc., 2010. `doi:10.1109/TIME.2010.17`.

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

9 Luke Hunsberger. Efficient execution of dynamically controllable simple temporal networks with uncertainty. *Acta Informatica*, 53(2):89–147, 2015. `doi:10.1007/s00236-015-0227-0`.

10 Luke Hunsberger and Roberto Posenato. Checking the Dynamic Consistency of Conditional 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`.

11 Luke Hunsberger and Roberto Posenato. A new approach to checking the dynamic consistency of conditional simple temporal networks. In *22nd International Conference* *on Principles and Practice of Constraint Programming, CP 2016*, pages 268–286, 2016. doi:10.1007/978-3-319-44953-1_18.

12 Luke Hunsberger and Roberto Posenato. Simpler and faster algorithm for checking the dynamic consistency of conditional simple temporal networks. In *26th International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 1324–1330, 2018. doi:10.24963/ijcai.2018/184.

13 Luke Hunsberger, Roberto Posenato, and Carlo Combi. The Dynamic Controllability of Conditional STNs with Uncertainty. In *PlanEx at ICAPS 2012*, pages 1–8, 2012.

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

15 Andreas Lanz and Manfred Reichert. Enabling time-aware process support with the atapis toolset. In *BPM Demo Sessions 2014*, volume 1295 of *CEUR Workshop Proceedings*, pages 41–45, 2014.

16 Richard Lenz and Manfred Reichert. It support for healthcare processes - premises, challenges, perspectives. *Data Knowl. Eng.*, 61(1):39–58, 2007. doi:10.1016/j.datak.2006.04.007.

17 Dian Liu, Hongwei Wang, Chao Qi, Peng Zhao, and Jian Wang. Hierarchical task network-based emergency task planning with incomplete information, concurrency and uncertain duration. *Knowledge-Based Systems*, 112:67–79, 2016. doi:10.1016/j.knosys.2016.08.029.

18 Paul Morris. A structural characterization of temporal dynamic controllability. In *CP 2006*, volume 4204, pages 375–389, 2006. doi:10.1007/11889205_28.

19 Paul Morris. Dynamic controllability and dispatchability relationships. In *Integration of AI and OR Techniques in Constraint Programming*, volume 8451 of *LNCS*, pages 464–479. Springer, 2014.

20 Paul H. Morris and Nicola Muscettola. Temporal dynamic controllability revisited. In *AAAI-05/IAAI-05*, pages 1193–1198, 2005.

21 Paul H. Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal uncertainty. In *IJCAI 2001*, pages 494–502, 2001.

22 Roberto Posenato. The CSTNU toolset. version 1.23. http://profs.scienze.univr.it/~posenato/software/cstnu, 2018.

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