Reducing ϵ-DC Checking for Conditional Simple Temporal Networks to DC Checking

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

Recent work on Conditional Simple Temporal Networks (CSTNs) has introduced the problem of checking the dynamic consistency (DC) property for the case where the reaction time of an execution strategy to observations is bounded below by some fixed $\epsilon > 0$, the so-called ϵ-DC-checking problem. This paper proves that the ϵ-DC-checking problem for CSTNs can be reduced to the standard DC-checking problem for CSTNs – without incurring any computational cost. Given any CSTN $\mathcal{S}$ with k observation time-points, the paper defines a new CSTN $\mathcal{S}_0$ that is the same as $\mathcal{S}$, except that for each observation time-point $P?$ in $\mathcal{S}$: (i) $P?$ is demoted to a non-observation time-point in $\mathcal{S}_0$; and (ii) a new observation time-point $P_0?$, constrained to occur exactly ϵ units after $P?$, is inserted into $\mathcal{S}_0$. The paper proves that $\mathcal{S}$ is ϵ-DC if and only if $\mathcal{S}_0$ is (standard) DC, and that the application of the ϵ-DC-checking constraint-propagation rules to $\mathcal{S}$ is equivalent to the application of the corresponding (standard) DC-checking constraint-propagation rules to $\mathcal{S}_0$. Two versions of these results are presented that differ only in whether a dynamic strategy for $\mathcal{S}_0$ can react *instantaneously* to observations, or only after some arbitrarily small, positive delay. Finally, the paper demonstrates empirically that building $\mathcal{S}_0$ and DC-checking it incurs no computational cost as the sizes of the instances increase.

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 Conditional Simple Temporal Networks, Dynamic Consistency, Temporal Constraints

Digital Object Identifier 10.4230/LIPIcs.TIME.2018.15

1 Overview

A Conditional Simple Temporal Network (CSTN) is a data structure for reasoning about time in domains where some constraints may apply only in certain scenarios. For example, a patient who tests positive for a certain disease may need to receive care more urgently than someone who tests negative. In different research fields CSTNs have been used to model temporal reasoning tasks, for example, in planning [18, 21] and automating business processes [8, 15, 17].

Conditions in a CSTN are represented by propositional letters whose truth values are not controlled, but instead are *observed* in real time. Just as doing a blood test generates a positive or negative result that is only learned in real time, the execution of an *observation time-point* $P?$ in a CSTN generates a truth value for its corresponding propositional letter p.

An execution strategy for a CSTN specifies when the time-points will be executed, but cannot affect which truth values are generated by observations. However, a strategy can be *dynamic* in that its decisions can react to information from past observations. A CSTN is said to be dynamically consistent (DC) if it admits a dynamic strategy that guarantees the satisfaction of all relevant constraints no matter which outcomes are observed during execution.

Different varieties of the DC property have been defined that differ in how reactive a dynamic strategy can be. Originally, Tsamardinos *et al.* [20] stipulated that a dynamic strategy could react to an observation only after some positive delay, but that the delay could be arbitrarily small. Comin *et al.* [9] defined ϵ-DC, which assumes that a strategy's reaction time is bounded below by some fixed $\epsilon > 0$. Finally, Cairo *et al.* [2] defined π-DC, which allows a strategy to react instantaneously (i.e., after zero delay).

Several approaches to DC-checking algorithms have been presented in the literature to address the different flavors of DC [20, 4, 5, 9]. However, the only approach that has been demonstrated to be practical is the one based on the propagation of labeled constraints due to Hunsberger et al. [14, 11]. Different versions of their algorithm have been used to solve the (standard) DC-checking, ϵ-DC-checking, and π-DC-checking problems.

This paper makes the following contributions. First, it proves that the ϵ-DC-checking problem for CSTNs can be reduced to the standard DC-checking problem. The proof involves transforming the original CSTN instance $\mathcal{S}$ into a related CSTN, $\mathcal{S}_0$, and then showing that $\mathcal{S}$ is ϵ-DC if and only if $\mathcal{S}_0$ is (standard) DC. Second, the paper proves that the application of the ϵ-DC-checking constraint-propagation rules to $\mathcal{S}$ is equivalent to the application of the corresponding DC-checking constraint-propagation rules to $\mathcal{S}_0$. Third, it empirically compares the performance of (1) building $\mathcal{S}_0$ and DC-checking it versus (2) ϵ-DC-checking the original instance $\mathcal{S}$. On a suite of benchmark instances from the business application domain, the results demonstrate that building $\mathcal{S}_0$ and DC-checking it incurs no computational cost as the sizes of the instances increase.

In this way, the global contribution of the paper is to show that the ϵ-DC-checking problem is not a *new* problem (as has been suggested in recent papers), but is in fact reducible to the standard DC-checking problem. The empirical evaluation simply confirms that the ϵ-DC-checking algorithm and the algorithm based on transforming to a standard DC problem have comparable performances.

2 Background

Dechter *et al.* [10] introduced Simple Temporal Networks (STNs) to facilitate reasoning about time. An STN comprises real-valued variables, called *time-points,* and binary difference constraints on those variables. Typically, an STN includes a special time-point, Z, whose value is fixed at zero. A *consistent* STN is one that has a solution as a constraint satisfaction problem.

Tsamardinos *et al.* [20] introduced CSTNs, which augment STNs to include *observation time-points (OTPs)* and their associated *propositional letters.* In a CSTN, the execution of an OTP $P?$ generates a truth value for its associated propositional letter p. In addition, each time-point can be labeled by a conjunction of propositional literals specifying the scenarios in which that time-point must be executed. And since constraints among labeled time-points may similarly be applicable only in certain scenarios, later work generalized CSTNs to also include labels on constraints [13].

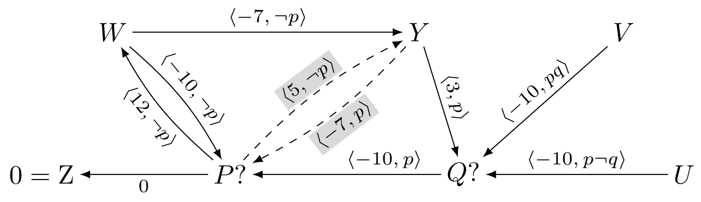

Figure 1 A sample CSTN.

Building on prior observations by Tsamardinos et al. [20], Hunsberger *et al.* [13, 6, 7] formalized properties that must be satisfied by the labels in a *well-defined* CSTN. But then Cairo *et al.* [3] showed that for any well-defined CSTN, no loss of generality results from subsequently *removing* the labels from its time-points, the result being a so-called *streamlined* CSTN. Therefore, to simplify our presentation and results, without any loss of generality, the rest of this paper restricts attention to streamlined CSTNs (i.e., CSTNs whose constraints can have propositional labels, but whose time-points cannot).

Fig. 1 shows a sample CSTN in its graphical form, where the nodes represent time-points, and the directed edges represent binary difference constraints. In the figure, Z is fixed at 0; and $P?$ and $Q?$ are OTPs whose execution generates truth values for p and q, respectively. The edge from U to $Q?$ being labeled by $p\neg q$ indicates that it applies only in scenarios where p is $\top$ and q is $\bot$. The dashed edges with shaded labels are generated by the DC-checking algorithm by Hunsberger *et al.* [14], to be discussed later on.

2.1 (Streamlined) CSTNs

The following definitions are from Hunsberger *et al.* [14], except that propositional labels appear only on constraints. Henceforth, the term CSTN shall refer to streamlined CSTNs.

▶ **Definition 1** (Labels). Given a set $\mathcal{P}$ of propositional letters: a *label* is a (possibly empty) conjunction of (positive or negative) literals from $\mathcal{P}$. The empty label is notated $\boxdot$; for any label ℓ, and any $p \in \mathcal{P}$, if $\ell \models p$ or $\ell \models \neg p$, then we say that p *appears* in ℓ; for any labels ℓ_1 and ℓ_2, if $\ell_1 \models \ell_2$, then ℓ_1 is said to *entail* ℓ_2; if $\ell_1 \wedge \ell_2$ is satisfiable, then ℓ_1 and ℓ_2 are called *consistent;* and $\mathcal{P}^*$ denotes the set of all *consistent* labels whose literals are drawn from $\mathcal{P}$.

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

- $\mathcal{T}$ is a finite set of real-valued time-points (i.e., variables);
- $\mathcal{P}$ is a finite set of propositional letters (or propositions);
- $\mathcal{C}$ is a set of *labeled* constraints, each having 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 (OTPs); and
- $\mathcal{O} \colon \mathcal{P} \to \mathcal{OT}$ is a bijection that associates a unique OTP to each propositional letter.

In a CSTN graph, the OTP for p (i.e., $\mathcal{O}(p)$) is typically denoted by $P?$; and each labeled constraint, $(Y - X \leq \delta, \ell)$, is represented by an arrow from X to Y annotated by the *labeled value,* $\langle \delta, \ell \rangle$: $X \xrightarrow{\langle \delta, \ell \rangle} Y$. Since any time-points X and Y may participate in multiple constraints of the form, $(Y - X \leq \delta_i, \ell_i)$, the edge from X to Y may have multiple labeled values of the form, $\langle \delta_i, \ell_i \rangle$.

▶ **Definition 3** (Scenario). A function, $s \colon \mathcal{P} \to \{\top, \bot\}$, that assigns a truth value to each $p \in \mathcal{P}$ is called a *scenario.* For any label $\ell \in \mathcal{P}^*$, the truth value of ℓ determined by s is denoted by $s(\ell)$. $\mathcal{I}$ denotes the set of all scenarios over $\mathcal{P}$.

▶ **Definition 4** (Schedule). A *schedule* for a set of time-points $\mathcal{T}$ is a mapping, $\psi\colon \mathcal{T} \to \mathbb{R}$. The set of all schedules over $\mathcal{T}$ is denoted by Ψ.

The projection of a CSTN onto a scenario, s, is the STN obtained by restricting attention to the constraints whose labels are true under s (i.e., that must be satisfied in that scenario).

▶ **Definition 5** (Projection). Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O}\rangle$ be any CSTN, and s any scenario over $\mathcal{P}$. The *projection* of $\mathcal{S}$ onto s – notated $\mathcal{S}(s)$ – is the STN, $(\mathcal{T}, \mathcal{C}_s^+)$, where:

$$\mathcal{C}_s^+ = \{(Y - X \leq \delta) \mid \exists \ell, (Y - X \leq \delta, \ell) \in \mathcal{C} \text{ and } s(\ell) = \top\}$$

▶ **Definition 6** (Execution Strategy). An *execution strategy* for a CSTN $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O}\rangle$ is a mapping, $\sigma\colon \mathcal{I} \to \Psi$, from scenarios to schedules. The execution time for the time-point X in the schedule $\sigma(s)$ is denoted by $[\sigma(s)]_X$. An execution strategy σ for a CSTN $\mathcal{S}$ is *viable* if for each scenario s, the schedule $\sigma(s)$ is a solution to the projection $\mathcal{S}(s)$ (i.e., σ is guaranteed to satisfy all of the relevant constraints).

3 CSTN Reduction

This paper presents two related sets of results. Each set of results involves reducing a form of ϵ-DC checking to a form of standard DC checking (one of which involves instantaneous reaction). In each case, a given CSTN $\mathcal{S}$ is transformed (or *reduced*) to a related CSTN $\mathcal{S}_0$ such that the form of ϵ-DC checking for $\mathcal{S}$ is equivalent to the corresponding form of standard DC checking for $\mathcal{S}_0$. Although there are two different versions of this result, the translation from $\mathcal{S}$ to $\mathcal{S}_0$ is the same.

▶ **Definition 7** (Reduction CSTN, $\mathcal{S}_0$). Let $\mathcal{S}$ be any CSTN, and let $\epsilon > 0$ be arbitrary. The *reduction* of $\mathcal{S}$ is the CSTN $\mathcal{S}_0$ that is the same as $\mathcal{S}$ except that for each OTP $P?$ in $\mathcal{S}$, and its associated propositional letter p:

- $P?$ is demoted from an OTP in $\mathcal{S}$ to an ordinary time-point in $\mathcal{S}_0$;[1]
- $\mathcal{S}_0$ contains a new OTP $P_0?$ that is associated with the letter p in $\mathcal{S}_0$; and
- the constraint, $(P_0? = P? + \epsilon, \boxdot)$, is contained in $\mathcal{S}_0$.[2]

More formally, $\mathcal{S}_0 = \langle \mathcal{T} \cup \mathcal{OT}_0, \mathcal{P}, \mathcal{C} \cup \mathcal{C}_0, \mathcal{OT}_0, \mathcal{O}_0\rangle$, where:

$$\begin{array}{lcl}
\mathcal{OT}_0 & = & \{P_0? \mid P? \in \mathcal{OT}\}; \\
\mathcal{O}_0(p) = P_0? & \Leftrightarrow & \mathcal{O}(p) = P?; \\
\mathcal{C}_0 & = & \{(P_0? = P? + \epsilon, \boxdot) \mid P_0? \in \mathcal{OT}_0\}.
\end{array}$$

Fig. 2 shows the reduction $\mathcal{S}_0$ that corresponds to the CSTN $\mathcal{S}$ from Fig. 1. Note that in any given instance, the value of ϵ will be fixed and known (e.g., $\epsilon = 3$).

3.1 Computational Complexity

Let $\epsilon > 0$ be arbitrary, and let $\mathcal{S}$ be any CSTN with k OTPs. The reduction of $\mathcal{S}$ to $\mathcal{S}_0$ involves $O(k)$ elementary operations for:

1. demoting original OTPs to non-OTPs;

[1] Although the demoted time-point is not an OTP in $\mathcal{S}_0$, it shall still be notated as $P?$ because keeping its name the same will simplify subsequent definitions and proofs.

[2] The constraint, $P_0? = P? + \epsilon$, abbreviates the pair of constraints, $P_0? - P? \leq \epsilon$ and $P? - P_0? \leq -\epsilon$.

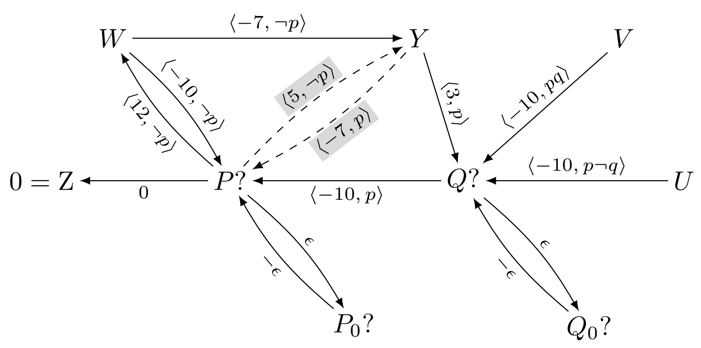

Figure 2 The reduction $\mathcal{S}_0$ corresponding to $\mathcal{S}$ from Fig. 1.

2. adding k new OTPs; and
3. inserting a pair of constraints between each demoted time-point $P?$ and the corresponding new OTP $P_0?$.

Therefore, the overall computational complexity for the reduction is $O(k)$.

4 Dynamic Strategies/Dynamic Consistency

The truth values of propositions in a CSTN are not known in advance, but a *dynamic* execution strategy can *react* to observations in real time. This paper addresses the following four flavors of dynamic strategy that differ in how reactive the strategy can be, ordered from most reactive to least reactive.

Type	Reaction Time, ρ
π-dynamic	$\rho \geq 0$
dynamic	$\rho > 0$
ϵ-dynamic	$\rho \geq \epsilon > 0$
$\hat{\epsilon}$-dynamic	$\rho > \epsilon > 0$

A π-dynamic strategy can react instantaneously to observations, but must specify an order of dependence among simultaneous observations [2]. The reaction times for a (standard) dynamic strategy can be arbitrarily small, but must be positive [20]. The reaction times for an ϵ-dynamic strategy must be greater than or equal to some fixed $\epsilon > 0$ [9]. The reaction times for an $\hat{\epsilon}$-dynamic strategy must be greater than some fixed $\epsilon > 0$. Since a CSTN is DC if and only if it has a viable and dynamic execution strategy, each distinct version of dynamic strategy gives rise to a distinct version of dynamic consistency: DC, π-DC, ϵ-DC, and $\hat{\epsilon}$-DC, respectively.

The paper will show that the $\hat{\epsilon}$-DC-checking problem can be reduced to (standard) DC checking; and that the ϵ-DC-checking problem can be reduced to π-DC checking. These reductions can be used to simplify the automated management of DC checking for CSTNs.

5 Reducing $\hat{\epsilon}$-DC Checking to (Standard) DC Checking

The following sections recall the relevant definitions for DC and $\hat{\epsilon}$-DC, and show that $\mathcal{S}$ is $\hat{\epsilon}$-DC if and only if $\mathcal{S}_0$ is DC.

5.1 Dynamic Strategies and (Standard) DC

(Standard) dynamic strategies were first defined by Tsamardinos *et al.* [20]. To facilitate comparisons with later definitions, the following are in the form given by Hunsberger *et al.* [14].

To begin, a *history* at time t comprises the truth values of all propositions that were observed *before* time t.

▶ **Definition 8** (History). Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ be any CSTN, s any scenario, σ any execution strategy for $\mathcal{S}$, and t any real number. The *history* of t in the scenario s, for the strategy σ – notated $\mathit{Hist}(t, s, \sigma)$ – is the set of observations made before time t according to the schedule $\sigma(s)$:

$$\mathit{Hist}(t, s, \sigma) = \{(p, s(p)) \mid P? \in \mathcal{OT} \text{ and } [\sigma(s)]_{P?} < t\}$$

▶ **Definition 9** (Dynamic Strategy). An execution strategy σ for a CSTN $\mathcal{S}$ is called *dynamic* if for any scenarios s_1 and s_2, and any time-point X:

let: $t = [\sigma(s_1)]_X$
if: $\mathit{Hist}(t, s_1, \sigma) = \mathit{Hist}(t, s_2, \sigma)$
then: $[\sigma(s_2)]_X = t$.

In other words, if a dynamic strategy σ executes X at time t in scenario s_1, and the schedules $\sigma(s_1)$ and $\sigma(s_2)$ have the same history of *past* observations, then σ must also execute X at time t in s_2. That is, execution decisions can only depend on *past* observations, even if arbitrarily recent.

▶ **Definition 10** (Dynamic Consistency (DC)). A CSTN is *dynamically consistent* (DC) if there exists an execution strategy for it that is both viable and dynamic.

5.2 $\hat{\epsilon}$-Dynamic Strategies and $\hat{\epsilon}$-DC

The $\hat{\epsilon}$-dynamic consistency property has not been presented before in the literature. However, it differs only slightly from ϵ-DC, defined later on. Informally, an $\hat{\epsilon}$-dynamic strategy must schedule a time-point X at the same time t in two different scenarios s_1 and s_2 if both scenarios have the same history of *past* observations *before* time $t - \epsilon$ (i.e., ϵ units before the current time t).

▶ **Definition 11** ($\hat{\epsilon}$-History). Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ be any CSTN, s any scenario, σ any execution strategy for $\mathcal{S}$, t any real number, and $\epsilon > 0$. The *$\hat{\epsilon}$-history* of t in the scenario s, for the strategy σ, notated $\hat{\epsilon}\mathit{Hist}(t, s, \sigma)$, is the set of observations made *before* time $t - \epsilon$ according to $\sigma(s)$:

$$\hat{\epsilon}\mathit{Hist}(t, s, \sigma) = \{(p, s(p)) \mid P? \in \mathcal{OT} \text{ and } [\sigma(s)]_{P?} < t - \epsilon\}$$

▶ **Definition 12** ($\hat{\epsilon}$-Dynamic Execution Strategy). Let $\epsilon > 0$; and let $\mathcal{S}$ be any CSTN. An execution strategy σ for $\mathcal{S}$ is called *$\hat{\epsilon}$-dynamic* if for any scenarios s_1 and s_2, and any time-point X:

let: $t = [\sigma(s_1)]_X$
if: $\hat{\epsilon}\mathit{Hist}(t, s_1, \sigma) = \hat{\epsilon}\mathit{Hist}(t, s_2, \sigma)$
then: $[\sigma(s_2)]_X = t$.

▶ **Definition 13** ($\hat{\epsilon}$-DC)**.** For any $\epsilon > 0$, a CSTN $\mathcal{S}$ is *$\hat{\epsilon}$-dynamically consistent* if it has a viable and $\hat{\epsilon}$-dynamic execution strategy.

The following theorem explicates the relation between $\hat{\epsilon}$-DC and DC.

▶ **Theorem 14.** *Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ be any CSTN; let $\epsilon > 0$ be arbitrary; and let $\mathcal{S}_0 = \langle \mathcal{T} \cup \mathcal{OT}_0, \mathcal{P}, \mathcal{C} \cup \mathcal{C}_0, \mathcal{OT}_0, \mathcal{O}_0 \rangle$ be the reduction of $\mathcal{S}$. Then $\mathcal{S}$ is $\hat{\epsilon}$-DC if and only if $\mathcal{S}_0$ is DC.*

Proof. ($\Rightarrow$) Suppose that $\mathcal{S}$ is $\hat{\epsilon}$-DC. Then there exists an $\hat{\epsilon}$-dynamic and viable strategy σ for $\mathcal{S}$. Define a strategy σ_0 for the reduction $\mathcal{S}_0$, as follows. For any scenario s:

(1) For each $X \in \mathcal{T}$:	let $[\sigma_0(s)]_X$	$= [\sigma(s)]_X$
(1) For each $P_0? \in \mathcal{OT}_0$:	let $[\sigma_0(s)]_{P_0?}$	$= [\sigma(s)]_{P?} + \epsilon$

Since σ is viable for $\mathcal{S}$, σ satisfies all of the constraints in $\mathcal{C}$. And since, by (1) above, σ and σ_0 agree on all of the time-points in $\mathcal{T}$, σ_0 must also satisfy all of the constraints in $\mathcal{C}$. Finally, by (2) above, σ_0 must satisfy all of the constraints in $\mathcal{C}_0$. Therefore, σ_0 must be viable for $\mathcal{S}_0$.

Next, suppose that σ_0 is not dynamic. Then for some scenarios s_1 and s_2, and some time-point $X \in \mathcal{T} \cup \mathcal{T}_0$, $\mathit{Hist}(t, s_1, \sigma_0) = \mathit{Hist}(t, s_2, \sigma_0)$, but $[\sigma_0(s_2)]_X \neq t$, where $t = [\sigma_0(s_1)]_X$. With no loss of generality, assume that t is minimal for this circumstance. Then:

$$\begin{aligned}
\hat{\epsilon}\mathit{Hist}(t, s_2, \sigma) &= \{(p, s_2(p)) \mid P? \in \mathcal{OT} \text{ and } [\sigma(s_2)]_{P?} < t - \epsilon\} \\
&= \{(p, s_2(p)) \mid P_0? \in \mathcal{OT}_0 \text{ and } [\sigma_0(s_2)]_{P_0?} < t\}, \text{ by (2) above} \\
&= \mathit{Hist}(t, s_2, \sigma_0) \\
&= \mathit{Hist}(t, s_1, \sigma_0) \\
&= \{(p, s_1(p)) \mid P_0? \in \mathcal{OT}_0 \text{ and } [\sigma_0(s_1)]_{P_0?} < t\} \\
&= \{(p, s_1(p)) \mid P? \in \mathcal{OT} \text{ and } [\sigma(s_1)]_{P?} < t - \epsilon\}, \text{ by (2) above} \\
&= \hat{\epsilon}\mathit{Hist}(t, s_1, \sigma). \qquad (\dagger)
\end{aligned}$$

Now, if $X \in \mathcal{T}$, then $[\sigma(s_1)]_X = [\sigma_0(s_1)]_X = t$, but $[\sigma(s_2)]_X = [\sigma_0(s_2)]_X \neq t$, which, given that the relevant $\hat{\epsilon}$-histories are equal, contradicts that σ is $\hat{\epsilon}$-dynamic. Therefore, $X \notin \mathcal{T}$; thus, X must be some $R_0? \in \mathcal{OT}_0$, where $[\sigma(s_1)]_{R?} = [\sigma_0(s_1)]_{R_0?} - \epsilon = t - \epsilon \neq [\sigma_0(s_2)]_{R_0?} - \epsilon = [\sigma(s_2)]_{R?}$. Thus, the schedules $\sigma(s_1)$ and $\sigma(s_2)$ are different. Consider those schedules, each annotated with the relevant observations as they occur. Let $t^* \leq t - \epsilon$ be the earliest time at which the annotated schedules differ. There are two ways they can differ at t^*.

Case 1. Both schedules execute some OTP $Q?$ at t^*, but with different results (i.e., $s_1(q) \neq s_2(q)$). If $t^* < t - \epsilon$, it would contradict that $\hat{\epsilon}\mathit{Hist}(t, s_1, \sigma) = \hat{\epsilon}\mathit{Hist}(t, s_2, \sigma)$. Therefore, $t^* = t - \epsilon$.
Now, the definition of t^* ensures that $\hat{\epsilon}\mathit{Hist}(t^*, s_1, \sigma) = \hat{\epsilon}\mathit{Hist}(t^*, s_2, \sigma)$. But then $[\sigma(s_1)]_{R?} \neq [\sigma(s_2)]_{R?}$, shown earlier, contradicts that σ is $\hat{\epsilon}$-dynamic.

Case 2. One of the schedules executes some time-point Y at t^* while the other schedule executes Y at some later time: $[\sigma(s_i)]_Y = t^* < [\sigma(s_j)]_Y$, where $\{s_i, s_j\} = \{s_1, s_2\}$. But this, together with $\hat{\epsilon}\mathit{Hist}(t^*, s_1, \sigma) = \hat{\epsilon}\mathit{Hist}(t^*, s_2, \sigma)$, contradicts that σ is $\hat{\epsilon}$-dynamic.

($\Leftarrow$) Suppose that $\mathcal{S}_0$ is DC. Then there exists a viable and dynamic strategy σ_0 for $\mathcal{S}_0$. Let σ be the strategy for $\mathcal{S}$ such that for each scenario s, and each time-point $X \in \mathcal{T}$, $[\sigma(s)]_X = [\sigma_0(s)]_X$. Since σ_0 is viable for $\mathcal{S}_0$, it satisfies all of the constraints in $\mathcal{C} \cup \mathcal{C}_0$. And since σ and σ_0 agree on all time-points in $\mathcal{T}$, it follows that σ satisfies all of the constraints in $\mathcal{C}$. Thus, σ is viable for $\mathcal{S}$.

Next, suppose that σ is not $\hat{\epsilon}$-dynamic. Then for some scenarios s_1 and s_2, and some time-point $X \in \mathcal{T}$, $\hat{\epsilon}Hist(t, s_1, \sigma) = \hat{\epsilon}Hist(t, s_2, \sigma)$, where $t = [\sigma(s_1)]_X$, but $[\sigma(s_2)]_X \neq t$. Arguing similarly to (†) above, it follows that $Hist(t, s_1, \sigma_0) = Hist(t, s_2, \sigma_0)$. And since $X \in \mathcal{T}$, $[\sigma_0(s_1)]_X = [\sigma(s_1)]_X = t \neq [\sigma(s_2)]_X = [\sigma_0(s_2)]_X$, contradicting that σ_0 is dynamic. ◀

6 Reducing ϵ-DC Checking to π-DC Checking

This section uses the same reduction of $\mathcal{S}$ to $\mathcal{S}_0$ to reduce the problem of ϵ-DC checking to that of π-DC checking.

6.1 ϵ-Dynamic Execution Strategy and ϵ-DC

The semantics for ϵ-DC is the same as that for $\hat{\epsilon}$-DC, except that an ϵ-history records the observations *at or before* time $t - \epsilon$, instead of *strictly before* $t - \epsilon$. Nonetheless, for easy reference, the full definitions are given below.

▶ **Definition 15** (ϵ-History). Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ be any CSTN, s any scenario, σ any execution strategy for $\mathcal{S}$, t any real number, and $\epsilon > 0$. The *ϵ-history* of t in the scenario s, for the strategy σ, notated $\epsilon Hist(t, s, \sigma)$, is the set of observations made *at or before* $t - \epsilon$ according to $\sigma(s)$:

$$\epsilon Hist(t, s, \sigma) = \{(p, s(p)) \mid P? \in \mathcal{OT} \text{ and } [\sigma(s)]_{P?} \leq t - \epsilon\}$$

▶ **Definition 16** (ϵ-Dynamic Execution Strategy). Let $\epsilon > 0$. An execution strategy σ is *ϵ-dynamic* if for any scenarios s_1 and s_2, and any time-point X:

let: $t = [\sigma(s_1)]_X$
if: $\epsilon Hist(t, s_1, \sigma) = \epsilon Hist(t, s_2, \sigma)$
then: $[\sigma(s_2)]_X = t$.

▶ **Definition 17** (ϵ-DC). Given any $\epsilon > 0$, a CSTN is *ϵ-dynamically consistent* (ϵ-DC) if there exists an execution strategy for it that is both viable and ϵ-dynamic.

6.2 π-Dynamic Execution Strategy and π-DC

This section summarizes the π-DC semantics introduced by Cairo *et al.* [2] that allows a dynamic strategy to react instantaneously to observations, but requires an order of dependence among simultaneous observations.

▶ **Definition 18** (Order of dependence). For any ordering $(P_1?, \ldots, P_k?)$ of observation time-points, where $k = |\mathcal{OT}|$, an *order of dependence* is a permutation π over $(1, 2, \ldots, k)$; and for each $P? \in \mathcal{OT}$, $\pi(P?) \in \{1, 2, ..., k\}$ denotes the integer position of $P?$ in that order. For any *non*-observation time-point X, we set $\pi(X) = \infty$. Finally, Π_k denotes the set of all permutations over $(1, 2, \ldots, k)$.

▶ **Definition 19** (π-Execution Strategy). For any CSTN $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$, where $k = |\mathcal{OT}|$, a *π-execution strategy* for $\mathcal{S}$ is a mapping, $\sigma\colon \mathcal{I} \to (\Psi \times \Pi_k)$, such that for each scenario s, $\sigma(s)$ is a pair (ψ, π) where $\psi\colon \mathcal{T} \to \mathbb{R}$ is a schedule and $\pi \in \Pi_k$ is an order of dependence. For any $X \in \mathcal{T}$, $[\sigma(s)]_X$ denotes the execution time of X (i.e., $\psi(X)$); and for any $P? \in \mathcal{OT}$, $[\sigma(s)]^{\pi}_{P?}$ denotes the position of $P?$ in the order of dependence (i.e., $\pi(P?)$). Finally, a π-dynamic strategy must be *coherent:* for any scenario s, and any $P?, Q? \in \mathcal{OT}$, $[\sigma(s)]_{P?} < [\sigma(s)]_{Q?}$ implies $[\sigma(s)]^{\pi}_{P?} < [\sigma(s)]^{\pi}_{Q?}$ (i.e., if $\sigma(s)$ *schedules* $P?$ before $Q?$, then it *orders* $P?$ before $Q?$).

▶ **Definition 20** (Viability)**.** The π-execution strategy $\sigma = (\psi, \pi)$ is called *viable* for the CSTN $\mathcal{S}$ if for each scenario s, the schedule $\psi(s)$ is a solution to the projection $\mathcal{S}(s)$.

▶ **Definition 21** (π-History)**.** Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ be any CSTN. Let σ be any π-execution strategy for $\mathcal{S}$, s any scenario, t any real number, and $d \in \{1, 2, \ldots, |\mathcal{OT}|\} \cup \{\infty\}$ any integer position (or infinity). The *π-history* of (t, d) for the scenario s and strategy σ – denoted by $\pi Hist(t, d, s, \sigma)$ – is the set

$$\{(p, s(p)) \mid P? \in \mathcal{OT}, [\sigma(s)]_{P?} \le t, \text{ and } \pi(P?) < d\}.$$

Thus, the π-history specifies the truth values of each proposition p that is observed *before* time t in the schedule ψ, or observed *at* time t if its corresponding observation time-point $P?$ is ordered *before* position d by the permutation π.

The following definition of a π-dynamic strategy is equivalent to that given by Cairo *et al.* [2]. (The straightforward proof has been omitted to save space.)

▶ **Definition 22** (π-Dynamic Strategy)**.** A π-execution strategy, σ, for a CSTN is *π-dynamic* if for every pair of scenarios, s_1 and s_2, and every time-point $X \in \mathcal{T}$:

$$\begin{array}{ll} \text{let:} & t = [\sigma(s_1)]_X, \text{ and } d = [\sigma(s_1)]^{\pi}_X. \\ \text{if:} & \pi Hist(t, d, s_1, \sigma) = \pi Hist(t, d, s_2, \sigma) \\ \text{then:} & [\sigma(s_2)]_X = t \text{ and } [\sigma(s_2)]^{\pi}_X = d. \end{array}$$

Thus, if σ executes X at time t and position d in scenario s_1, and the histories, $\pi Hist(t, d, s_1, \sigma)$ and $\pi Hist(t, d, s_2, \sigma)$, are the same, then σ must also execute X at time t and in position d in scenario s_2. (X may be an observation time-point.)

▶ **Definition 23** (π-Dynamic Consistency)**.** A CSTN, $\mathcal{S}$, is *π-dynamically consistent* (π-DC) if there exists a π-execution strategy for $\mathcal{S}$ that is both viable and π-dynamic.

Theorem 24 explicates the relationship between the ϵ-DC and π-DC properties. Its proof, which uses techniques similar to those used to prove Theorem 14, extended to accommodate the order of dependence, is omitted to save space.

▶ **Theorem 24.** *Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ be any CSTN; let $\epsilon > 0$ be arbitrary; and let $\mathcal{S}_0 = \langle \mathcal{T}_0, \mathcal{P}, \mathcal{C}_0, \mathcal{OT}_0, \mathcal{O}_0 \rangle$ be the reduction of $\mathcal{S}$. Then $\mathcal{S}$ is ϵ-DC if and only if $\mathcal{S}_0$ is π-DC.*

7 The π-DC- and ϵ-DC-Checking Algorithms

This section summarizes two versions of the constraint-propagation algorithm due to Hunsberger *et al.* [14][12]: the π-DC-checking algorithm and the ϵ-DC-checking algorithm. Each algorithm uses only three constraint-propagation rules.[3] This section proves that applying the rules for the ϵ-DC-checking algorithm to the CSTN $\mathcal{S}$ is equivalent to applying the rules for the π-DC-checking algorithm to the corresponding reduced CSTN $\mathcal{S}_0$.

[3] An earlier version of the π-DC-checking algorithm, called the IR-DC-checking algorithm (IR for “instantaneous reaction”) used six rules, but recently, Hunsberger and Posenato [12] showed that three rules are sufficient. We applied similar techniques to create a three-rule version of the ϵ-DC-checking algorithm.

Table 1 Propagation rules for the π-DC-checking algorithm (above) and instances of their use (below).

LP:	$\begin{tikzcd} X \arrow[r, "\langle u,\alpha\rangle"] \arrow[rr, bend right, "\langle u+v,\alpha\beta\rangle"'] & Y \arrow[r, "\langle v,\beta\rangle"] & Z \end{tikzcd}$	if $\alpha\beta \in \mathcal{P}^*$ and $u+v<0$
qR_0:	$\begin{tikzcd} P? \arrow[r, "\langle w,\alpha\tilde{p}\rangle", "\langle w,\alpha\rangle"'] & Z \end{tikzcd}$	if $w<0$, $\tilde{p} \in \{p,\neg p,?p\}$, and $\alpha \in \mathcal{Q}^*$
qR_3^*:	$\begin{tikzcd} P? \arrow[r, "\langle w,\alpha\rangle"] & Z & Y \arrow[l, "\langle v,\beta\tilde{p}\rangle"', "\langle m,\alpha\star\beta\rangle"] \end{tikzcd}$	if $w<0$, $\tilde{p} \in \{p,\neg p,?p\}$, and $\alpha,\beta \in \mathcal{Q}^*$
$X,Y \in \mathcal{T}$; $P? \in \mathcal{OT}$; $\mathrm{Z}=0$. In $\mathrm{qR}_0/\mathrm{qR}_3^*$, p does not appear in α or β; and $m=\max\{v,w\}$.		

LP:	$\begin{tikzcd} X \arrow[r, "\langle -3,pqr\rangle"] \arrow[rr, bend right, "\langle -7,pqrs\neg t\rangle"'] & Y \arrow[r, "\langle -4,rs\neg t\rangle"] & Z \end{tikzcd}$
qR_0:	$\begin{tikzcd} P? \arrow[r, "\langle -9,(?p)qr\rangle", "\langle -9,qr\rangle"'] & Z \end{tikzcd}$
qR_3^*:	$\begin{tikzcd} A? \arrow[r, "\langle -1,b\neg c\rangle"] & Z & B? \arrow[l, "\langle -1,ac\rangle"', "\langle -1,b(?c)\rangle"] \end{tikzcd}$

7.1 The π-DC-Checking Algorithm

Table 1 lists the three constraint propagation rules used by the π-DC-checking algorithm. Note that the qR_3^* rule can generate a new kind of propositional label, called a *q-label* (defined below); and the qR_0 and qR_3^* rules can each be applied to q-labeled edges. Each q-label is a conjunction of *q-literals* (defined below). Whereas a constraint labeled by p must hold in all scenarios in which p is true, a constraint labeled by the q-literal $?p$ need only hold as long as the truth value of p is unknown (i.e., as long as $P?$ has not been executed).

▶ **Definition 25** (Q-literals, q-labels). A *q-literal* is a literal of the form $?p$, where $p \in \mathcal{P}$. A *q-label* is a conjunction of literals and/or q-literals. $\mathcal{Q}^*$ denotes the set of all q-labels. For any scenario s, and any q-literal $?p$, it is convenient to stipulate that $s \not\models ?p$.

For example, $p(?q)\neg r$ and $(?p)(?q)(?r)$ are both q-labels.

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

▶ **Definition 26** ($\star$). The operator, $\star\colon \mathcal{Q}^* \times \mathcal{Q}^* \to \mathcal{Q}^\star$, is defined thusly. First, for any $p \in \mathcal{P}$, $p \star p = p$ and $\neg p \star \neg p = \neg p$; and for any $p_1, p_2 \in \{p, \neg p, ?p\}$, such that $p_1 \neq p_2$, $p_1 \star p_2 = ?p$. 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: $(p\neg q(?r)t) \star (qr\neg s) = p(?q)(?r)\neg st$.

7.2 The ϵ-DC-checking Algorithm

The ϵ-DC-checking algorithm uses the same rules as the π-DC-checking algorithm, except that in the qR_3^* rule, it uses $m = \max\{v, w-\epsilon\}$. For clarity, we shall refer to this version of the qR_3^* rule as qR_3^ϵ; and $\{\mathrm{LP}, \mathrm{qR}_0, \mathrm{qR}_3^\epsilon\}$ shall be called the ϵ-DC-checking rules.

▶ **Theorem 27.** *Let $\epsilon > 0$; let $\mathcal{S}$ be any CSTN; and let $\mathcal{S}^*$ be the CSTN that results from exhaustively applying the ϵ-DC-checking constraint-propagation rules to $\mathcal{S}$. Let $\mathcal{S}_0$ be the corresponding reduced CSTN for $\mathcal{S}$; and let $\mathcal{S}_0^*$ be the CSTN that results from exhaustively applying the π-DC-checking rules to $\mathcal{S}_0$. Then $\mathcal{S}^*$ and $\mathcal{S}_0^*$ are equivalent in the following sense:*

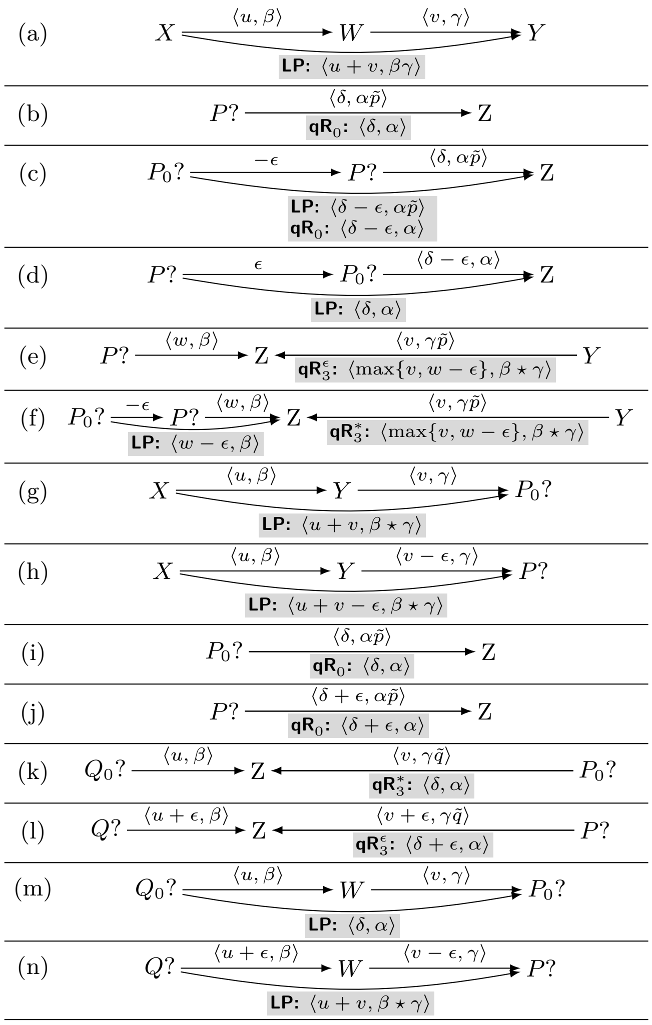
(a) $X \xrightarrow{\langle u,\beta\rangle} W \xrightarrow{\langle v,\gamma\rangle} Y$; LP: $\langle u+v, \beta\gamma\rangle$
(b) $P? \xrightarrow{\langle \delta, \alpha\tilde{p}\rangle} Z$; qR$_0$: $\langle \delta, \alpha\rangle$
(c) $P_0? \xrightarrow{-\epsilon} P? \xrightarrow{\langle \delta, \alpha\tilde{p}\rangle} Z$; LP: $\langle \delta-\epsilon, \alpha\tilde{p}\rangle$; qR$_0$: $\langle \delta-\epsilon, \alpha\rangle$
(d) $P? \xrightarrow{\epsilon} P_0? \xrightarrow{\langle \delta-\epsilon, \alpha\rangle} Z$; LP: $\langle \delta, \alpha\rangle$
(e) $P? \xrightarrow{\langle w,\beta\rangle} Z \xleftarrow{\langle v,\gamma\tilde{p}\rangle} Y$; qR$_3^\epsilon$: $\langle \max\{v, w-\epsilon\}, \beta\star\gamma\rangle$
(f) $P_0? \xrightarrow{-\epsilon} P? \xrightarrow{\langle w,\beta\rangle} Z \xleftarrow{\langle v,\gamma\tilde{p}\rangle} Y$; LP: $\langle w-\epsilon, \beta\rangle$; qR$_3^*$: $\langle \max\{v, w-\epsilon\}, \beta\star\gamma\rangle$
(g) $X \xrightarrow{\langle u,\beta\rangle} Y \xrightarrow{\langle v,\gamma\rangle} P_0?$; LP: $\langle u+v, \beta\star\gamma\rangle$
(h) $X \xrightarrow{\langle u,\beta\rangle} Y \xrightarrow{\langle v-\epsilon,\gamma\rangle} P?$; LP: $\langle u+v-\epsilon, \beta\star\gamma\rangle$
(i) $P_0? \xrightarrow{\langle \delta, \alpha\tilde{p}\rangle} Z$; qR$_0$: $\langle \delta, \alpha\rangle$
(j) $P? \xrightarrow{\langle \delta+\epsilon, \alpha\tilde{p}\rangle} Z$; qR$_0$: $\langle \delta+\epsilon, \alpha\rangle$
(k) $Q_0? \xrightarrow{\langle u,\beta\rangle} Z \xleftarrow{\langle v,\gamma\tilde{q}\rangle} P_0?$; qR$_3^*$: $\langle \delta, \alpha\rangle$
(l) $Q? \xrightarrow{\langle u+\epsilon,\beta\rangle} Z \xleftarrow{\langle v+\epsilon,\gamma\tilde{q}\rangle} P?$; qR$_3^\epsilon$: $\langle \delta+\epsilon, \alpha\rangle$
(m) $Q_0? \xrightarrow{\langle u,\beta\rangle} W \xrightarrow{\langle v,\gamma\rangle} P_0?$; LP: $\langle \delta, \alpha\rangle$
(n) $Q? \xrightarrow{\langle u+\epsilon,\beta\rangle} W \xrightarrow{\langle v-\epsilon,\gamma\rangle} P?$; LP: $\langle u+v, \beta\star\gamma\rangle$

Figure 3 Constraint propagations for the proof of Theorem 27.

(1) *Every constraint in $\mathcal{S}^*$ is also in $\mathcal{S}_0^*$.*

(2) *For each $P_0?, Q_0? \in \mathcal{OT}_0$, $X \in \mathcal{T}\backslash\mathcal{OT}_0$, $\delta \in \mathbb{R}$, and $\alpha \in \mathcal{Q}^*$:*

(a) $(P_0?-X \leq \delta, \alpha) \in \mathcal{S}_0^* \;\Rightarrow (P? - X \leq \delta-\epsilon, \alpha) \in \mathcal{S}^*$
(b) $(X-P_0? \leq \delta, \alpha) \in \mathcal{S}_0^* \;\Rightarrow (X - P? \leq \delta+\epsilon, \alpha) \in \mathcal{S}^*$
(c) $(P_0?-Q_0? \leq \delta, \alpha) \in \mathcal{S}_0^* \Rightarrow (P? - Q? \leq \delta, \alpha) \in \mathcal{S}^*$

Proof. (Part 1) Let Σ be some arbitrary sequence of applications of $\hat{\epsilon}$-DC-checking rules to edges from $\mathcal{S}^*$. Let $(Y - X \leq \delta, \alpha)$ in $\mathcal{S}^*$ be the *first* edge generated by that sequence that does *not* belong to $\mathcal{S}_0^*$. Call that constraint C. (Note that X and Y must be in $\mathcal{T}$ since C is in $\mathcal{S}^*$.)

Case 1.1: The constraint C was generated by applying the LP rule to edges in $\mathcal{S}^*$ (e.g., as shown in Fig. 3a, where $\delta = u+v$ and $\alpha = \beta\gamma$). But then, by assumption, the pre-existing edges (from X to W to Y) must also be in $\mathcal{S}_0^*$. Since the LP rule is also one of the DC-checking rules, the generated edge (from X to Y) must also be in $\mathcal{S}_0^*$. But that edge represents the constraint C, a contradiction.

Case 1.2: The constraint C was generated by the qR_0 rule (e.g., as shown in Fig. 3b). But then the pre-existing edge from $P?$ to Z must be in $\mathcal{S}_0^*$; and so is the edge from $P_0?$ to $P?$, as shown in Fig. 3c. Applying the LP rule to these edges, followed by the qR_0 rule, then yields the shaded labeled values for the edge from $P_0?$ to Z in $\mathcal{S}_0^*$, as shown in Fig. 3c. Next, applying the LP rule to the edges from $P?$ to $P_0?$ to Z, as shown in Fig. 3d, generates the edge from $P?$ to Z for $\mathcal{S}_0^*$. But that edge represents the constraint C, a contradiction.

Case 1.3: The constraint C was generated by the qR_3^ϵ rule (e.g., as shown in Fig. 3e, where $m = \max\{v, w - \epsilon\}$ and $\alpha = \beta \star \gamma$). But then the pre-existing edges (from $P_0?$ to $P?$ to Z) must be in $\mathcal{S}_0^*$, which allows the LP rule to generate the edge from $P_0?$ to Z in $\mathcal{S}_0^*$, shown in Fig. 3f, and then the qR_3^* rule to generate the shaded value for the edge from Y to Z in $\mathcal{S}_0^*$, which represents the constraint C, a contradiction.

(Part 2) Let Σ_0 be some arbitrary sequence of recursive applications of DC-checking rules to edges from $\mathcal{S}_0^*$. Let K in $\mathcal{S}_0^*$ be the *first* edge generated by that sequence that does *not* have a corresponding edge in $\mathcal{S}^*$ according to (2a), (2b) and (2c) in the statement of the theorem.

Case 2a: K has the form $(P_0? - X \leq \delta, \alpha)$. Given that $P_0? \not\equiv$ Z, K can only have been generated by an application of the LP rule to edges in $\mathcal{S}_0^*$, as shown in Fig. 3g, where $\delta = u + v$ and $\alpha = \beta \star \gamma$. By assumption, the pre-existing edge from Y to $P_0?$ in $\mathcal{S}_0^*$ must have a corresponding edge from Y to $P?$ in $\mathcal{S}^*$. For example, if $Y \in \mathcal{T}$, then by (2a) there must be an edge from Y to $P?$ as shown in Fig. 3h. That edge enables the LP rule to then be used to generate the edge from X to $P?$ in $\mathcal{S}^*$, also shown in Fig. 3h. But that is the edge in $\mathcal{S}^*$ that corresponds to K, a contradiction. The case where Y is some $Q_0? \in \mathcal{OT}_0$ is even easier.

Case 2b.1: K has the form $(X - P_0? \leq \delta, \alpha)$ and was generated by applying the LP rule to edges in $\mathcal{S}_0^*$. This case is similar to Case 2a, but focusing on the lefthand side of the LP rule (i.e., the edge from X to Y) instead of the right.

Case 2b.2: K has the form $(\mathrm{Z} - P_0? \leq \delta, \alpha)$ and was generated by applying the qR_0 rule to edges in $\mathcal{S}_0^*$, as shown in Fig. 3i. But then, by (2b), the pre-existing edge from $P_0?$ to Z has a corresponding edge in $\mathcal{S}^*$ from $P?$ to Z, leading to the propagation in Fig. 3j, which generates the edge in $\mathcal{S}^*$ that corresponds to K, a contradiction.

Case 2b.3: K has the form $(\mathrm{Z} - P_0? \leq \delta, \alpha)$ and was generated by applying the qR_3^* rule to edges in $\mathcal{S}_0^*$, as shown in Fig. 3k, where $\delta = \max\{u, v\}$ and $\alpha = \beta \star \gamma$. By (2b), the pre-existing edges from $Q_0?$ to Z, and from $P_0?$ to Z in $\mathcal{S}_0$ have corresponding edges from $Q?$ to Z, and from $P?$ to Z in $\mathcal{S}^*$, as shown in Fig. 3l, leading to the generated edge from $P?$ to Z, where $\delta + \epsilon = \max\{u + \epsilon, v + \epsilon\}$. That edge corresponds to K, a contradiction.

Case 3: K has the form $(P_0? - Q_0? \leq \delta, \alpha)$. Since $P_0? \not\equiv$ Z, then K must have been generated by an application of the LP rule. Fig. 3m illustrates the case where the intermediate time-point W is not in $\mathcal{OT}_0$, and where $\delta = u + v$ and $\alpha = \beta\gamma$. By (2b) and (2a), respectively, the pre-existing edges from $Q_0?$ to W to $P_0?$ have corresponding edges from $Q?$ to W to $P?$ in $\mathcal{S}^*$, leading to the propagation in Fig. 3n, where the edge from $Q?$ to $P?$ corresponds to K, a contradiction. The case where $W \in \mathcal{OT}_0$ is handled similarly. ◀

Theorem 27 shows that the ϵ-DC-checking and π-DC-checking algorithms perform equivalent constraint propagations. However, providing an analogous theorem that relates $\hat{\epsilon}$-DC-checking and standard DC-checking algorithms must be left to future work, since (1) no algorithm has yet been introduced for solving the $\hat{\epsilon}$-DC-checking problem; and (2) the sound 6-rule DC-checking algorithm for the standard DC-checking problem has not yet been proven complete [11].

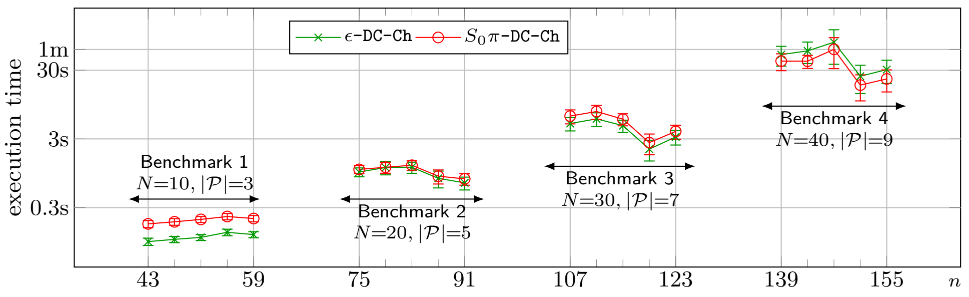

Figure 4 Execution time vs. number of time-points n.

8 Empirical Evaluation

Theorem 24 shows that the ϵ-DC-checking problem can be reduced to the π-DC-checking problem. In particular, the ϵ-DC-checking problem for a CSTN $\mathcal{S}$ is equivalent to the π-DC-checking problem for the reduced CSTN $\mathcal{S}_0$. This section compares the performance of implementations of an ϵ-DC-checking algorithm [11] and the π-DC-checking algorithm [12]. In what follows, the first algorithm is called ϵ-`DC-Ch` and the second is called $\mathcal{S}_0\pi$-`DC-Ch` to reflect that it is applied to the reduced CSTN $\mathcal{S}_0$. Both implementations were obtained from Posenato [19], but we modified the ϵ-DC-checking algorithm to use only three rules, as discussed in Footnote 3 and Section 7.2. Algorithms and procedures for this evaluation were implemented in Java and executed on a JVM 8 in a Linux machine with two AMD Opteron 4334 CPUs and 64GB of RAM.

To facilitate comparisons with prior work, we tested both implementations on instances of the four benchmarks proposed by Hunsberger and Posenato [11]. Each benchmark has at least 60 DC and 60 non-DC CSTNs, obtained from random workflow schemata generated by the ATAPIS toolset [16]. The numbers of activities (N) and observations ($|\mathcal{P}|$) were varied, as shown in Fig. 4. Since non-DC networks were regularly solved one to two orders of magnitude faster than similarly sized DC ones, the rest of this section focuses on DC networks.

Fig. 4 displays the average execution times of the two algorithms over all four benchmarks. For $\mathcal{S}_0\pi$-`DC-Ch`, the execution times include the time required to build $\mathcal{S}_0$. Each data point represents the average of the execution times for instances of the given size. We extended the benchmarks, adding up to 50 DC instances, to generate tight error bars. In Figure 4, each error bar represents a 95% confidence interval for the average of the execution times.

The results demonstrate that, although ϵ-`DC-Ch` performs better in Benchmark 1, the performance difference becomes statistically insignificant as the sizes of the instances increase. The main reason for this behavior is that in small instances the number of observation time-points is quite small, which results in much less constraint propagation. As a result, the linear time required to build $\mathcal{S}_0$ and do the extra propagations can have a significant impact. In contrast, in larger instances, the number of observation time-points is larger, in which case the time to build $\mathcal{S}_0$ is dwarfed by the exponential time required for propagating constraints. The benchmark does not provide any larger instances because such instances are determined from random workflow instances where the maximum instance size is commonly limited to 30-40 tasks [16].

The experiments summarized in this section show that for all but the smallest CSTNs, there is no computational penalty associated with solving the ϵ-DC-checking problem by first computing the reduced CSTN $\mathcal{S}_0$ and then applying the π-DC-checking algorithm to it.

9 Conclusions

This paper presented a reduction of the $\hat{\epsilon}$-DC-checking problem to the (standard) DC-checking problem for CSTNs, and a reduction of the ϵ-DC-checking problem to the π-DC-checking problem. It also showed that the constraint-propagation rules for the π-DC-checking algorithm are equivalent to the rules for the ϵ-DC-checking algorithm. As a result, the paper showed that the ϵ-DC-checking problem for CSTNs can be easily represented within the standard CSTN framework (i.e., the ϵ-DC-checking problem is not a "new" problem, as has been suggested in recent related work). Furthermore, solving the ϵ-DC-checking problem by applying the π-DC-checking algorithm to the reduced CSTN incurs no computational penalty.

Results of this paper can be applied also in other possible variants of CSTNs. Bhargava et al. [1] defined the *delay controllability* of a Simple Temporal Network with Uncertainty (STNU). This modification allowed a network designer to insert a delay between the execution of a contingent time-point and the time that the executing agent learns of the duration of the corresponding contingent link. And the delay associated with each contingent link can be different. Similarly, a version of delay consistency for CSTNs could be defined to allow different values of epsilon for each observation time-point. In addition, the techniques used in this paper to reduce epsilon-DC for CSTNs to ordinary DC for CSTNs could be used to reduce delay controllability for STNUs to ordinary DC for STNUs.

References

1 Nikhil Bhargava, Christian Muise, Tiago Vaquero, and Brian Williams. Delay Controllability: Multi-Agent Coordination under Communication Delay. Technical Report MIT-CSAIL-TR-2018-002, MIT, 2018. URL: `http://hdl.handle.net/1721.1/113340`.

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 - 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`.

4 A. Cimatti, L. Hunsberger, A. Micheli, R. Posenato, and M. 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. IEEE, 2014. `doi:10.1109/TIME.2014.21`.

5 A. Cimatti, L. Hunsberger, A. Micheli, R. Posenato, and M. Roveri. Dynamic controllability via Timed Game Automata. *Acta Informatica*, 53(6-8):681–722, 2016. `doi:10.1007/s00236-016-0257-2`.

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

7 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. 2014. `doi:10.1007/978-3-662-44440-5_19`.

8 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`.

9 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. IEEE, 2015. `doi:10.1109/TIME.2015.18`.

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

11 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`.

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. International Joint Conferences on Artificial Intelligence Organization, July 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 *22st International Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 4–18, 2015. `doi:10.1109/TIME.2015.26`.

15 Akhil Kumar and Russell R. Barton. Controlled violation of temporal process constraints – models, algorithms and results. *Information Systems*, 64:410 – 424, 2017. `doi:10.1016/j.is.2016.06.003`.

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

17 Andreas Lanz, Barbara Weber, and Manfred Reichert. Time patterns for process-aware information systems. *Requirements Engineering*, 19(2):113–141, 2014. `doi:10.1007/s00766-012-0162-3`.

18 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`.

19 Roberto Posenato. A CSTN(U) consistency check algorithm implementation in Java. version 1.22. http://profs.scienze.univr.it/∼posenato/software/cstnu, 2017.

20 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`.

21 Peng Yu, Jiaying Shen, Peter Z. Yeh, and Brian Williams. Resolving over-constrained conditional temporal problems using semantically similar alternatives. In *25th International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 3300–3307, 2016.