UNIVERSITÀ di VERONA

Dipartimento di INFORMATICA

Rapporto di ricerca
Research report

104/2018

February, 14 2018

Reducing Dynamic-Consistency (DC) Checking for Conditional Simple Temporal Networks (CSTNs) with Bounded Reaction Times to Standard DC Checking for CSTNs

Luke Hunsberger

Vassar College – Poughkeepsie, NY USA

Roberto Posenato

University of Verona – Verona, Italy

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 of an execution strategy to observations is bounded below by some fixed $\epsilon > 0$. This paper shows how the ϵ-DC-checking problem can be easily reduced to the standard DC-checking problem for CSTNs. 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 it includes k new observation time-points. For each observation time-point $P?$ in $\mathcal{S}$ that observes some proposition p, the time-point $P?$ in $\mathcal{S}_0$ is demoted from an observation time-point to an ordinary time-point; and the job of observing p is taken over by a new observation time-point $P_0?$ that is constrained to occur exactly ϵ after $P?$. The paper proves that $\mathcal{S}$ is ϵ-DC if and only if $\mathcal{S}_0$ is DC; and shows 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 $\mathcal{S}_0$. Two versions of these results are presented, depending on whether a dynamic strategy for $\mathcal{S}_0$ can react instantaneously or only after some arbitrarily small, positive delay. Finally, the paper demonstrates empirically that the performance of building $\mathcal{S}_0$ and DC-checking it is even better than ϵ-DC-checking the original instance $\mathcal{S}$.

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. Conditions in a CSTN are represented by propositional letters whose truth values are not controlled, but instead *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* in a CSTN generates a truth value for its corresponding propositional letter. An execution strategy for a CSTN specifies when the time-points will be executed. 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.* [1] stipulated that a strategy can react to an observation after any arbitrarily small, but positive delay. Comin *et al.* [2] defined ϵ-DC, which assumes that a strategy's reaction time is bounded below by some $\epsilon > 0$. Finally, Cairo *et al.* [3] defined π-DC, which allows a strategy to react instantaneously (i.e., after zero delay). To avoid an undesirable form of circular dependence among simultaneous observations, a π-dynamic strategy must specify an order-of-dependence among simultaneous observations.

Although several approaches to DC-checking algorithms have been presented to address the different flavors of DC [1, 4, 5, 2], the approach based on the propagation of labeled constraints is the only one that has been demonstrated to be practical [6, 7]. By making slight changes to the constraint-propagation rules, different versions of that algorithm have been used to solve the DC-checking, ϵ-DC-checking, and π-DC-checking problems.

This paper shows how the ϵ-DC-checking problem can be easily reduced to the standard DC-checking problem for CSTNs. Given any instance of the ϵ-DC-checking problem for some CSTN $\mathcal{S}$ that has k observation time-points, the reduction to the DC-checking problem is obtained by creating a new CSTN $\mathcal{S}_0$ that is the same as $\mathcal{S}$ except that it includes k new observation time-points. In particular, for each observation time-point $P?$ in $\mathcal{S}$ that is associated with a propositional letter p, the time-point $P?$ in $\mathcal{S}_0$ is demoted from an observation time-point to an ordinary time-point; and the job of observing p is taken over by a new observation time-point $P_0?$ that is constrained to occur exactly ϵ after $P?$. The paper proves that $\mathcal{S}$ is ϵ-DC if and only if $\mathcal{S}_0$ is DC; and shows 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 $\mathcal{S}_0$. Two versions of these results are presented. In the first version, the reaction times ρ of an ϵ-dynamic strategy for $\mathcal{S}$ must satisfy $\rho > \epsilon > 0$, and the corresponding reaction times ρ_0 for a dynamic strategy for $\mathcal{S}_0$ must satisfy $\rho_0 > 0$. In the second version, the reaction times of an ϵ-dynamic strategy for $\mathcal{S}$ need only satisfy $\rho \geq \epsilon > 0$, while the corresponding π-dynamic strategy for $\mathcal{S}_0$ can react instantaneously to observations (i.e., $\rho_0 \geq 0$). The paper also offers an empirically comparison of the performance of building $\mathcal{S}_0$ and DC-checking it with respect to the performance of ϵ-DC-checking the original instance $\mathcal{S}$

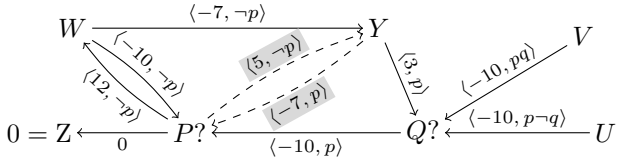

Figure 1: A sample CSTN

demostrating a practical equivalent performance on instances coming from the business application domain.

2 Background

Dechter *et al.* [8] introduced Simple Temporal Networks (STNs) to facilitate representing and reasoning about temporal constraints. 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.* [1] presented CSTNs, which augment STNs to include *observation time-points* and their associated *propositional letters.* In a CSTN, the execution of an observation time-point $P?$ generates a truth value for its associated propositional letter p. In addition, each time-point in a CSTN can be labeled by a conjunction of propositional literals that specifies the scenarios in which that time-point must be executed. Tsamardinos *et al.* noted that for any reasonable CSTN, the propositional labels on its time-points must satisfy certain properties. Hunsberger *et al.* [9, 10] later extended these properties to accommodate labels on constraints. Together, their properties formalized the notion of a *well-defined* CSTN. Recently, Cairo *et al.* [11] showed that for any well-defined CSTN, no loss of generality results from subsequently *removing* the labels from its time-points. Therefore, this paper restricts attention to CSTNs whose time-points do not have any propositional labels, what Cairo *et al.* [11] called *streamlined* CSTNs. Since streamlined CSTNs are necessarily well defined, and the applicability conditions of the constraint-propagation rules become simpler when there are no labels on time-points, the following presentation benefits dramatically from the restriction to streamlined CSTNs.

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 observation time-points 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 *true* and q is *false*. The dashed edges with shaded labels are generated by the DC-checking algorithm [6], to be discussed later on.

2.1 Streamlined CSTNs

The following definitions are from Hunsberger *et al.* [6], 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, ℓ_1 and ℓ_2 are called *consistent.*
- the *label universe* of $\mathcal{P}$, denoted by $\mathcal{P}^*$, is 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
- $\mathcal{O} : \mathcal{P} \rightarrow \mathcal{OT}$ is a bijection that associates a unique observation time-point to each propositional letter.

In a CSTN graph, $\mathcal{O}(p)$ (i.e., the observation time-point associated with p) may be 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,* $\langle \delta, \ell \rangle$. Since any time-points X and Y may participate in multiple constraints of the form, $(Y - X \leq \delta_i, \ell_i)$, the corresponding edge from X to Y may have multiple labeled values of the form, $\langle \delta_i, \ell_i \rangle$.

Definition 3 (Scenario)**.** A *scenario* over a set $\mathcal{P}$ of propositional letters is a function, $s : \mathcal{P} \rightarrow \{true, false\}$, that assigns a truth value to each letter in $\mathcal{P}$. Any such function also provides the truth value for any label $\ell \in \mathcal{P}^*$, denoted by $s(\ell)$. The set of all scenarios over $\mathcal{P}$ is denoted by $\mathcal{I}$.

Definition 4 (Schedule)**.** A *schedule* for a set of time-points $\mathcal{T}$ is a mapping, $\psi : \mathcal{T} \rightarrow \mathbb{R}$. 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., must be satisfied in scenario s).

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:

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

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, $\sigma : \mathcal{I} \rightarrow \Psi$, from scenarios to schedules. The execution time for the time-point X in the schedule $\sigma(s)$ is denoted by $[\sigma(s)]_X$.

Definition 7 (Viable Strategy)**.** 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 $\mathcal{S}(s)$.

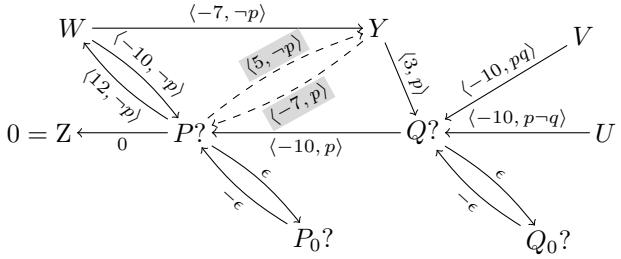

Figure 2: The reduction $\mathcal{S}_0$ corresponding to $\mathcal{S}$ from Fig. 1

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/reduction from $\mathcal{S}$ to $\mathcal{S}_0$ is the same. And, since it is a very basic translation, it is presented first.

Definition 8 (Reduction CSTN, $\mathcal{S}_0$)**.** Let $\mathcal{S}$ be any CSTN and $\epsilon > 0$ arbitrary. The *reduction* of $\mathcal{S}$ is the CSTN $\mathcal{S}_0$ that is the same as $\mathcal{S}$ except that for each observation time-point $P?$ in $\mathcal{S}$, and its associated propositional letter p:

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

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., $\epsilon = 3$).

4 Dynamic Strategies and 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	$\rho > \epsilon > 0$

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

A π-dynamic strategy can react instantaneously to observations, but must specify an order of dependence among simultaneous observations [3]. The reaction times for a (standard) dynamic strategy can be arbitrarily small, but must be positive [1]. The reaction times for an ϵ-dynamic strategy must be greater than or equal to some fixed $\epsilon > 0$ [2]. 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 DC checking; and the ϵ-DC-checking problem can be reduced to π-DC checking.

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

The following sections recall the relevant definitions for DC and $\hat{\epsilon}$-DC.

5.1 (Standard) Dynamic Strategies and (Standard) DC

(Standard) dynamic strategies were first defined by Tsamardinos *et al.* [1]. For convenience, the following definitions are in the form given by Hunsberger *et al.* [6]. A *history* at time t comprises the truth values of all propositions that were observed *before* time t.

Definition 9 (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 $Hist(t, s, \sigma)$—is the set of observations made before time t according to the schedule $\sigma(s)$:

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

Definition 10 (Dynamic Strategy)**.** An execution strategy σ for a CSTN $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$ is called *dynamic* if for any scenarios s_1 and s_2, and any time-point X:

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

In other words, if a (standard) 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.

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

$\hat{\epsilon}$-dynamic consistency has not been presented in the literature. It differs only slightly from ϵ-DC, defined later on.

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 $\hat{\epsilon}Hist(t, s, \sigma)$, is the set of observations made *before* $t - \epsilon$ according to $\sigma(s)$:

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

Definition 12 ($\hat{\epsilon}$-Dynamic Execution Strategy)**.** Let $\epsilon > 0$. An execution strategy σ 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: $\hat{\epsilon}Hist(t, s_1, \sigma) = \hat{\epsilon}Hist(t, s_2, \sigma)$
then: $[\sigma(s_2)]_X = t$.

Definition 13 ($\hat{\epsilon}$-DC)**.** Let $\epsilon > 0$. A CSTN is $\hat{\epsilon}$-*dynamically consistent* if it has a viable and $\hat{\epsilon}$-dynamic execution strategy.

Theorem 1. *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 $\hat{\epsilon}$-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:

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

Clearly, σ_0 is viable for $\mathcal{S}_0$. Suppose that it is not dynamic. Then for some scenarios s_1 and s_2, and time-point $X \in \mathcal{T}_0$, $Hist(t, s_1, \sigma_0) = Hist(t, s_2, \sigma_0)$, 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. Next:

- $\hat{\epsilon}Hist(t, s_2, \sigma)$
  $= \{(p, s_2(p)) \mid P? \in \mathcal{OT} \text{ and } [\sigma(s_2)]_{P?} < t - \epsilon\}$
  $= \{(p, s_2(p)) \mid P_0? \in \mathcal{OT}_0 \text{ and } [\sigma_0(s_2)]_{P_0?} < t\}$
  $= Hist(t, s_2, \sigma_0)$
  $= Hist(t, s_1, \sigma_0)$
  $= \{(p, s_1(p)) \mid P_0? \in \mathcal{OT}_0 \text{ and } [\sigma_0(s_1)]_{P_0?} < t\}$
  $= \{(p, s_1(p)) \mid P? \in \mathcal{OT} \text{ and } [\sigma(s_1)]_{P?} < t - \epsilon\}$
  $= \hat{\epsilon}Hist(t, s_1, \sigma)$. (†)

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 observation time-point $Q?$ at t^*, but with different results (i.e., $s_1(q) \neq s_2(q)$). If $t^* < t - \epsilon$, it would contradict that $\hat{\epsilon}Hist(t, s_1, \sigma) = \hat{\epsilon}Hist(t, s_2, \sigma)$. Therefore, $t^* = t - \epsilon$.

Now, the definition of t^* ensures that $\hat{\epsilon}Hist(t^*, s_1, \sigma) = \hat{\epsilon}Hist(t^*, s_2, \sigma)$. 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 $\hat{\epsilon}Hist(t^*, s_1, \sigma) = \hat{\epsilon}Hist(t^*, s_2, \sigma)$, 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 $X \in \mathcal{T}$, $[\sigma(s)]_X = [\sigma_0(s)]_X$. Clearly, σ is viable for $\mathcal{S}$. Suppose that it 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* $t - \epsilon$.

Definition 14 (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 $\sigma(s)$:

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

Definition 15 (ϵ-Dynamic Execution Strategy)**.** Let $\epsilon > 0$. An execution strategy σ called *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 16 (ϵ-DC)**.** Let $\epsilon > 0$. A CSTN is *ϵ-dynamically consistent* if it has a viable and ϵ-dynamic execution strategy.

6.2 π-Dynamic Execution Strategy and π-DC

This section summarizes the π-DC semantics introduced by Cairo *et al.* [3] that allows a dynamic strategy to react instantaneously to observations, but requires an order of dependence among simultaneous observations.

Definition 17 (Order of dependence)**.** For any scenario s, let $(P_1?, \ldots, P_k?)$ be an arbitrarily chosen ordering of the observation time-points in $\mathcal{OT}$, where $k = |\mathcal{OT}|$. An *order of dependence* for the time-points in $\mathcal{OT}$ is any 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 the order determined by π. In addition, it is convenient

to set $\pi(X) = \infty$ for any *non*-observation time-point X. Finally, let Π_k denote the set of all permutations over $(1, 2, \ldots, k)$.

Definition 18 (π-Execution Strategy)**.** Given any CSTN $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$, a *$\pi$-execution strategy* for $\mathcal{S}$ is a mapping, $\sigma : \mathcal{I} \to (\Psi \times \Pi_k)$, where $k = |\mathcal{OT}|$, such that for each scenario s, $\sigma(s)$ is a pair (ψ, π) such that $\psi : \mathcal{T} \to \mathbb{R}$ is a schedule for the time-points in $\mathcal{T}$; and $\pi \in \Pi_k$ is a permutation that determines an order of dependence among the time-points in $\mathcal{OT}$. For convenience, for any time-point X, $[\sigma(s)]_X$ shall denote the execution time of X (i.e., $\psi(X)$); for any observation time-point $P?$, $[\sigma(s)]^{\pi}_{P?}$ shall denote the position of $P?$ in the order of dependence (i.e., $\pi(P?)$); and for any non-observation time-point X, $[\sigma(s)]^{\pi}_{X} = \infty$. 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 $\sigma(s)$ orders $P?$ before $Q?$).

Definition 19 (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 20 (π-History)**.** Let σ be any π-execution strategy for some CSTN $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$, 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

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

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.* [3, 12].

Definition 21 (π-Dynamic Strategy)**.** A π-execution strategy, σ, for a CSTNU is called *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 X is some observation time-point $P?$ that, in the scenario s_1, σ executes at time t and position d, and the histories, $\pi Hist(t, d, s_1, \sigma)$ and $\pi Hist(t, d, s_2, \sigma)$, are the same, then in the scenario s_2, σ must execute $P?$ at the same time t, and in the same position d. The requirement for a non-observation time-point X is weaker because executing X at time t does not generate any information; therefore, it can be presumed to be ordered after any observation time-points that are executed at that same time.

Definition 22 (π-Dynamic Consistency)**.** A CSTNU, $\mathcal{S}$, is *π-dynamically consistent* (π-DC) if there exists a π-execution strategy for $\mathcal{S}$ that is both viable and π-dynamic.

Theorem 2. *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 $\mathcal{S}_0 = \langle \mathcal{T}_0, \mathcal{C}_0, \mathcal{OT}_0, \mathcal{O}_0, \mathcal{P} \rangle$ be the reduction of $\mathcal{S}$. Then $\mathcal{S}$ is ϵ-DC if and only if $\mathcal{S}_0$ is π-DC.*

Proof. ($\Rightarrow$). Suppose that $\mathcal{S}$ is ϵ-DC. Then there exists an ϵ-dynamic and viable strategy σ for $\mathcal{S}$. Define a strategy σ_0 for $\mathcal{S}_0$ exactly as in the proof of Theorem 1. Clearly, σ_0 is viable for $\mathcal{S}_0$. Suppose that it is not π-dynamic. Then for some scenarios s_1 and s_2, and time-point $X \in \mathcal{T}_0$, $\pi Hist(t, d, s_1, \sigma_0) = \pi Hist(t, d, s_2, \sigma_0)$, but $[\sigma_0(s_2)]_X \neq t$, where $t = [\sigma_0(s_1)]_X$ and $d = [\sigma_0(s_1)]^\pi_X$. With no loss of generality, assume that t is minimal for this circumstance and that, for that t, d is minimal.

Case 1: $X \in \mathcal{T}$ (i.e., $X \notin \mathcal{OT}_0$). Hence, $d = \infty$ and:

- $\epsilon Hist(t, s_2, \sigma)$
  $= \{(p, s_2(p)) \mid P? \in \mathcal{OT} \text{ and } [\sigma(s_2)]_{P?} \leq t - \epsilon\}$
  $= \{(p, s_2(p)) \mid P_0? \in \mathcal{OT}_0 \text{ and } [\sigma_0(s_2)]_{P_0?} \leq t\}$
  $= \pi Hist(t, \infty, s_2, \sigma_0) = \pi Hist(t, d, s_2, \sigma_0)$
  $= \pi Hist(t, d, s_1, \sigma_0) = \pi Hist(t, \infty, s_1, \sigma_0)$
  $= \{(p, s_1(p)) \mid P_0? \in \mathcal{OT}_0 \text{ and } [\sigma_0(s_1)]_{P_0?} \leq t\}$
  $= \{(p, s_1(p)) \mid P? \in \mathcal{OT} \text{ and } [\sigma(s_1)]_{P?} \leq t - \epsilon\}$
  $= \epsilon Hist(t, s_1, \sigma)$.

But then $[\sigma(s_2)]_X = [\sigma_0(s_2)]_X \neq [\sigma_0(s_1)]_X = [\sigma(s_1)]_X$ contradicts that σ is ϵ-dynamic.

Case 2: X is some $R_0? \in \mathcal{OT}_0$. Now $[\sigma(s_2)]_{R?} = [\sigma_0(s_2)]_{R_0?} - \epsilon \neq t - \epsilon = [\sigma_0(s_1)]_{R_0?} - \epsilon = [\sigma(s_1)]_{R?}$. Thus, the schedules for $\sigma(s_1)$ and $\sigma(s_2)$ differ at $t - \epsilon$. Let $t^* \leq t - \epsilon$ be the first time at which these schedules, annotated by their observations, differ. There are two possibilities.

Case 2a: Both execute $Q?$ at t^*, but $s_1(q) \neq s_2(q)$. But then $\sigma_0(s_1)$ and $\sigma_0(s_2)$ both execute $Q_0?$ at $t^* + \epsilon$ with different results. Now, if $t^* < t - \epsilon$, then $t^* + \epsilon < t$, which would contradict that $\pi Hist(t, d, s_1, \sigma_0) = \pi Hist(t, d, s_2, \sigma_0)$. Thus, $t^* = t$. But, by construction, $\epsilon Hist(t^*, s_1, \sigma) = \epsilon Hist(t^*, s_2, \sigma)$ which, given that $[\sigma(s_1)]_{R?} \neq [\sigma(s_2)]_{R?}$, contradicts that σ is ϵ-dynamic.

Case 2b: 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, given that $\epsilon Hist(t^*, s_1, \sigma) = \epsilon Hist(t^*, s_2, \sigma)$, this contradicts that σ is ϵ-dynamic.

($\Leftarrow$). Suppose $\mathcal{S}_0$ is π-DC. Then there is a strategy σ_0 that is π-dynamic and valid for $\mathcal{S}_0$. For each scenario s and $X \in \mathcal{T}$, let $[\sigma(s)]_X = [\sigma_0(s)]_X$. Clearly, σ is valid for $\mathcal{S}$. Suppose it is not ϵ-DC. Then for some scenarios s_1 and s_2, and some $X \in \mathcal{T}$, $\epsilon Hist(t, s_1, \sigma) = \epsilon Hist(t, s_2, \sigma)$, but $[\sigma(s_2)]_X \neq t$, where $t = [\sigma(s_1)]_X$. Since $X \in \mathcal{T}$, $d = [\sigma(s_1)]^\pi_X = \infty$ and, as shown earlier, $\pi Hist(t, \infty, s_1, \sigma_0) = \epsilon Hist(t, s_1, \sigma)$ and $\pi Hist(t, \infty, s_2, \sigma_0) = \epsilon Hist(t, s_2, \sigma)$. Then $[\sigma_0(s_2)]_X = [\sigma(s_2)]_X \neq t$ contradicts that σ_0 is π-dynamic. □

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

This section summarizes two versions of the constraint-propagation algorithm due to Hunsberger *et al.* [6]. The first version, called the IR-DC-checking algorithm (IR for *instantaneous reaction*), solves the π-DC-checking problem; the second, called the ϵ-DC-checking algorithm, solves the ϵ-DC-checking problem. Both work using only three constraint-propagation rules: The original algorithm used six rules, but in [12] authors showed that three rules are sufficient.

$\mathrm{LP}(X, u, \alpha, W, v, \beta)$:	$X \xrightarrow{\langle u, \alpha\rangle} W \xrightarrow{\langle v, \beta\rangle} \mathrm{Z}$ $\langle u+v, \alpha\beta\rangle$
$\mathrm{qR}_0(P?, w, \alpha, \tilde{p})$:	$P? \xrightarrow[\langle w, \alpha\rangle]{\langle w, \alpha\tilde{p}\rangle} \mathrm{Z}$
$\mathrm{qR}_3^*(P?, w, \alpha, v, \beta, \tilde{p}, Y)$:	$P? \xrightarrow{\langle w, \alpha\rangle} \mathrm{Z} \xleftarrow[\langle m, \alpha \star \beta\rangle]{\langle v, \beta\tilde{p}\rangle} Y$

$X, Y, W \in \mathcal{T}$; $P? \in \mathcal{OT}$; and Z is the *zero* time-point. LP applies if $\alpha\beta \in \mathcal{P}^*$; qR_0 and qR_3^* apply only if $w < 0$. In qR_0 and qR_3^*, $\tilde{p} \in \{p, \neg p, ?p\}$; p does not appear in α or β (in any form); $\alpha \star \beta$ is defined in the text; and $m = \max\{v, w\}$.

$\mathrm{LP}(X, 3, pqr, W, 4, rs\neg t)$:	$X \xrightarrow{\langle 3, pqr\rangle} W \xrightarrow{\langle 4, rs\neg t\rangle} \mathrm{Z}$ $\langle 7, pqrs\neg t\rangle$
$\mathrm{qR}_0(P?, -9, qr, ?p)$:	$P? \xrightarrow[\langle -9, qr\rangle]{\langle -9, (?p)qr\rangle} \mathrm{Z}$
$\mathrm{qR}_3^*(P?, -7, \neg q\neg r, -9, ?q, \neg p, Y)$:	$P? \xrightarrow{\langle -7, \neg q\neg r\rangle} \mathrm{Z} \xleftarrow[\langle -7, (?q)\neg r\rangle]{\langle -9, \neg p(?q)\rangle} Y$

Table 1: Constraint propagation rules for IR-DC Checking (above) and instances of their application (below)

This section then proves that applying the rules for the ϵ-DC-checking algorithm to the CSTN $\mathcal{S}$ is equivalent to applying the rules for the IR-DC-checking algorithm to the corresponding reduced CSTN $\mathcal{S}_0$.

7.1 The IR-DC-Checking Algorithm

Table 1 lists the three constraint propagation rules used by the IR-DC-checking algorithm. Note that the qR_3^* rule can generate a new kind of propositional label, called a *q-label*; and the qR_0 and qR_3^* rules can each be applied to q-labeled edges. Below, q-literals and q-labels are summarized, along with the $\star$ operator that extends conjunction to q-labels.

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 23 (Q-literals, q-labels)**.**

- A *q-literal* is a literal of the form $?p$, where $p \in \mathcal{P}$.
- For convenience, if $p \in \mathcal{P}$, then $\tilde{p}, \tilde{p}_1$ or $\tilde{p}_2$ may be used to denote arbitrary elements of $\{p, \neg p, ?p\}$.
- A *q-label* is a conjunction of literals and/or q-literals.
- $\mathcal{Q}^*$ denotes the set of all q-labels.

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 C_1 is labeled by p, and C_2 is labeled by $\neg p$, then *both* constraints must hold as long as the value of p is unknown, which is represented by $p \star \neg p = ?p$.

Definition 24 ($\star$)**.** The operator, $\star : \mathcal{Q}^* \times \mathcal{Q}^* \rightarrow \mathcal{Q}^\star$, is defined in two steps. First, for any $p \in \mathcal{P}$, $p \star p = p$ and $\neg p \star \neg p = \neg p$; otherwise, $\tilde{p}_1 \star \tilde{p}_2 = ?p$. Next, for any q-labels, $\ell_1, \ell_2 \in \mathcal{Q}^*$, $\ell_1 \star \ell_2 \in \mathcal{Q}^*$ denotes the conjunction obtained by applying the $\star$ operator in pairwise fashion to corresponding literals from ℓ_1 and ℓ_2, as follows.

- If $\tilde{p}_1, \tilde{p}_2$ are literals in ℓ_1 and ℓ_2, respectively, then $\tilde{p}_1 \star \tilde{p}_2$ is contained in the conjunction, $\ell_1 \star \ell_2$.
- If $\tilde{p}$ is in ℓ_1, but proposition p does not appear in ℓ_2, then $\tilde{p}$ is contained in the conjunction, $\ell_1 \star \ell_2$.
- If $\tilde{p}$ is in ℓ_2, but proposition p does not appear in ℓ_1, then $\tilde{p}$ is contained in the conjunction $\ell_1 \star \ell_2$.

For example: $(p\neg q(?r)) \star (q\neg s) = p(?q)(?r)\neg s$.

The ϵ-DC-checking algorithm uses the same three rules, 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 $\{\text{LP}, \text{qR}_0, \text{qR}_3^\epsilon\}$ shall be called the ϵ-DC-checking rules.

Lemma 1. *Let $\mathcal{S}$ be any CSTN, $\epsilon > 0$, and $\mathcal{S}^*$ 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 IR-DC-checking rules to $\mathcal{S}_0$. Then $\mathcal{S}^*$ and $\mathcal{S}_0^*$ are equivalent in the following sense:*

(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.

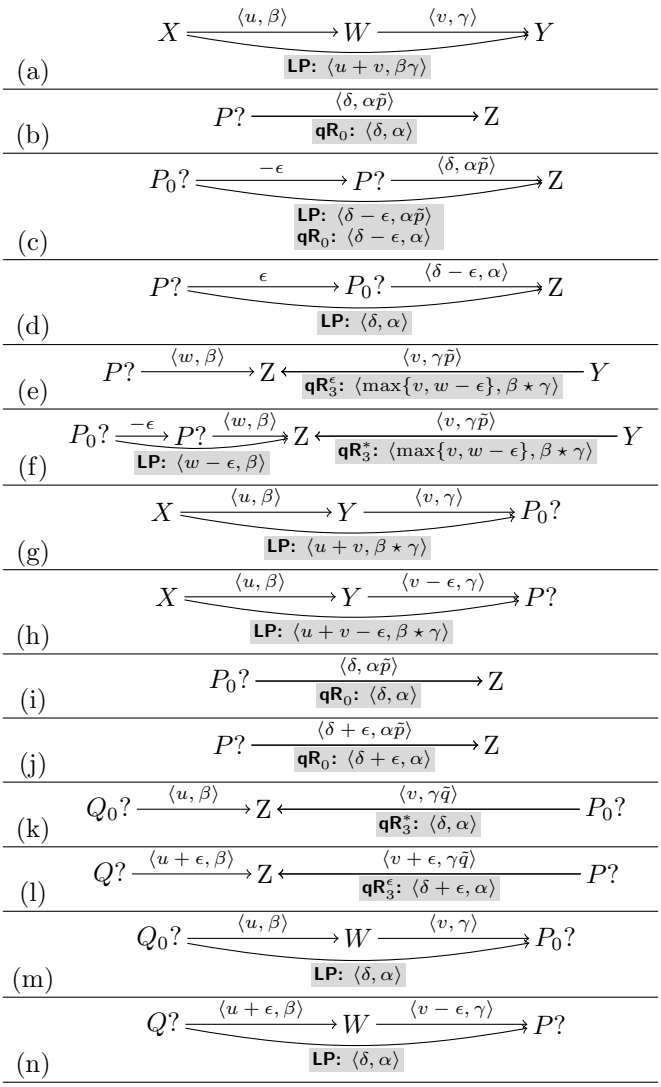

Figure 3: Constraint propagations for the proof of Lemma 1

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 lemma.

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 $\mathcal{S}^*$, as shown in Fig. 3h, which 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 preceding argument assumed that $Y \in \mathcal{T}$; however, 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 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 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 assumption, 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, which corresponds to K, a contradiction.

Case 3: K has the form $(P_0? - Q_0? \leq \delta, \alpha)$. Since $P_0? \not\equiv \mathrm{Z}$, 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 $\mathcal{OT}_0$, where $\delta = u + v$ and $\alpha = \beta\gamma$. By assumption, 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 $P?$ that corresponds to K, a contradiction. The case where $W \in \mathcal{OT}_0$ is handled similarly. □

A similar set of six constraint-propagation rules has been proposed for solving the (standard) DC-checking problem [7]. However, that DC-checking algorithm has not been proven to be complete. Therefore, we do not address it here. No algorithm for solving the $\hat{\epsilon}$-DC-checking problem has been presented.

8 Empirical Evaluation

This section compares the performances of an implementation of ϵ-DC-checking algorithm [7], `ϵ-DC-Ch`, and an implementation IR-DC-checking algorithm (that solves the π-DC-checking problem), S_0`IR-DC-Ch`, applied to the reduced CSTN $\mathcal{S}_0$. Both the implementations were obtained from Posenato [13]. 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. Both implementations were tested on instances of the four benchmarks from Hunsberger and Posenato [7]. Each benchmark has at least 60 DC and 60 non-DC CSTNs, obtained from random workflow schemata generated by the ATAPIS toolset [14]. The numbers of activites (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. Each data point represents the average of the execution times

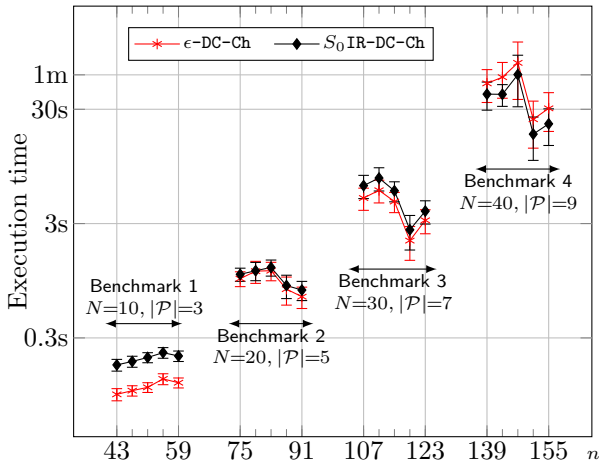

Figure 4: Execution time vs. number of time-points n

for instances of the given size. We extended the benchmarks, adding up to 50 DC instances, to generate tight error bars, each representing a 95% confidence interval.

The results demostrate that, while ϵ-DC-Ch algorithm has a clear better performance in Benchmark 1, the performance difference changes as the size of the instances increases. The main reason for such behavior is that in small instances the addition of new nodes has a significant impact on the number of to-do propagations, while such impact is less important when the number of nodes increases—as in Benchmark 4. Experiments have shown that it is possible to adopt S_0IR-DC-Ch algorithm instead of ϵ-DC-Ch algorithm without practical loss of performances.

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 IR-DC-checking algorithm (that solves the π-DC-checking problem) 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.

References

[1] I. Tsamardinos, T. Vidal, and M. E. Pollack, "CTP: A new constraint-based formalism for conditional, temporal planning," *Constraints*, vol. 8, pp. 365–388, 2003. doi:10.1023/A:1025894003623.

[2] C. Comin and R. 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)*, pp. 19–28, IEEE, Sept. 2015. doi:10.1109/TIME.2015.18.

[3] M. Cairo, C. Comin, and R. Rizzi, "Instantaneous reaction-time in dynamic-consistency checking of conditional simple temporal networks," in *TIME 2016*, Oct. 2016.

[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)*, pp. 27–36, IEEE, Sept. 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*, vol. 53, no. 6-8, pp. 681–722, 2016. doi:10.1007/s00236-016-0257-2.

[6] L. Hunsberger, R. Posenato, and C. 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)*, pp. 4–18, IEEE, Sept. 2015. doi:10.1109/TIME.2015.26.

[7] L. Hunsberger and R. Posenato, "Checking the Dynamic Consistency of Conditional Temporal Networks with Bounded Reaction Times," in *Proceedings of the 26th International Conference on Automated Planning and Scheduling, ICAPS 2016*, pp. 175–183, 2016. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/view/13108.

[8] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, vol. 49, no. 1-3, pp. 61–95, 1991. doi:10.1016/0004-3702(91)90006-6.

[9] L. Hunsberger, R. Posenato, and C. Combi, "The Dynamic Controllability of Conditional STNs with Uncertainty," in *PlanEx at ICAPS 2012*, pp. 1–8, June 2012.

[10] C. Combi, L. Hunsberger, and R. Posenato, "An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty," in *Proceedings of the 5th International Conference on Agents and Artificial Intelligence (ICAART-2013)*, vol. 2, pp. 144–156, Feb. 2013.

[11] M. Cairo, L. Hunsberger, R. Posenato, and R. Rizzi, "A Streamlined Model of Conditional Simple Temporal Networks - Semantics and Equivalence Results," in *24th International Symposium on Temporal Representation*

and Reasoning (TIME 2017), vol. 90 of *LIPIcs*, pp. 10:1–10:19, 2017. doi:10.4230/LIPIcs.TIME.2017.10.

[12] L. Hunsberger and R. Posenato, "Dynamic-consistency checking for conditional simple temporal networks: Strengthening the theoretical foundations and presenting a faster algorithm," Tech. Rep. 103, Computer Scienze Department-University of Verona, Feb. 2018. URL: http://hdl.handle.net/11562/973404.

[13] R. Posenato, "A CSTN(U) consistency check algorithm implementation in Java.." http://profs.scienze.univr.it/∼posenato/software/cstnu, Apr. 2015.

[14] A. Lanz and M. Reichert, "Enabling time-aware process support with the atapis toolset," in *Proceedings of the BPM Demo Sessions 2014* (L. Limonad and B. Weber, eds.), vol. 1295 of *CEUR Workshop Proceedings*, pp. 41–45, 2014.

UNIVERSITÀ
di **VERONA**

Dipartimento
di **INFORMATICA**

University of Verona
Department of Computer Science
Strada Le Grazie, 15
I-37134 Verona
Italy

http://www.di.univr.it