Simpler and Faster Algorithm for Checking the Dynamic Consistency of Conditional Simple Temporal Networks

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

Roberto Posenato
Department of Computer Science
University of Verona, Italy
roberto.posenato@univr.it

Abstract

Recent work on Conditional Simple Temporal Networks (CSTNs) has focused on checking the dynamic consistency (DC) property assuming that execution strategies can react instantaneously to observations. Three alternative semantics—IR-DC, 0-DC, and π-DC—have been presented. The most practical DC-checking algorithm for CSTNs has only been analyzed with respect to the IR-DC semantics, while the 0-DC semantics was shown to have a serious flaw that the π-DC semantics fixed. Whether the IR-DC semantics had the same flaw and, if so, what the consequences would be for the DC-checking algorithm remained open questions.

This paper (1) shows that the IR-DC semantics is also flawed; (2) shows that one of the constraint-propagation rules from the IR-DC-checking algorithm is *not sound* with respect to the IR-DC semantics; (3) presents a simpler algorithm, called the π-DC-checking algorithm; (4) proves that it is sound and complete with respect to the π-DC semantics; and (5) empirically evaluates the new algorithm.

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. Tsamardinos *et al.* [2003] stipulated that a strategy can react to an observation after any arbitrarily small delay. Comin *et al.* [2015] defined ϵ-DC, which assumes that a strategy's reaction times are bounded below by a fixed $\epsilon > 0$. Finally, three different versions of DC for strategies that can react instantaneously (i.e., after zero delay) have been defined: IR-DC [Hunsberger *et al.*, 2015]; 0-DC and π-DC [Cairo *et al.*, 2016].

Several approaches to DC-checking algorithms have been presented to address the different flavors of DC [Tsamardinos *et al.*, 2003; Cimatti *et al.*, 2014; 2016; Comin and Rizzi, 2015], but the approach based on the propagation of labeled constraints [Hunsberger *et al.*, 2015; Hunsberger and Posenato, 2016] is the only one that has been demonstrated to be practical. Three variations of their DC-checking algorithm have been presented: one for DC, one for ϵ-DC, and one for IR-DC. This paper focuses on their IR-DC-checking algorithm.

Subsequently, Cairo *et al.* [2016] showed that the 0-DC semantics (i.e., the ϵ-DC semantics where $\epsilon = 0$) had a serious flaw: it allows a kind of circular dependence among simultaneous observations. To correct this, their π-DC semantics requires a strategy to specify an order-of-dependence among simultaneous observations. Whether the IR-DC semantics is similarly flawed and, if so, what the consequences would be for the IR-DC-checking algorithm, remained open questions.

This paper (1) shows that the IR-DC semantics is also flawed; (2) shows that one of the constraint-propagation rules from the IR-DC-checking algorithm is *not sound* with respect to the IR-DC semantics; (3) presents a simpler algorithm, called the π-DC-checking algorithm; (4) proves that the new algorithm is sound and complete with respect to the π-DC semantics; and (5) empirically evaluates the new algorithm.

2 Background

Dechter *et al.* [1991] 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 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.* [2003] 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 letter p. In addition, each time-point can be labeled by

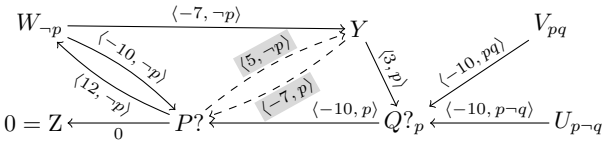

Figure 1: A sample CSTN

a conjunction of literals specifying the scenarios in which that time-point must be executed. Later work generalized CSTNs to also include labels on constraints [Hunsberger *et al.*, 2012].

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. $Q?$ being labeled by p indicates that $Q?$ is executed only if p is observed to be $true$; and 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 are generated by the IR-DC-checking algorithm [Hunsberger *et al.*, 2015], discussed later.

Tsamardinos *et al.* [2003] noted that for any reasonable CSTN, the propositional labels on its time-points must satisfy certain properties. Hunsberger *et al.* [2012] extended those properties to cover labels on constraints, and formalized the notion of a *well-defined* CSTN. Recently, Cairo *et al.* [2017] showed that for any well-defined CSTN, no loss of generality results from *removing* the labels from its time-points. Therefore, this paper restricts attention to CSTNs whose time-points do not have any labels, the so-called *streamlined* CSTNs.

2.1 Streamlined CSTNs

The following definitions are from Hunsberger *et al.* [2015], 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
- $\mathcal{O}\colon \mathcal{P} \rightarrow \mathcal{OT}$ is a bijection that associates a unique observation time-point to each propositional letter.

In a CSTN graph, the observation time-point for p (i.e., $\mathcal{O}(p)$) may be denoted by $P?$; and each labeled constraint, $(Y - X \leq \delta, \ell)$, is represented by an arrow from X to Y, $X \xrightarrow{\langle \delta, \ell \rangle} Y$, annotated by the *labeled value* $\langle \delta, \ell \rangle$. 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, $s\colon \mathcal{P} \rightarrow \{true, false\}$, 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, $\psi\colon \mathcal{T} \rightarrow \mathbb{R}$. The set of all schedules *for any subset of* $\mathcal{T}$ 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:

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

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

3 Strategies with Instantaneous Reaction

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. Three *distinct* semantics for strategies that can react *instantaneously* have been defined: IR-dynamic [Hunsberger *et al.*, 2015]; and 0-dynamic and π-dynamic [Cairo *et al.*, 2016], each yielding a distinct version of DC. Since Cairo *et al.* [2016] showed that the 0-DC semantics is flawed, this paper focuses on the IR-DC and π-DC semantics.

3.1 IR-Dynamic Strategies and IR-DC

This section demonstrates that the definition of IR-dynamic strategies (IR for "instantaneous reaction") [2015] is flawed in that it also allows a kind of circular dependence among simultaneous observations.

Definition 7 (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 8 (IR-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 *IR-dynamic* if for any scenarios s_1 and s_2, and any time-point $X \in \mathcal{T}$:

let: $t = [\sigma(s_1)]_X$
if: $Hist(t, s_1, \sigma) = Hist(t, s_2, \sigma)$
then: $[\sigma(s_2)]_X = t$
unless: $[\sigma(s_1)]_{Q?} = [\sigma(s_2)]_{Q?} = t$ and, for some $Q? \in \mathcal{OT}$, $Q? \not\equiv X$ and $s_1(q) \neq s_2(q)$.

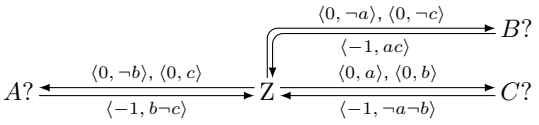

Figure 2: An absurd CSTN that is nonetheless IR-DC

LP$(X, u, \alpha, Y, v, \beta)$:	$X \xrightarrow{\langle u,\alpha\rangle} Y \xrightarrow{\langle v,\beta\rangle} \mathrm{Z}$, $\langle u+v, \alpha\beta\rangle$
$\mathrm{qR}_0(P?, w, \alpha, \tilde{p})$:	$P? \xrightarrow{\langle w,\alpha\tilde{p}\rangle} \mathrm{Z}$, $\langle w,\alpha\rangle$
$\mathrm{qR}_3^*(P?, w, \alpha, v, \beta, \tilde{p}, Y)$:	$P? \xrightarrow{\langle w,\alpha\rangle} \mathrm{Z} \xleftarrow{\langle v,\beta\tilde{p}\rangle} Y$, $\langle m, \alpha\star\beta\rangle$
$X, Y \in \mathcal{T}$; $P? \in \mathcal{OT}$; and Z is fixed at 0. LP applies if $\alpha\beta \in \mathcal{P}^*$; qR_0 and qR_3^* apply if $w < 0$. In qR_0 and qR_3^*, $\tilde{p} \in \{p, \neg p, ?p\}$; p does not appear in α or β; and $m = \max\{v, w\}$.	

LP$(X, -3, pqr, Y, -4, rs\neg t)$:	$X \xrightarrow{\langle -3,pqr\rangle} Y \xrightarrow{\langle -4,rs\neg t\rangle} \mathrm{Z}$, $\langle -7, pqrs\neg t\rangle$
$\mathrm{qR}_0(P?, -9, qr, ?p)$:	$P? \xrightarrow{\langle -9,(?p)qr\rangle} \mathrm{Z}$, $\langle -9, qr\rangle$
$\mathrm{qR}_3^*(A?, -1, b\neg c, -1, c, a, B?)$:	$A? \xrightarrow{\langle -1,b\neg c\rangle} \mathrm{Z} \xleftarrow{\langle -1,ac\rangle} B?$, $\langle -1, b(?c)\rangle$

Table 1: The LP, qR_0 and qR_3^* propagation rules for CSTNs (above) and instances of their use (below)

Thus, if an IR-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 observations, then σ must also execute X at t in s_2, unless some observation at time t, where $Q?$ and X are distinct time-points, yielded different results in the two scenarios.

That the definition of IR-dynamic strategies is flawed is demonstrated by the CSTN in Fig. 2, which is equivalent to a CSTN from Cairo *et al.* [2016]. It has three observation time-points, $A?$, $B?$ and $C?$, with corresponding propositional letters, a, b and c. $A?, B?$ and $C?$ are each forced to execute at 0 in some scenarios, and at or above 1 in another scenario. Thus, there is no time-point that can be executed first, at or after Z, implying that no implementable strategy can exist. Yet, the strategy, σ', defined below, is viable and IR-dynamic.

$$\begin{array}{lll} \text{if } s \models b\neg c, & [\sigma'(s)]_{A?} = 1, & \text{else } [\sigma'(s)]_{A?} = 0. \\ \text{if } s \models ac, & [\sigma'(s)]_{B?} = 1, & \text{else } [\sigma'(s)]_{B?} = 0. \\ \text{if } s \models \neg a\neg b, & [\sigma'(s)]_{C?} = 1, & \text{else } [\sigma'(s)]_{C?} = 0. \end{array}$$

A tedious check confirms that for any scenarios, s_1 and s_2, and any $X? \in \{A?, B?, C?\}$, where $[\sigma'(s_1)]_{X?} = 0$ and $[\sigma'(s_2)]_{X?} = 1$, there is always *another* observation time-point $Y? \not\equiv X?$ that σ' executes at 0, but for which $s_1(y) \neq s_2(y)$. Thus, the "unless" clause of Defn. 8 invariably applies, allowing a circular dependence among $A?, B?$ and $C?$.

3.2 The IR-DC-checking Algorithm

Hunsberger *et al.* [2015] claimed that their IR-DC-checking algorithm was sound and complete. But their algorithm says the CSTN in Fig. 2 is *not* IR-DC. Something is wrong.

Table 1 lists three of the six constraint-propagation rules from their IR-DC-checking algorithm. Unlike the original, the LP rule shown in Table 1 focuses on generating edges terminating at Z, but that restriction does not affect what follows.

The qR_3^* rule can generate a new kind of label, called a *q-label*; the qR_0 and qR_3^* rules can each be applied to q-labeled edges. 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.

Definition 9 (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 $?p$, we 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 10 ($\star$)**.** The operator, $\star$: $\mathcal{Q}^* \times \mathcal{Q}^* \rightarrow \mathcal{Q}^\star$, is defined thusly. First, for any $p \in \mathcal{P}$, $p \star p = p$ and $\neg p \star \neg p = \neg p$. Next, for any $p_1, p_2 \in \{p, \neg p, ?p\}$ for which $p_1 \neq p_2$, $p_1 \star p_2 = ?p$. Finally, 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 $\star$ in pairwise fashion to matching literals from ℓ_1 and ℓ_2, and conjoining any unmatched literals.

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

Lemma 1. *The IR-DC-checking algorithm is not sound with respect to the IR-DC semantics.*

Proof. The CSTN from Fig. 2 is IR-DC, given the valid and IR-dynamic strategy σ'. But the IR-DC-checking algorithm declares it to be *not* IR-DC, as follows. First, applying the qR_3^* rule as shown in the last row of Table 1 generates the constraint $(B? \geq 1, b(?c))$.[1] Next, applying qR_0 to this constraint yields: $(B? \geq 1, (?c))$ (i.e., B must be at least 1 as long as c is unknown). But this constraint is not satisfied by σ'. For example, in the scenario $\neg a\neg bc$, σ' executes $B?$ at 0, but $C?$ at 1. Furthermore, the *Spreading Lemma* [Hunsberger *et al.*, 2015] ensures that continued application of the qR_3^* and qR_0 rules will generate the constraints, $(A? \geq 1, \boxdot)$, $(B? \geq 1, \boxdot)$ and $(C? \geq 1, \boxdot)$ (i.e., $A?$, $B?$ and $C?$ must all be executed at or after 1 in *all* scenarios), which is inconsistent, for example, with the constraint $(A? \leq 0, c)$. This inconsistency is detected by the LP rule, which generates a negative self-loop with a consistent propositional label, causing the algorithm to declare that the network is *not* IR-DC, which is wrong. □

3.3 π-Dynamic Strategies and π-DC

This section summarizes the π-DC semantics [Cairo *et al.*, 2016] that forces strategies to select orders of dependence among simultaneous observations. The Table 1 rules will be shown to be sound and complete for the π-DC semantics.

Definition 11 (Order of dependence)**.** For any scenario s, and 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 $(1, 2, \ldots, k)$.

[1] $(B? \geq 1)$ is an abbreviation for $(\mathrm{Z} - B? \leq -1)$.

Definition 12 (π-Execution Strategy)**.** Given any CSTN $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$, let $k = |\mathcal{OT}|$. A *π-execution strategy* for $\mathcal{S}$ is a mapping, $\sigma \colon \mathcal{I} \rightarrow (\Psi \times \Pi_k)$, such that for each scenario s, $\sigma(s)$ is a pair (ψ, π) where $\psi \colon \mathcal{T} \rightarrow \mathbb{R}$ 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 13 (Viability)**.** The π-execution strategy σ is called *viable* for the CSTN $\mathcal{S}$ if for each scenario s, the schedule ψ is a solution to the projection $\mathcal{S}(s)$, where $\sigma(s) = (\psi, \pi)$.

Definition 14 (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

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

The π-history specifies the truth value of each $p \in \mathcal{P}$ that is observed *before* t in the schedule ψ, or *at* t if the corresponding $P?$ is ordered *before* position d by the permutation π.

The following definition is equivalent to that given by Cairo *et al.* [2016]. (Proof omitted to save space.) It was chosen to facilitate comparison with IR-dynamic strategies (Defn. 8).

Definition 15 (π-Dynamic Strategy)**.** A π-execution strategy, σ, for a CSTN is called *π-dynamic* if for every pair of scenarios, s_1 and s_2, and every time-point $X \in \mathcal{T}$:

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

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 16 (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.

4 Sound and Complete π-DC Checking

This section proves that the constraint-propagation rules in Table 1 are sound and complete for the π-DC semantics.

Lemma 2 (Soundness of the LP rule)**.** *If σ is a valid and π-dynamic strategy for a CSTN $\mathcal{S}$ that includes the constraints $(W - X \leq u, \alpha)$ and $(Y - W \leq v, \beta)$, where $\alpha\beta \in \mathcal{P}^*$, then σ also satisfies the constraint $(Y - X \leq u + v, \alpha\beta)$ (i.e., for each scenario s, if $s \models \alpha\beta$, then $[\sigma(s)]_Y - [\sigma(s)]_X \leq u + v$).*

Proof. Let s be any scenario for which $s \models \alpha\beta$. Since σ is valid and $s \models \alpha$, $[\sigma(s)]_W - [\sigma(s)]_X \leq u$. Similarly, since σ is valid and $s \models \beta$, $[\sigma(s)]_Y - [\sigma(s)]_W \leq v$. Summing these inequalities yields: $[\sigma(s)]_Y - [\sigma(s)]_X \leq u + v$. □

Definition 17 (π-before)**.** For a given scenario s, we say that an execution strategy σ observes p (or executes $P?$) *π-before* executing Y if $[\sigma(s)]_{P?} \leq [\sigma(s)]_Y$ and $[\sigma(s)]^{\pi}_{P?} < [\sigma(s)]^{\pi}_Y$.

Intuitively, if σ observes p π-before Y, then σ's decision to execute Y can depend on the value of p. In addition, note that if $P?$ and $Q?$ are distinct, then a coherent strategy σ observes p π-before $Q?$ if and only if $[\sigma(s)]^{\pi}_{P?} < [\sigma(s)]^{\pi}_{Q?}$.

Since the qR_0 and qR^*_3 rules involve lower-bound constraints whose labels may be q-labels, we first provide a semantics for such constraints for π-dynamic strategies. The following definitions extends that of Hunsberger *et al.* [2015].

Definition 18 (Satisfying a lower-bound constraint)**.** A strategy σ satisfies the lower-bound constraint $(Y \geq \delta, \alpha)$, where $\alpha \in \mathcal{Q}^*$, iff for each scenario s: (1) $[\sigma(s)]_Y \geq \delta$; or (2) some $\tilde{a} \in \{a, \neg a, ?a\}$ appears in α, where σ observes a π-before Y, and $s \not\models \tilde{a}$. (i.e., $\sigma(s)$ can execute Y *before* δ only if some observation was made π-before Y that ensured that $s \not\models \alpha$).

Lemma 3 (Soundness of qR_0)**.** *If σ is a valid and π-dynamic strategy that satisfies $(P? \geq \delta, \alpha\tilde{p})$, where $\alpha \in \mathcal{Q}^*$ and $\tilde{p} \in \{p, \neg p, ?p\}$, then σ also satisfies $(P? \geq \delta, \alpha)$.*

Proof. If σ satisfies $(P? \geq \delta, \alpha\tilde{p})$, for some $\tilde{p} \in \{p, \neg p, ?p\}$ and $\alpha \in \mathcal{Q}^*$, then for any scenario s either: (1) $[\sigma(s)]_{P?} \geq \delta$; or (2) for some $\tilde{a} \in \alpha\tilde{p}$, σ observes a π-before $P?$ (and hence $A? \not\equiv P?$) and $s \not\models \tilde{a}$. Since qR_0 stipulates that p does not appear in α, (1) and (2) are equivalent to the satisfaction conditions for $(P? \geq \delta, \alpha)$. □

Lemma 4 (Soundness of the qR^*_3 rule)**.** *Let σ be any viable and π-dynamic strategy for a CSTN $\mathcal{S}$. If σ satisfies the constraints $(P? \geq g, \alpha)$ and $(Y \geq h, \beta\tilde{p})$, where $\alpha, \beta \in \mathcal{Q}^*$, then σ also satisfies $(Y \geq x, \alpha \star \beta)$, where $x = \min\{g, h\}$.*[2]

Proof. Suppose the conditions of the lemma hold, but σ does not satisfy $(Y \geq x, \alpha \star \beta)$. Then for some scenario s, the *negation* of Defn. 18 holds: ($1^\dagger$) $[\sigma(s)]_Y < x = \min\{g, h\}$; and ($2^\dagger$) for each $\tilde{a} \in \{a, \neg a, ?a\}$ appearing in $\alpha \star \beta$ such that σ observes a π-before Y, $s \models \tilde{a}$. Since σ satisfies $(Y \geq h, \beta\tilde{p})$, one of the following holds: (1) $[\sigma(s)]_Y \geq h \geq x$; or (2) some $\tilde{b} \in \{b, \neg b, ?b\}$ appears in $\beta\tilde{p}$ such that σ observes b π-before Y and $s \not\models \tilde{b}$. Since (1) contradicts ($1^\dagger$), (2) must hold for some $\tilde{b} \in \beta\tilde{p}$. If $\tilde{b} = b \in \beta$ and $b \in \alpha \star \beta$, then $s \not\models b$ contradicts ($2^\dagger$); but if $?b \in \alpha \star \beta$, then $s \not\models ?b$ again contradicts ($2^\dagger$). Similar remarks apply to the cases, $\tilde{b} = \neg b$ and $\tilde{b} = ?b$. Finally, if (2) holds for $\tilde{b} = \tilde{p}$, then σ observes p π-before Y and $s \not\models \tilde{p}$. Since σ satisfies $(P? \geq g, \alpha)$, but $[\sigma(s)]_{P?} \leq [\sigma(s)]_Y < g$, Defn. 18 implies that some $\tilde{c} \in \{c, \neg c, ?c\}$ appears in α such that σ observes c π-before $P?$ and $s \not\models \tilde{c}$. But since σ observes p π-before Y, it follows that σ observes c π-before Y (i.e., $[\sigma(s)]^{\pi}_{C?} < [\sigma(s)]^{\pi}_{P?} < [\sigma(s)]^{\pi}_Y$). And $\tilde{c} \in \alpha$ implies that either $\tilde{c}$ or $?c$ is in $\alpha \star \beta$, contradicting ($2^\dagger$). □

4.1 The π-DC-Checking Algorithm for CSTNs

Unlike the 6-rule IR-DC-checking algorithm, our **new** π-DC-checking algorithm, whose pseudo-code is shown in Algorithm 1, (1) uses only the three rules from Table 1; and (2) only generates edges terminating at Z. It begins (Lines 1–2) by constraining each time-point X thusly: $0 \leq X \leq h = Mn$, where M is the maximum absolute value of any negative

[2] Substitutions from Table 1: $g = -w$; $h = -v$; and $x = -m$.

Algorithm 1: π-DC-Check

Input: $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$, any (streamlined) CSTN
1 **foreach** $(X \in \mathcal{T})$ **do**
2 Insert $(X \geq 0, \boxdot)$ and $(X \leq Mn, \boxdot)$ // $Mn =$ horizon
3 $\mathcal{C}_{\text{new}} := \emptyset$; $\mathcal{C}_{\text{prev}} := \mathcal{C}$
4 **while** *(*$\mathcal{C}_{\text{prev}} \neq \emptyset$*)* **do**
5 **foreach** *(*$(\mathrm{Z} - X \leq v, \ell) \in \mathcal{C}_{\text{prev}}$*)* **do**
6 **if** *(*$(X \equiv P?)$ *and* $(\ell = \alpha\tilde{p})$*)* **then**
7 $\mathcal{C}_{\text{new}} := \mathcal{C}_{\text{new}} \cup \text{qR}_0(P?, v, \alpha, \tilde{p})$
8 **foreach** *(*$(\mathrm{Z} - Q? \leq w, \alpha) \in \mathcal{C} \mid \ell = \beta\tilde{q}$*)* **do**
9 $\mathcal{C}_{\text{new}} := \mathcal{C}_{\text{new}} \cup \text{qR}_3^*(Q?, w, \alpha, v, \beta, \tilde{q}, X)$
10 **foreach** *(*$(X - Y \leq w, \ell') \in \mathcal{C}$*)* **do**
11 $\mathcal{C}_{\text{new}} := \mathcal{C}_{\text{new}} \cup \text{LP}(Y, v, \ell, X, w, \ell')$
12 **if** *(*$\exists$ *a negative self-loop in* $\mathcal{C}_{\text{new}}$ *with label in* $\mathcal{P}^*$*)* **then**
13 **return** $\mathcal{S}$ *is* not π*-DC*
14 $\mathcal{C} := \mathcal{C} \cup \mathcal{C}_{\text{new}}$; $\mathcal{C}_{\text{prev}} := \mathcal{C}_{\text{new}}$; $\mathcal{C}_{\text{new}} = \emptyset$
15 **return** $\mathcal{S}$ *is* π*-DC*

weight in the CSTN, and $n = |\mathcal{T}|$. Cairo *et al.* [2017] proved that inserting such horizon constraints preserves the DC property assuming that all weights are rational.

Each iteration of the main loop (Lines 5–13) processes each constraint $C \in \mathcal{C}_{\text{prev}}$ generated by the *previous* iteration, checking for possible applications of qR_0 (Lines 6–7), qR_3^* (Lines 8–9), and LP (Lines 10–11). Only constraints that are not entailed by constraints already in $\mathcal{C}$ are collected in $\mathcal{C}_{\text{new}}$. The main loop ends when: (1) a negative self-loop with a consistent label is found (Line 12); or (2) no new constraints were generated by the current iteration (Line 4). The algorithm reports *not* π-DC in the first case; π-DC in the second.

An upper bound for the computational complexity of the π-DC-checking algorithm is $O(M|\mathcal{T}|(|\mathcal{P}||\mathcal{T}|3^{|\mathcal{P}|} + |\mathcal{T}|^2 2^{|\mathcal{P}|}))$ $= O(M|\mathcal{T}|^4 3^{|\mathcal{P}|})$. Although exponential in the worst case, it will be shown to be practical across a variety of networks. The space complexity is $O(|\mathcal{T}|^2 4^{|\mathcal{P}|})$, but is typically *much* smaller in practice.

4.2 Completeness of π-DC Checking

This section proves that the π-DC-checking algorithm is sound and complete with respect to the π-DC semantics. It generalizes the approach used by Hunsberger *et al.* [2015].

During execution, the observations made so far can be represented by a label $\ell \in \mathcal{P}^*$ that is equivalent to the π-history at that point. For convenience, such a label is called a *current partial scenario* (CPS). A q-label $\alpha \in \mathcal{Q}^*$ is called *applicable* in a given CPS ℓ, if the observations in ℓ are consistent with α.

Definition 19 (Applicable q-Label)**.** A q-label α is *applicable* in a CPS ℓ, notated as $appl(\alpha, \ell)$, if whenever any letter p appears in both ℓ and α, it appears identically (i.e., as p in both or as $\neg p$ in both).

Note that if $?p \in \alpha$, then $appl(\alpha, \ell)$ only holds if p has not yet been observed (i.e., if p does not appear in ℓ).

Lemma 5. *If σ is a viable and π-dynamic strategy for some CSTN $\mathcal{S}$, then σ satisfies the constraint $(X \geq \delta, \alpha)$ if and only if for each scenario s, $[\sigma(s)]_X < \delta \Rightarrow \neg appl(\alpha, \ell)$, where $\ell = \pi Hist([\sigma(s)]_X, [\sigma(s)]_X^{\pi}, s, \sigma)$ is the relevant CPS.*

Proof. Suppose $[\sigma(s)]_X < \delta$. Since σ satisfies $(X \geq \delta, \alpha)$, there must be some $\tilde{a} \in \{a, \neg a, ?a\}$ appearing in α such that σ observes a π-before X, and $s \not\models \tilde{a}$. Since σ observes a π-before X, then either a or $\neg a$ appears in the CPS ℓ. Now, if $\tilde{a} = a$, then $a \in \alpha$, but $s \not\models a$ implies that $\neg a \in \ell$; and, hence, $\neg appl(\alpha, \ell)$. The case, $\tilde{a} = \neg a$, is similar. And if $\tilde{a} = ?a$, then $?a \in \alpha$ and a appearing in ℓ imply that $\neg appl(\alpha, \ell)$. □

Lemma 6 (Spreading Lemma)**.** *Let $\mathcal{S}$ be any CSTN whose set of constraints $\mathcal{C}$ is closed under the* qR_0 *and* qR_3^* *rules. Let $\ell \in \mathcal{P}^*$ be any consistent label, representing a possible CPS. Let $\mathcal{T}_\ell = \{P? \mid p \text{ appears in } \ell\}$ be the observation time-points corresponding to the observations in ℓ. Let $\mathcal{T}_x$ be any set of time-points such that $\mathcal{T}_\ell \subseteq \mathcal{T}_x \subset \mathcal{T}$, and $\mathcal{T}_x \cap \mathcal{OT} = \mathcal{T}_\ell$. $\mathcal{T}_x$ represents the time-points that have been executed so far. And let $\mathcal{T}_u = \mathcal{T} \setminus \mathcal{T}_x$ be the as-yet-unexecuted time-points. For each $X \in \mathcal{T}_u$, let $ELB(X, \ell)$ be its* effective lower bound:

$$\max\{\delta \mid \exists \alpha \in \mathcal{Q}^* \text{ and } (X \geq \delta, \alpha) \in \mathcal{C} \text{ such that } appl(\alpha, \ell)\}.$$

And let $\Lambda(\ell, \mathcal{T}_u)$ be the minimum ELB value among all unexecuted time-points: $\Lambda(\ell, \mathcal{T}_u) = \min\{ELB(X, \ell) \mid X \in \mathcal{T}_u\}$. Then for each $X \in \mathcal{T}_u$, the constraint $(X \geq \Lambda(\ell, \mathcal{T}_u), \ell)$ is entailed by constraints in $\mathcal{C}$.

Proof. Let Λ denote $\Lambda(\ell, \mathcal{T}_u)$, $ELB(X)$ denote $ELB(X, \ell)$, and $\mathcal{P}_u = \{p \in \mathcal{P} \mid P? \in \mathcal{T}_u\}$ be the as-yet-unobserved letters. By construction, for each $p \in \mathcal{P}_u$, $ELB(P?) \geq \Lambda$; so there must be a constraint, $(P? \geq \lambda_p, \alpha_p) \in \mathcal{C}$, for some $\lambda_p \geq \Lambda$ and $\alpha_p \in \mathcal{Q}^*$ such that $appl(\alpha_p, \ell)$. Now, if p appears in α_p, then qR_0 can remove it from α_p; thus, since $\mathcal{C}$ is closed under qR_0, it can be assumed that p does not appear in α_p.

Next, let r be any other letter in $\mathcal{P}_u$. Again, there must be a constraint, $(R? \geq \lambda_r, \alpha_r)$, for some $\lambda_r \geq \Lambda$, and $r \in \mathcal{Q}^*$ such that $appl(\alpha_r, \ell)$. Now, if p appears in α_r, then it can be removed using qR_3^*, generating $(R? \geq \lambda_{pr}, \alpha_{pr})$, where $\lambda_{pr} = \min\{\lambda_p, \lambda_r\} \geq \Lambda$, and $\alpha_{pr} = \alpha_p \star \alpha'_r$, where α'_r is obtained by removing any $\tilde{p}$ from α_r. Since $appl(\alpha_p, \ell)$ and $appl(\alpha_r, \ell)$, it follows that $appl(\alpha_{pr}, \ell)$. As before, since $\mathcal{C}$ is closed under qR_0 and qR_3^*, this constraint must be entailed by constraints in $\mathcal{C}$. Thus, we can assume that p does not appear in α_r. Continuing in this way, each observation time-point $T? \in \mathcal{T}_u$ must have a corresponding constraint $(T? \geq \lambda_t, \alpha_t)$ where $\lambda_t \geq \Lambda$ and α_t does not include p.

Once p has been removed from all such labels, then r can similarly be removed, and so on, until each $T? \in \mathcal{T}_u$ is seen to have a lower-bound constraint, $(T? \geq \lambda_t, \alpha_t)$, where $\lambda_t \geq \Lambda$ and α_t has no letters from $\mathcal{P}_u$. Thus, each α_t can only have letters that appear in ℓ; whence $appl(\alpha_t, \ell)$ implies that $\ell \models \alpha_t$. Finally, qR_3^* can be used to similarly process the labels from lower-bound constraints on all $X \in \mathcal{T}_u$, removing all occurrences of letters from $\mathcal{P}_u$. Thus, for each $X \in \mathcal{T}_u$, $(X \geq \Lambda, \ell)$ must be entailed by constraints in $\mathcal{C}$. □

Lemma 7. *Let $\mathcal{S}$ be any CSTN, s any scenario, and $\mathcal{S}(s)$ the corresponding STN projection. If P is a path from X to Y of length δ in $\mathcal{S}(s)$, then there is a corresponding path P' in $\mathcal{S}$ from X to Y whose length is δ and whose edges have labels that are consistent with s. In addition, if the edges in $\mathcal{S}$ are closed under the* LP *rule, then there is a single edge in $\mathcal{S}$ from X to Y of length δ whose label is consistent with s.*

Algorithm 2: EarliestFirstExecutionStrategy($\mathcal{S}$)

Input: $\mathcal{S} = \langle \mathcal{T}, \mathcal{P}, \mathcal{C}, \mathcal{OT}, \mathcal{O} \rangle$, a CSTN
$t := 0, d := 1, \ell := \boxdot$ // time, dependency position, CPS
$[\sigma(s)]_{\mathrm{Z}} := 0, \quad \mathcal{T}_u := \mathcal{T} \setminus \{\mathrm{Z}\}$ // execute Z at 0
while $(\mathcal{T}_u \neq \emptyset)$ **do**
  $t := \Lambda(\ell, \mathcal{T}_u)$ // as in Lemma 6
  $\chi := \{X \in \mathcal{T}_u \mid ELB(X, \ell) = t\}$ // as in Lemma 6
  foreach $(X \in \chi)$ **do**
    $[\sigma(s)]_X := t, \quad \mathcal{T}_u := \mathcal{T}_u \setminus \{X\}$ // execute X at t
    if $(X \in \mathcal{OT})$ **then** // record observation
      $[\sigma(s)]^{\pi}_X := d, \quad d := d + 1, \quad \ell := \ell \wedge s(x)$
return $\langle \sigma(s), s \rangle$ // $\ell = s$ at end

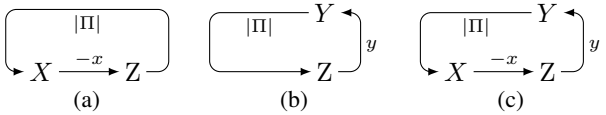

Figure 3: Loops discussed in proof of Theorem 1

Proof. Follows from Defn. 5 and the LP rule. □

Theorem 1. *Algorithm 1 is sound and complete for π-DC checking for CSTNs with rational weights.*

Proof. Soundness follows from Lemmas 2–4. Termination is guaranteed by the horizon constraints: for each $X \in \mathcal{T}$, $0 \leq X \leq h$. A finite number of rational weights can be put over a common denominator D, yielding, in effect, an integer domain. Because there are n^2 edges, each with at most 3^k different q-labels and D different numerical weights, the algorithm generates at most $n^2(3^k)D$ incremental updates.

Next, let $\mathcal{S}$ be any *fully propagated* CSTN that the algorithm says is π-DC. Then $\mathcal{S}$ must be closed under LP, qR_0 and qR^*_3.

Let σ be the earliest-first execution strategy (Algorithm 2). This strategy can be incrementally computed in real time as the execution of observation time-points incrementally reveals the scenario s. We must show that σ is viable and π-dynamic.

(I) *σ is π-dynamic.* Lemma 6 ensures that every unexecuted time-point is constrained to occur at or after $t = \Lambda(\ell, \mathcal{T}_u)$. Hence, the next round of ELB values cannot go below t. As a result, the next $\Lambda(\ell, \mathcal{T}_u)$ value cannot go below t. Thus, the strategy is well defined. By construction, each decision depends only on past observations, or concurrent observations with a lower d value. Thus, each execution decision depends only on the relevant π-history, as required by Defn. 15.

(II) *For any scenario s, the projection $\mathcal{S}(s)$ is consistent.* If $\mathcal{S}(s)$ contains a negative loop, then Lemma 7 implies there is a single-edge negative loop in $\mathcal{S}$ whose label is consistent with s, contradicting the algorithm's report that $\mathcal{S}$ is π-DC.

(III) *σ is valid.* Let s be any scenario, and $\mathcal{S}(s)$ the corresponding projection. Suppose that $\sigma(s)$ is *not* a solution to the STN $\mathcal{S}(s)$. For each X, let $x = [\sigma(s)]_X$. The corresponding execution constraints are $(\mathrm{Z} - X \leq x)$ and $(X - \mathrm{Z} \leq x)$ (i.e., $X = x$). Since $\sigma(s)$ is not a solution, inserting these constraints into $\mathcal{S}(s)$ must create a negative loop—call it L. Without loss of generality, L has only one occurrence of Z.

Case 1 (Fig. 3a). L consists of the lower-bound execution constraint $(\mathrm{Z} - X \leq -x)$ followed by a path Π from Z to X, where: (1) $x = \Lambda(\ell, \mathcal{T}_u) = ELB(X, \ell)$, where ℓ is the CPS when X was executed; (2) $|\Pi| < x$, and (3) the edges in Π are from $\mathcal{S}(s)$. By Lemma 7, there must be an edge in $\mathcal{S}$ from Z to X of length $|\Pi|$ whose label is consistent with s. Lemma 6 ensures that the constraint, $(\mathrm{Z} - X \leq -x, \ell)$, is entailed by constraints in $\mathcal{S}$. And ℓ is necessarily consistent with s. Applying the LP rule to these two edges would yield a single-edge negative loop in $\mathcal{S}$ with a consistent label, causing the algorithm to report that $\mathcal{S}$ was not π-DC, a contradiction.

Case 2 (Fig. 3b). L consists of the constraint $(Y - \mathrm{Z} \leq y)$ followed by a path Π from Y to Z, where: (1) $y = \Lambda(\ell, \mathcal{T}_u) = ELB(Y, \ell)$, where ℓ is the CPS when Y was executed; (2) the edges in Π are from $\mathcal{S}(s)$; and (3) $|\Pi| < -y$. By Lemma 7, there must be an edge in $\mathcal{S}$ from Y to Z of length $|\Pi|$ whose label is consistent with s. But then $ELB(Y, \ell)$ must satisfy $-|\Pi| \leq ELB(Y, \ell) = y$, which contradicts that $|\Pi| < -y$.

Case 3 (Fig. 3c). L has a path Π from Y to X with edges in $\mathcal{S}(s)$, a lower-bound (LB) edge $(\mathrm{Z} - X \leq -x)$, and an upper-bound edge $(Y - \mathrm{Z} \leq y)$. Here, $|\Pi| - x + y < 0$. By Lemma 7, $\mathcal{S}$ has an edge Π' of length $|\Pi|$ whose label is consistent with s. By Case 1, the LB constraint for X is entailed by constraints in $\mathcal{S}$. Thus, applying LP to Π' and the LB edge for X would have yielded an edge, $(\mathrm{Z} - Y \leq |\Pi| - x, \alpha)$ in $\mathcal{S}$, with α consistent with s. So $-|\Pi| + x$ must be a lower bound for Y. Hence, $ELB(Y, \ell_y) \geq -|\Pi| + x > y$, a contradiction. □

Corollary 1. *The six-rule IR-DC-checking algorithm [Hunsberger* et al.*, 2015] is also sound and complete for π-DC checking.*

Proof. The IR-DC-checking and π-DC-checking algorithms both use the qR_0 and qR^*_3 rules from Table 1. However, whereas the π-DC-checking algorithm uses the LP rule from Table 1, which only generates edges terminating at Z, the IR-DC-checking algorithm uses a more general version of that rule that can generate edges pointing at any time-point. On top of that, the IR-DC-checking algorithm uses three *more* rules, called qLP , R_0 and R^*_3. The qLP rule is a generalization of the LP rule that accommodates parent edges with inconsistent labels and, thus, can generate q-labeled edges; the R_0 and R^*_3 rules are analogous to the qR_0 and qR^*_3 rules, except that they can generate edges terminating at any time-point, not just Z, but they require the labels on the parent edges to be consistent and, thus, can only generate edges with consistent labels. From these observations, it follows that the propagations done by the IR-DC-checking algorithm form a superset of the propagations done by the π-DC-checking algorithm. Therefore, if a given CSTN passes the IR-DC-checking algorithm, then it must also pass the π-DC-checking algorithm.

It remains to show that if a given CSTN fails the IR-DC-checking algorithm, then it must also fail the π-DC-checking algorithm. In other words, it remains to show that the additional propagations done by the IR-DC-checking algorithm are sound with respect to the π-DC semantics. Due to space limitations, we leave this part of the proof to the reader. □

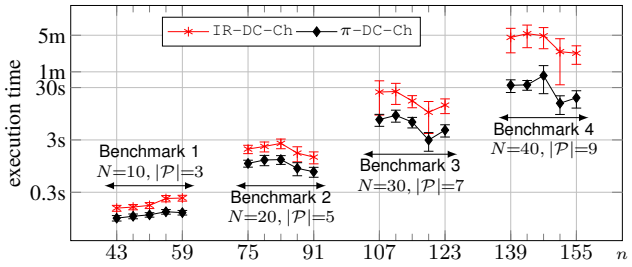

Figure 4: Execution time vs. number of time-points n

5 Empirical Evaluation

This section compares the performance of the π-DC-checking and IR-DC-checking algorithms. `π-DC-Ch` is our implementation of Algorithm 1; `IR-DC-Ch` is the freely available implementation of the IR-DC-checking algorithm [Posenato, 2017]. All algorithms and procedures were implemented in Java and executed on a JVM 8 in a Linux box with two AMD Opteron 4334 CPUs and 64GB of RAM.

Both implementations were tested on instances of the four benchmarks from Hunsberger and Posenato [2016]. Each benchmark has at least 60 DC and 60 non-DC CSTNs, obtained from random workflow schemata generated by the ATAPIS toolset [Lanz and Reichert, 2014]. 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 networks, this section focuses on the results for the DC networks.

Fig. 4 displays the average execution times for the two algorithms over all four benchmarks, each point representing the average execution time for instances of the given size. We extended the benchmarks, adding up to 50 DC instances, to generate tight 95% confidence intervals. The results demonstrate that the 3-rule `π-DC-Ch` algorithm is much faster than the 6-rule `IR-DC-Ch` algorithm. Moreover, the performance improvement increases as the instance size increases.

Regarding non-DC networks, we verified that the difference in performance of the two algorithms is statistically insignificant: for each network size, the confidence intervals for the average execution times of the two algorithms invariably overlap.

6 Conclusions

This paper began by showing that the IR-DC semantics for CSTNs is flawed, and that one of the six rules from the IR-DC-checking algorithm for CSTNs is *unsound* with respect to the IR-DC semantics. However, the major contribution of the paper was to present the simpler, three-rule π-DC-checking algorithm and prove that it is sound and complete with respect to the (unflawed) π-DC semantics from Cairo et al. [2016]. An empirical evaluation showed that the π-DC-checking algorithm is faster than the IR-DC-checking algorithm, which also happens to be sound and complete for the π-DC-checking problem.

References

[Cairo *et al.*, 2016] Massimo Cairo, Carlo Comin, and Romeo Rizzi. Instantaneous reaction-time in dynamic-consistency checking of conditional simple temporal networks. In *TIME 2016*, 2016.

[Cairo *et al.*, 2017] 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.

[Cimatti *et al.*, 2014] 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.

[Cimatti *et al.*, 2016] 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.

[Comin and Rizzi, 2015] 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.

[Dechter *et al.*, 1991] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.

[Hunsberger and Posenato, 2016] Luke Hunsberger and Roberto 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*, pages 175–183, 2016.

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

[Hunsberger *et al.*, 2015] 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. IEEE, 2015.

[Lanz and Reichert, 2014] Andreas Lanz and Manfred Reichert. Enabling time-aware process support with the ATAPIS Toolset. In *Proceedings of the BPM Demo Sessions 2014*, volume 1295, pages 41–45, 2014.

[Posenato, 2017] Roberto Posenato. A CSTN(U) consistency check algorithm implementation in Java. Version 1.20. http://profs.scienze.univr.it/∼posenato/software/cstnu, November 2017.

[Tsamardinos *et al.*, 2003] Ioannis Tsamardinos, Thierry Vidal, and Martha E. Pollack. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints*, 8:365–388, 2003.