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 ϵ > 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 ϵ = 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
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?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. There-
fore, 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 P of propositional letters: a
label is a (possibly empty) conjunction of (positive or negative)
literals from P. The empty label is notated []; for any label ℓ, and any p ∈ P, if ℓ |= p or ℓ |= ¬p, then we say that p
appears in ℓ; for any labels ℓ1 and ℓ2, if ℓ1 |= ℓ2, then ℓ1
is said to entail ℓ2; if ℓ1 ∧ ℓ2 is satisfiable, then ℓ1 and ℓ2
are called consistent; and P* denotes the set of all consistent
labels whose literals are drawn from P.

Definition 2 (CSTN). A Conditional Simple Temporal Net-
work (CSTN) is a tuple, ⟨T, P, C, OT, O⟩, where:

- T is a finite set of real-valued time-points (i.e., variables);
- P is a finite set of propositional letters (or propositions);
- C is a set of labeled constraints, each having the form,
  ⟨Y − X ≤ δ, ℓ⟩, where X, Y ∈ T, δ ∈ R, and ℓ ∈ P*;
- OT ⊆ T is a set of observation time-points (OTPs); and
- O: P → OT is a bijection that associates a unique
  observation time-point to each propositional label.

In a CSTN graph, the observation time-point for p (i.e.,
O(p)) may be denoted by P?; and each labeled constraint,
⟨Y − X ≤ δ, ℓ⟩, is represented by an arrow from X to Y,
X\rightarrow{\delta,\ell}Y, annotated by the labeled value ⟨δ, ℓ⟩. X and Y
may participate in multiple constraints of the form, ⟨Y − X ≤ δ, ℓ⟩,
the edge from X to Y may have multiple labeled values of the form, ⟨δ, ℓ⟩.

Definition 3 (Scenario). A function, s: P → {true, false},
that assigns a truth value to each p ∈ P is called a scenario.
For any label ℓ ∈ P*, the truth value of ℓ determined by s
is denoted by s(ℓ). I denotes the set of all scenarios over P.

Definition 4 (Schedule). A schedule for a set of time-points
T is a mapping, ψ: T → R. The set of all schedules for any
subset of 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 S = ⟨T, P, C, OT, O⟩ be any
CSTN, and s any scenario over P. The projection of S onto
s—notated S(s)—is the STN, ⟨T, C+⟩, where:

C+ = {⟨Y − X ≤ δ⟩ | [∃ℓ(⟨Y − X ≤ δ, ℓ⟩ ∈ C ∧ s(ℓ) = true)}

Definition 6 (Execution Strategy). An execution strategy for a
CSTN S = ⟨T, P, C, OT, O⟩ is a mapping, σ: I → Ψ, from
scenarios to schedules. The execution time for the time-point
X in the schedule σ(s) is denoted by |σ(s)|X. An execution
strategy σ for a CSTN S is viable if for each scenario s, the
schedule σ(s) is a solution to the projection 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 obser-
vations 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 S = ⟨T, P, C, OT, O⟩ be any
CSTN, s any scenario, σ any execution strategy for S, and t
any real number. The history of t in the scenario s, for the
strategy σ—notated Hist(t, s, σ)—is the set of observations
made before time t according to the schedule σ(s):

Hist(t, s, σ) = {(p, s(p)) | P? ∈ OT and |σ(s)|p? < t}

Definition 8 (IR-Dynamic Strategy). An execution strategy
σ for a CSTN S = ⟨T, P, C, OT, O⟩ is called IR-dynamic if
for any scenarios s1 and s2, and any time-point X ∈ T:

let: t = [σ(s1)]X

if: Hist(t, s1, σ) = Hist(t, s2, σ)

then: [σ(s2)]X = t

unless: |σ(s1)⟩Q? = |σ(s2)⟩Q? = t and, for some
Q? ∈ OT, Q? ≠ X and s1(q) ≠ s2(q).
Yet, the strategy, Thus, there is no time-point that can be executed first, at or after 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, $\sigma'$, defined below, is viable and IR-dynamic.

![Figure 2: An absurd CSTN that is nonetheless IR-DC](image)

<table>
<thead>
<tr>
<th>LP($X, u, \alpha, Y, \beta$): $X \xrightarrow{u, \alpha} Y \xrightarrow{v, \beta} Z$</th>
<th>$qR_0(P?, w, \alpha, p)$: $P? \xrightarrow{(w, \alpha)} \xrightarrow{(w, \alpha)} Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$qR_1(P?, w, \alpha, v, \beta, p, Y)$: $P? \xrightarrow{(w, \alpha, v, \beta)} \xrightarrow{(w, \alpha, v, \beta)} Y$</td>
<td>$X, Y \in T; P? \in OT$; and $Z$ is fixed at 0. LP applies if $\alpha \beta \in P$; $qR_0$ and $qR_1$ apply if $w &lt; 0$. In $qR_0$ and $qR_1$, $p \in {p, \neg p}$; $p$ does not appear in $\alpha$ or $\beta$, and $m = \max{v, w}$.</td>
</tr>
</tbody>
</table>

Table 1: The LP, $qR_0$ and $qR_1$ propagation rules for CSTNs (above) and instances of their use (below)

Thus, if an IR-dynamic strategy $\sigma$ 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 $\sigma$ 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.

The $\sigma'$ 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 $p \ast p = \varnothing$.

Definition 10 (*). The operator, $\ast: Q^* \times Q^* \rightarrow Q^*$, is defined thusly. First, for any $p \in P$, $p \ast p = p$ and $\neg p \ast \neg p = \varnothing$. Next, for any $p_1, p_2 \in \{p, \neg p\}$ for which $p_1 \neq p_2, p_1 \ast p_2 = \varnothing$. Finally, for any q-labels $\ell_1, \ell_2 \in Q^*$, $\ell_1 \ast \ell_2 \in Q^*$ denotes the conjunction obtained by applying $\ast$ in pairwise fashion to matching literals from $\ell_1$ and $\ell_2$, and conjoining any unmatched literals.

For example: $(p \ast q(\tau)t) \ast (q \ast s) = p(q)(\tau)t \ast s$.

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 $\sigma'$. 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 $(B? \geq 1, b(c))$. Next, applying $qR_3$ to this constraint yields: $(B? \geq 1, b(c))$ (i.e., $B$ must be at least 1 as long as $c$ is unknown). But this constraint is not satisfied by $\sigma'$. For example, in the scenario $\neg a \ast b$, $\sigma'$ 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 $qR_3$ rules will generate the constraints, $(A? \geq 1, a(c)), (B? \geq 1, b(c))$ and $(C? \geq 1, a(c), i.e., A?\neq B?$ 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 $\pi$-Dynamic Strategies and $\pi$-DC

This section summarizes the $\pi$-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 $\pi$-DC semantics.

Definition 11 (Order of dependence). For any scenario $s$, and ordering $(P_1, \ldots, P_k)$ of observation time-points, where $k = |OT|$, an order of dependence is a permutation $\pi$ over $\{1, 2, \ldots, k\}$; and for each $P_j \in OT, \pi(P_j) \in \{1, 2, \ldots, k\}$ denotes the integer position of $P_j$ in that order. For any non-observation time-point $X$, we set $\pi(X) = \infty$. Finally, $\Pi_k$ denotes the set of all permutations over $\{1, 2, \ldots, k\}$.

$^1(B? \geq 1)$ is an abbreviation for $(Z - B? \leq -1)$. 
Definition 12 (π-Execution Strategy). Let σ be any π-execution strategy for a CSTN $S = \langle T, \mathcal{P}, \mathcal{C}, \mathcal{O}_T, \mathcal{O}_\sigma \rangle$, let $k = |\mathcal{O}_T|$. A π-execution strategy for $S$ is a mapping, $\pi : \mathcal{T} \times \mathcal{P} \times \mathcal{I}_k$, such that for each scenario $s$, $\pi(s)$ is a pair $(\psi, \pi)$ where $\psi : T \rightarrow \mathcal{P}$ 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 $P \in \mathcal{O}_T$, $[\sigma(s)]_{P^T}$ denotes the position of $P$ in the order of dependence (i.e., $\pi(P^T)$). Finally, a π-strategy must be coherent: for any scenario $s$, and any $P^T, Q^T \in \mathcal{O}_T$, $[\sigma(s)]_{P^T} < [\sigma(s)]_{Q^T}$ implies $[\sigma(s)]_{P^T} \leq [\sigma(s)]_{Q^T}$ (i.e., if $s$ schedules $P^T$ before $Q^T$, then it orders $P^T$ before $Q^T$).

Definition 13 (Viability). The π-execution strategy $\sigma$ is called viable for the CSTN $S$ if for each scenario $s$, the schedule $\psi$ is a solution to the projection $\mathcal{S}(s)$, where $\mathcal{S}(s) = (\psi, \pi)$.

Definition 14 (π-History). Let $\beta$ be any π-execution strategy for some CSTN $S = \langle T, \mathcal{P}, \mathcal{C}, \mathcal{O}_T, \mathcal{O}_\sigma \rangle$, $s$ any scenario, $t$ any real number, and $d \in \{1, 2, \ldots, |\mathcal{O}_T|\} \cup \{\infty\}$ any integer position (or infinity). The π-history of $(t, d)$ for the scenario $s$ and strategy $\sigma$—denoted by $\pi \mathcal{H}(t, d, s, \sigma)$—is the set
\[
\{(p, s(p)) | P^T \in \mathcal{O}_T, \mathcal{T}(s(p)) \leq t, \pi(P^T) < d\}.
\]
The π-history specifies the truth value of each $p \in \mathcal{P}$ that is observed before $t$ in the schedule $\psi$, or at $t$ if the corresponding $P^T$ is ordered before position $d$ by the permutation $\pi$.

The following definition is equivalent to that given by 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 $\sigma$, for a CSTN $S$, is called π-dynamic if for every pair of scenarios, $s_1$ and $s_2$, and every time-point $X \in T$: let
\[
t = [\sigma(s_1)]_X, \text{ and } d = [\sigma(s_1)]_X,
\]
if $\pi \mathcal{H}(t, d, s_1, \sigma) = \pi \mathcal{H}(t, d, s_2, \sigma)$
then $[\sigma(s_2)]_X = t$ and $[\sigma(s_2)]_X = d$.
Thus, if $\sigma$ executes $X$ at time $t$ and position $d$ in scenario $s_1$, and the histories, $\pi \mathcal{H}(t, d, s_1, \sigma)$ and $\pi \mathcal{H}(t, d, s_2, \sigma)$, are the same, then $\sigma$ 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, $S$, is π-dynamically consistent (π-DC) if there exists a π-execution strategy for $S$ 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 $\sigma$ is a valid and π-dynamic strategy for a CSTN $S$ that includes the constraints $(W \leq X, u, \alpha)$ and $(Y \leq W, v, \beta)$, where $\alpha, \beta \in P^T$, then $\sigma$ also satisfies the constraint $(Y \leq X, u + v, \alpha + \beta)$ (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 $\sigma$ is valid and $s \models \alpha$, $[\sigma(s)]_W - [\sigma(s)]_X \leq u$. Similarly, since $\sigma$ 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 $\sigma$ observes $p$ (or executes $P^T$) π-before executing $Y$ if $[\sigma(s)]_{P^T} \leq [\sigma(s)]_Y$ and $[\sigma(s)]_{P^T} < [\sigma(s)]_Y$.

Intuitively, if $\sigma$ observes $p$ π-before $Y$, then $\sigma$’s decision to execute $Y$ can depend on the value of $p$. In addition, note that if $P^T$ and $Q^T$ are distinct, then a coherent strategy $\sigma$ observes $p$ π-before $Q^T$ if and only if $[\sigma(s)]_{P^T} < [\sigma(s)]_{Q^T}$.

Since the qR$_0$ and qR$_1$ 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 et al. [2015].

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

Lemma 3 (Soundness of qR$_0$). If $\sigma$ is a valid and π-dynamic strategy that satisfies $(P^T \geq \delta, \alpha)$, where $\alpha \in Q^*$ and $\pi \in \{p, \neg p, ?p\}$, then $\sigma$ also satisfies $(P^T \geq \delta, \alpha)$.

Proof. If $\sigma$ satisfies $(P^T \geq \delta, \alpha)$, for some $\pi \in \{p, \neg p, ?p\}$ and $\pi \in \{p, \neg p, ?p\}$, then $\alpha$ satisfies $(P^T \geq \delta, \alpha)$.

Lemma 4 (Soundness of the qR$_1^2$ rule). Let $\sigma$ be any viable and π-dynamic strategy for a CSTN $S$. If $\sigma$ satisfies the constraints $(P^T \geq g, \alpha)$ and $(Y \geq h, \beta)$, where $\alpha, \beta \in Q^*$, then $\sigma$ also satisfies $(P^T \geq g, \alpha)$.($\sigma$).

Proof. Suppose the conditions of the lemma hold, but $\sigma$ does not satisfy $(\geq X, \alpha, \beta)$. Then for some scenario $s$, the negation of Defn. 18 holds: (1) $[\sigma(s)]_Y \geq \min\{g, h\}$; and (2) for each $a \in \{a, -a, ?a\}$ appearing in $\alpha$ such that $\sigma$ observes a π-before $Y$, $s \models a$. Since $\sigma$ satisfies $(Y \geq h, \beta)$, one of the following holds: (1) $[\sigma(s)]_Y \geq h \geq x$; or (2) some $\pi \in \{p, \neg p, ?p\}$ appears in $\beta$ such that $\sigma$ observes $b$ π-before $Y$ and $s \not\models b$. Since (1) contradicts (1), (2) must hold for some $\pi \in \beta$. If $b \models b$ in $Y$ and $b \models \alpha \beta$, then $s \not\models b$ contradicts (2). If $\pi \models b \models \beta$ and $b \models \alpha \beta$, then $s \not\models b$ again contradicts (2). Similar remarks apply to the cases.

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: $0 \leq X \leq h = M_n$, where $M$ is the maximum absolute value of any negative

Substitutions from Table 1: $g = -w; h = -v; and x = -m.$
weight in the CSTN, and \( n = |T| \). 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), \( qR^*_0 \) (Lines 8–9), and LP (Lines 10–11). Only constraints that are not entailed by constraints already in \( C \) 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 \( \pi\)-DC in the first case; \( \pi\)-DC in the second.

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

4.2 Completeness of \( \pi\)-DC Checking

This section proves that the \( \pi\)-DC-checking algorithm is sound and complete with respect to the \( \pi\)-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 \( \ell \in P^* \) that is equivalent to the \( \pi\)-history at that point. For convenience, such a label is called a current partial scenario (CPS). A q-label \( \alpha \in Q^* \) is called applicable in a given CPS \( \ell \), if the observations in \( \ell \) are consistent with \( \alpha \).

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

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

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

Proof. Suppose \( \sigma(s)|_X < \delta \). Since \( \sigma \) satisfies \((X \geq \delta, \alpha)\), there must be some \( \bar{a} \in \{a, \neg a\} \) appearing in \( \alpha \) such that \( \sigma \) observes \( a \)-before \( X \), and \( s \neq \bar{a} \). Since \( \sigma \) observes a \( \pi\)-before \( X \), then either \( a \) or \( \neg a \) appears in the CPS \( \ell \). Now, if \( \bar{a} = a \), then \( a \in \alpha \), but \( \neg a \) implies that \( \neg a \in \ell \); and, hence, \( \neg \text{appl}(\alpha, \ell) \). The case, \( \bar{a} = \neg a \), is similar. And if \( \bar{a} = a \), then \( \neg a \in \alpha \) and \( a \) appearing in \( \ell \) imply that \( \neg \text{appl}(\alpha, \ell) \).

Lemma 6 (Spreading Lemma). Let \( S \) be any CSTN whose set of constraints \( C \) is closed under the \( qR_0 \) and \( qR^*_0 \) rules. Let \( \ell \in P^* \) be any consistent label, representing a possible CPS. Let \( T_x = \{ P \mid \text{ appl appears in } \ell \} \) be the observation time-points corresponding to the observations in \( \ell \). Let \( T_x \) be any set of time-points such that \( T_x \subseteq T_x \subseteq T_x \). \( T_x \) represents the time-points that have been executed so far. And let \( T_u = T \setminus T_x \) be the as-yet-unexecuted time-points. For each \( X \in T_u \), let ELB\((X, \ell)\) be its effective lower bound:

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

And let \( \Lambda(\ell, T_u) \) be the minimum ELB value among all unexecuted time-points: \( \Lambda(\ell, T_u) = \min\{\text{ELB}(X, \ell) \mid X \in T_u \} \).

Then for each \( X \in T_u \), the constraint \( (X \geq \Lambda(\ell, T_u), \ell) \) is entailed by constraints in \( C \).

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

Next, let \( r \) be any other letter in \( P_u \). Again, there must be a constraint, \( (P \geq \lambda_r, \alpha_r) \), for some \( \lambda_r \geq \Lambda \), and \( r \in Q^* \) such that \( \text{appl}(\alpha_r, \ell) \). Now, if \( p \) appears in \( \alpha_r \), then it can be removed using \( qR^*_0 \), generating \( (P \geq \lambda_p \wedge \alpha_p, \alpha_p') \), where \( \alpha_p' = \alpha_p \setminus \alpha_r \). Since \( \text{appl}(\alpha_p', \ell) \), \( \alpha_p' \) is obtained by removing any \( \alpha_r \). Since \( \text{appl}(\alpha_p, \ell) \) and \( \text{appl}(\alpha_r, \ell) \), it follows that \( \text{appl}(\alpha_p \cup \alpha_r, \ell) \). As before, since \( C \) is closed under \( qR_0 \) and \( qR^*_0 \), this constraint must be entailed by constraints in \( C \). Thus, we can assume that \( p \) does not appear in \( \alpha_r \). Continuing in this way, each observation time-point \( T \in T_u \) must have a corresponding constraint \( (T \geq \lambda_t, \alpha_t) \) where \( \lambda_t \geq \Lambda \) and \( \alpha_t \) does not include \( p \).

Once \( p \) has been removed from all such labels, then \( r \) can similarly be removed, and so on, until each \( T \in T_u \) is seen to have a lower-bound constraint, \( (T \geq \lambda_t \wedge \alpha_t) \), where \( \lambda_t \geq \Lambda \) and \( \alpha_t \) has no letters from \( P_u \). Thus, each \( \alpha_t \) can only have letters that appear in \( \ell \); whence \( \text{appl}(\alpha_t, \ell) \) implies that \( \ell = \alpha_t \). Finally, \( qR_0 \) can be used to similarly process the labels from lower-bound constraints on all \( X \in T_u \), removing all occurrences of letters from \( P_u \). Thus, for each \( X \in T_u \), \( (X \geq \Lambda, \ell) \) must be entailed by constraints in \( C \).
Algorithm 2: EarliestFirstExecutionStrategy(S)

Input: $S = (\Pi, P, C, OT, C)$, a CSTN
$t := 0$, $d := 1$, $\ell := \emptyset$ // time, dependency position, CPS
$[\sigma(s)]_Z := 0$, $\mathcal{T}_0 := T \setminus \{Z\}$ // execute Z at 0
while ($\mathcal{T}_0 \neq \emptyset$) do
$\ell := \Lambda(\ell, \mathcal{T}_0)$ // as in Lemma 6
$\chi := \{X \in \mathcal{T}_0 \mid ELB(X, \ell) = t\}$ // as in Lemma 6
for each $(X \in \chi)$ do
$[\sigma(s)]_X := t$, $\mathcal{T}_0 := \mathcal{T}_0 \setminus \{X\}$ // execute X at t
if $(X \not\in OT)$ then
$[\sigma(s)]_X := d$, $d := d + 1$, $\ell := \ell \land s(x)$ // record observation
end
end
return $[\sigma(s), s)$ // $\ell = s$ 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 $n^3$ edges, each with at most $3^k$ different q-labels and $D$ different numerical weights, the algorithm generates at most $n^3 \cdot 3^k$ $D$ incremental updates.

Next, let $S$ be any fully propagated CSTN that the algorithm says is π-DC. Then $S$ must be closed under LP, qR$_0$ and qR$_1^*$.

Let $\sigma$ 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 $\sigma$ is viable and π-dynamic.

(I) $\sigma$ is π-dynamic. Lemma 6 ensures that every unexecuted time-point is constrained to occur at or after $t = \Lambda(\ell, \mathcal{T}_0)$. Hence, the next round of ELB values cannot go below $t$. As a result, the next $\Lambda(\ell, \mathcal{T}_0)$ 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 $S(s)$ is consistent. If $S(s)$ contains a negative loop, then Lemma 7 implies there is a single-edge negative loop in $S$ whose label is consistent with $s$, contradicting the algorithm’s report that $S$ is π-DC.

(III) $\sigma$ is valid. Let $s$ be any scenario, and $S(s)$ the corresponding projection. Suppose that $\sigma(s)$ is not a solution to the STN $S(s)$. For each $X$, let $x = [\sigma(s)]_X$. The corresponding execution constraints are $(Z - X \leq x)$ and $(X - Z \leq x)$ (i.e., $X = x$). Since $\sigma(s)$ is not a solution, inserting these constraints into $S(s)$ 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 $(Z - X \leq x)$ followed by a path $\Pi$ from $Z$ to $X$, where: (1) $x = \Lambda(\ell, \mathcal{T}_0) = ELB(X, \ell)$, where $\ell$ is the CPS when $X$ was executed; (2) $|\Pi| < x$, and (3) the edges in $\Pi$ are from $S(s)$. By Lemma 7, there must be an edge in $S$ from $Z$ to $X$ of length $|\Pi|$ whose label is consistent with $s$. Lemma 6 ensures that the constraint, $(Z - X \leq -x, \ell)$, is entailed by constraints in $S$. And $\ell$ is necessarily consistent with $s$. Applying the LP rule to these two edges would yield a single-edge negative loop in $S$ with a consistent label, causing the algorithm to report that $S$ was not π-DC, a contradiction.

Case 2 (Fig. 3b). $L$ consists of the constraint $(Y - Z \leq y)$ followed by a path $\Pi$ from $Y$ to $Z$, where: (1) $y = \Lambda(\ell, \mathcal{T}_0) = ELB(Y, \ell)$, where $\ell$ is the CPS when $Y$ was executed; (2) the edges in $\Pi$ are from $S(s)$; and (3) $|\Pi| < -y$. By Lemma 7, there must be an edge in $S$ 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 $\Pi$ from $Y$ to $X$ with edges in $S(s)$, a lower-bound (LB) edge $(Z - X \leq -x)$, and an upper-bound edge $(Y - Z \leq y)$. Here, $|\Pi| - x + y < 0$. By Lemma 7, $S$ has an edge $\Pi'$ of length $|\Pi|$ whose label is consistent with $s$. By Case 1, the LB constraint for $X$ is entailed by constraints in $S$. Thus, applying LP to $\Pi'$ and the LB edge for $X$ would have yielded an edge, $(Z - Y \leq |\Pi| - x, \alpha)$ in $S$, with $\alpha$ 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$_1^*$ 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 $R_1^*$. 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 $R_1^*$ rules are analogous to the qR$_0$ and qR$_1^*$ 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.

5 Empirical Evaluation

This section compares the performance of the \( \pi \)-DC-checking and IR-DC-checking algorithms. \( \pi \text{-DC-Ch} \) is our implementation of Algorithm 1; \( \pi \text{-DC-Ch} \) 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 activities (\( N \)) and observations (\( |P| \)) 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 \( \pi \text{-DC-Ch} \) algorithm is much faster than the 6-rule \( \pi \text{-DC-Ch} \) 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 \( \pi \)-DC-checking algorithm and prove that it is sound and complete with respect to the (unflawed) \( \pi \)-DC semantics from Cairo et al. [2016]. An empirical evaluation showed that the \( \pi \)-DC-checking algorithm is faster than the IR-DC-checking algorithm, which also happens to be sound and complete for the \( \pi \)-DC-checking problem.

References


