Checking the Dynamic Consistency of Conditional Simple Temporal Networks with Bounded Reaction Times

Luke Hunsberger
Vassar College
Poughkeepsie, NY USA

Roberto Posenato
Computer Science Department
University of Verona, Verona, Italy

Abstract

A Conditional Simple Temporal Network (CSTN) includes time-points, temporal constraints, and *observation time-points,* whose execution yields information during run-time. Time-points and constraints in a CSTN may only apply in certain scenarios. A CSTN is dynamically consistent (DC) if it has a strategy for executing its time-points such that all relevant constraints will be satisfied no matter how the observations turn out. A dynamic strategy can react to observations in real time, but only after arbitrarily small, but positive delays.

Recent work introduced a more realistic ϵ-DC property which, for a fixed $\epsilon > 0$, requires all reaction times to be bounded below by ϵ. That work presented an exponential algorithm for checking the ϵ-DC property by translating an exponential number of component networks into a Hyper Temporal Network. But it has not yet been implemented or empirically evaluated.

This paper begins by presenting an alternative, equivalent semantics for ϵ-dynamic consistency. It then presents a sound-and-complete ϵ-DC-checking algorithm based on the propagation of labeled constraints. Finally, it presents an empirical evaluation of the new algorithm, the first empirical evaluation of any ϵ-DC-checking algorithm in the literature.

Overview

A Conditional Simple Temporal Network (CSTN) is a data structure for representing and 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. Each condition in a CSTN is represented by a propositional letter whose truth value is not controlled, but instead *observed* in real time. Just as the performance of a blood-test action by a doctor might generate 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 obtained from past observations. The Conditional Simple Temporal Problem (CSTP) is that of determining whether a given CSTN admits a dynamic execution strategy that can guarantee the satisfaction of all relevant constraints no matter which outcomes are observed at run-time. If such a strategy exists, the CSTN is said to be dynamically consistent (DC). In other words, the CSTP is the DC-checking problem for CSTNs.

To our knowledge, prior approaches to solving the CSTP have resulted in exponential algorithms that either have not yet been implemented (Tsamardinos, Vidal, and Pollack 2003) or have only been successful on extremely small instances (Cimatti et al. 2014).

Recently, Hunsberger, Posenato and Combi (2015)—hereinafter HPC-15—presented a sound-and-complete DC-checking algorithm for CSTNs based on the propagation of labeled constraints. It employs a new kind of propositional literal of the form $?p$. A constraint labeled by $?p$ is only required to hold as long as the value of p is unknown, an important distinction for an execution strategy that can react to observations. A preliminary empirical evaluation demonstrated the practicality of their DC-checking algorithm.

Comin and Rizzi (2015)—hereinafter CR-15—defined *ϵ-dynamic consistency* (ϵ-DC), which specifies a fixed lower-bound on reaction times. For any $\epsilon > 0$, the decisions made by an ϵ-dynamic execution strategy can only depend on observations made at least ϵ in the past. They also presented an ϵ-DC-checking algorithm that begins by making exponentially many copies of the original CSTN, and then combining them into a single *Hyper Temporal Network.* Their approach has not yet been implemented or empirically evaluated.

This paper begins by providing an alternative, equivalent semantics for the ϵ-DC property. It then presents a new ϵ-DC-checking algorithm that extends the approach used by HPC-15. The paper proves that the new algorithm is sound and complete, and presents an empirical evaluation to demonstrate its practicality, the first empirical evaluation of any ϵ-DC-checking algorithm in the literature.

Background

Dechter *et al.* (1991) 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. The *Simple Temporal Problem* (STP) is that of determining whether an STN is consistent (i.e., has a solution).

Tsamardinos *et al.* (2003) augmented STNs to include

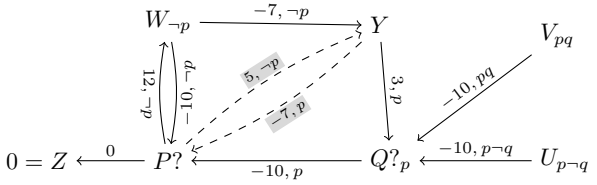

Figure 1: A sample CSTN

time-points and temporal constraints that apply only in certain scenarios, where each scenario is represented by a conjunction of propositional literals. In this paper, conjunctions such as $p \wedge \neg q \wedge r$ are notated as $p\neg qr$. Fig. 1 shows a sample CSTN in its graphical form, where the nodes represent time-points, and the directed edges represent binary difference constraints. $P?$ and $Q?$ are observation time-points whose execution generates truth values for p and q, respectively. In this example, $Q?$ is only executed if p happens to be $true$; thus, $Q?$ is labeled by p. Similarly, the edge from $Q?$ to $P?$, which represents the constraint, $P? - Q? \leq -10$ (i.e., $Q? \geq P? + 10$), is labeled by p. Similarly, the edge from U to $Q?$ is labeled by $p\neg q$, indicating that it applies only in scenarios where p is $true$ and q is $false$. The dashed edges with shaded labels are generated by the HPC-15 DC-checking algorithm, about which more will be said later.

Defns. 1-15, below, are drawn from HPC-15.

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 $\ell_1 \models \ell_2$ (i.e., if ℓ_1 contains all of the literals in ℓ_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.*
- 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, $\langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, \mathcal{P} \rangle$, where:

- $\mathcal{P}$ is a finite set of propositional letters (or propositions);
- $\mathcal{T}$ is a finite set of real-valued time-points (i.e., variables);
- $\mathcal{C}$ 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 $\ell \in \mathcal{P}^*$;
- $L : \mathcal{T} \rightarrow \mathcal{P}^*$ is a function assigning labels to time-points;
- $\mathcal{OT} \subseteq \mathcal{T}$ 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 (Honest Label)**.** A label ℓ in a CSTN, whether on a time-point or constraint, is called *honest* if for each q that appears in ℓ, ℓ entails $L(Q?)$ (i.e., ℓ contains all literals from the label of the observation time-point for q).

Definition 4 (WD_1: Label coherence)**.** A CSTN has *coherent labels* if for each labeled constraint, $(Y - X \leq \delta, \ell)$, the label ℓ is satisfiable and entails $L(X) \wedge L(Y)$.

Definition 5 ($\text{WD}_{2.1}$: Time-point Label Honesty)**.** A CSTN holds property $\text{WD}_{2.1}$ if its time-points all have honest labels.

Definition 6 ($\text{WD}_{2.2}$)**.** A CSTN holds property $\text{WD}_{2.2}$ if for each time-point T, and each propositional letter p appearing in $L(T)$, a constraint of the form $(P? - T \leq -e, L(T))$ (i.e., $(T \geq P? + e, L(T))$) is included in $\mathcal{C}$ for some $e > 0$.

Definition 7 (WD_3: Constraint Label Honesty)**.** A CSTN holds property WD_3 if its constraints all have honest labels.

A *well defined* CSTN is one for which WD_1, $\text{WD}_{2.1}$, $\text{WD}_{2.2}$ and WD_3 hold. The CSTN in Fig. 1 is well defined.

Definition 8 (Child nodes/literals)**.** If p appears in the label of an OTP $Q?$, then $Q?$ is called a *child* of $P?$; and both q and $\neg q$ are *children* of p. A similar definition applies to $\neg p$.

In a well-defined CSTN, such as the one in Fig. 1, if $Q?$ is a child of $P?$, then $Q?$ is constrained to occur after $P?$. In addition, if a child of p appears in an honest label ℓ, then p must also appear in ℓ. Conversely, if p does *not* appear in ℓ, then none of p's children can appear in ℓ either.

The dynamic consistency of CSTNs

The truth values of propositions in a CSTN are not known in advance. But a *dynamic strategy* for executing the time-points in a CSTN is allowed to react to observations–after a positive delay. A *viable* and *dynamic* execution strategy is a strategy that guarantees that all relevant constraints will be satisfied no matter which scenario is incrementally revealed. A CSTN for which such a strategy exists is called *dynamically consistent.*

Definition 9 (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 10 (Schedule)**.** A *schedule* for a set of time-points $\mathcal{T}$ is a mapping, $\psi : \mathcal{T} \rightarrow \mathbb{R}$, that assigns a real number to each time-point in $\mathcal{T}$. The set of all schedules *for any subset of* $\mathcal{T}$ is denoted by Ψ.

The projection of a CSTN, $\mathcal{S}$, onto a scenario, s, is the STN obtained by collecting all time-points and constraints from $\mathcal{S}$ whose labels are true under s (i.e., the time-points that must be executed and the constraints that must be satisfied).

Definition 11 (Projection)**.** Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, \mathcal{P} \rangle$ be any CSTN, and s any scenario over $\mathcal{P}$. The *projection* of $\mathcal{S}$ onto s—notated $\mathcal{S}(s)$—is the STN, $(\mathcal{T}_s^+, \mathcal{C}_s^+)$, where:

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

Definition 12 (Execution Strategy)**.** An *execution strategy* for a CSTN $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, \mathcal{P} \rangle$ is a mapping, $\sigma : \mathcal{I} \rightarrow \Psi$, such that for each scenario $s \in \mathcal{I}$, the domain of $\sigma(s)$ is $\mathcal{T}_s^+$ (cf. Defn. 11). If, in addition, for each scenario s, the schedule $\sigma(s)$ is a solution to the projection $\mathcal{S}(s)$, then σ is called *viable.* In any case, the execution time for the time-point X in the schedule $\sigma(s)$ is denoted by $[\sigma(s)]_X$.

LP:	$X \xrightarrow{\langle u,\alpha\rangle} W \xrightarrow{\langle v,\beta\rangle} Y$ $\xrightarrow[\langle u+v,\alpha\beta\rangle]{}$	
$\mathbf{R_0}$**:**	$P? \xrightarrow[\langle w,\alpha'\rangle]{\langle w,\alpha\rho\rangle} X$	if $w <^* 0, \rho \in \{p, \neg p\}$
$\mathbf{R_3^*}$**:**	$P? \xrightarrow{\langle w,\alpha\beta\rangle} X \xleftarrow[\langle \max\{v,w\},\alpha\beta\gamma'\rangle]{\langle v,\beta\gamma\rho\rangle} Y$	if $w \leq 0, \rho \in \{p, \neg p\}$
qLP :	$X \xrightarrow{\langle u,\alpha\rangle} W \xrightarrow{\langle v,\beta\rangle} \mathrm{Z}$ $\xrightarrow[\langle u+v,(\alpha\star\beta)^\dagger\rangle]{}$	if $u <^* 0, v < 0$
$\mathbf{qR_0}$**:**	$P? \xrightarrow[\langle w,\beta^\dagger\rangle]{\langle w,\beta\tilde{p}\theta\rangle} \mathrm{Z}$	if $w < 0$
$\mathbf{qR_3^*}$**:**	$P? \xrightarrow{\langle w,\gamma\rangle} \mathrm{Z} \xleftarrow[\langle \max\{v,w\},(\gamma\star\beta)^\dagger\rangle]{\langle v,\beta\tilde{p}\theta\rangle} Y$	if $w < 0$

In each rule, pre-existing edges are unshaded; generated edges are shaded. Pre-existing edges are presumed to have honest and coherent labels. For the first three rules, pre-existing edges are presumed to have satisfiable labels, and new edges are generated only if their labels are satisfiable. The $<^*$ operator is $\leq$ for standard DC semantics, $<$ for DC semantics that allows instantaneous reactions. For R_0, ρ must not appear in α or $L(X)$, and α' is obtained by removing any children of ρ from α. For R_3^*, α, β and γ must not share any letters, ρ must not appear in $\alpha, \beta, \gamma, L(X)$ or $L(Y)$, and γ' is obtained by removing children of ρ from γ. In the "q" rules, $\alpha \in \mathcal{P}^*$; $\beta, \gamma, \theta \in \mathcal{Q}^*$; $\tilde{p} \in \{p, \neg p, ?p\}$; and expressions such as $\ell^\dagger$ represent the q-label obtained from ℓ by removing the children of any q-literals that appear in ℓ. In qR_0 and qR_3^*, β contains no children of $\tilde{p}$, and θ *only* contains children of $\tilde{p}$. In qR_3^*, γ does not contain $\tilde{p}$ or any of its children.

Table 1: HPC-15 edge-generation rules for DC checking

The following definitions ensure that the decisions made by a *dynamic execution strategy* depend only on past observations.

Definition 13 (History)**.** Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, \mathcal{P} \rangle$ 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 $\sigma(s)$:

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

Definition 14 (Dynamic Execution Strategy)**.** An execution strategy σ for a CSTN is called *dynamic* if for any scenarios s_1 and s_2, and any time-point X:

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

In other words, if a dynamic execution 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 before time t, then σ must also execute X at time t in scenario s_2.

Definition 15 (Dynamic Consistency)**.** A CSTN $\mathcal{S}$ is *dynamically consistent* (DC) if there exists an execution strategy for it that is both dynamic and viable.

The HPC-15 DC-checking algorithm

The HPC-15 DC-checking algorithm uses rules for propagating labeled constraints (i.e., generating new edges). The rules are listed in Table 1. Each rule operates on one or two pre-existing labeled edges (unshaded) and generates a new labeled edge (shaded). With only slight differences (strict vs. non-strict inequalities in R_0 and qLP), the rules apply not only to the standard version of DC, as presented above, but also to the version of DC that allows instantaneous reactions.

The LP rule is similar to edge generation in an STN, except that the labels of the pre-existing edges are conjoined in the generated edge. The R_0 rule removes occurrences of p (and any children of p) from the label on a negative edge emanating from $P?$, the intuition being that any violation of the generated constraint would have to occur before the value of p could be known, implying that the constraint cannot depend on p. The R_3^* rule is similar, except that the label from which p is removed occurs on an adjacent edge.

The next three rules are analogous to the first three, with two important differences. First they involve negative edges terminating at the *zero time-point* Z, whose value is fixed at 0. Such edges represent lower-bound constraints that play a central role in the *earliest-first* execution strategy used to prove the DC-checking algorithm's completeness. Second, these edges may have a new kind of label, called a *q-label.*

Definition 16 (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 each of the form, $p, \neg p$ or $?p$, for some $p \in \mathcal{P}$. $\mathcal{Q}^*$ denotes the set of all q-labels. Note that $\mathcal{P}^* \subseteq \mathcal{Q}^*$.

A constraint whose label includes $?p$ need only be satisfied while the value of $?p$ is unknown, an important distinction for a dynamic execution strategy. For example, recall the CSTN from Fig. 1. The edge/constraint from $P?$ to Y must be satisfied as long as p is either unknown or known to be false. Similarly, the constraint from Y to $P?$ must be satisfied as long as p is either unknown or known to be true. As a result, *both* constraints must be satisfied as long as p is unknown—a condition represented by the q-label, $?p = p \star \neg p$. (The $\star$ operator, defined below, handles the conjunction of q-labels more generally.) The so-called *negative q-loop* from $P?$ to Y to $P?$ is resolvable in this case as long as Y executes after $P?$, because the value of p will make one of the edges in the loop inapplicable, leaving only the other edge to be satisfied.

Definition 17 ($\star$)**.** For any literals $\tilde{p}_1, \tilde{p}_2 \in \{p, \neg p, ?p\}$,

$$\tilde{p}_1 \star \tilde{p}_2 \;=\; \begin{cases} \tilde{p}_1 & \text{if } \tilde{p}_1 = \tilde{p}_2 \\ ?p & \text{if } \tilde{p}_1 \neq \tilde{p}_2 \end{cases}$$

Next, for any q-labels $\ell_1, \ell_2 \in \mathcal{Q}^*$, $\ell_1 \star \ell_2$ denotes the conjunction of literals obtained by applying $\star$ in pairwise fashion to corresponding literals from ℓ_1 and ℓ_2, with the caveat that if a literal $\tilde{p}$ appears in one of ℓ_1 and ℓ_2, but not the other, then $\tilde{p}$ appears in $\ell_1 \star \ell_2$. For example,

$$[p\neg q(?r)t] \;\star\; [pqrv(?w)] \;=\; p(?q)(?r)tv(?w).$$

HPC-15 showed that the following definition of satisfying a labeled constraint is entailed by the semantics of dynamic consistency (cf. Defns. 9-15). They used this definition to prove the soundness of all of the rules in Table 1.

Definition 18. A strategy σ *satisfies* a labeled constraint $(Y - X \leq \delta, \ell)$, where $\ell \in \mathcal{Q}^*$, if for every scenario $s \in P^*$:

(1) s is inconsistent with ℓ; or
(2) $[\sigma(s)]_Y - [\sigma(s)]_X \leq \delta$.

ϵ-Dynamic Consistency of CSTNs

CR-15 introduced a version of dynamic consistency that accommodates a fixed lower bound on reaction times. Their ϵ-DC property is defined in terms of ϵ-dynamic execution strategies, whose decisions can only depend on observations made at least ϵ in the past. This section first presents the CR-15 semantics for ϵ-dynamic consistency. Afterward, it introduces an equivalent semantics that extends the approach taken by HPC-15 for the DC property (cf. Defns. 9–15).

For ϵ-DC semantics, the value of e in $\text{WD}_{2.2}$ should equal the minimum reaction time, ϵ. Then, for example, each time-point labeled by p will be constrained to occur at least ϵ after $P?$. But $\text{WD}_{2.2}$ is not sufficient to ensure that a viable and dynamic strategy is ϵ-dynamic. For example, the CSTN from Fig. 1 satisfies $\text{WD}_{2.2}$ for $\epsilon = 10$, but is not ϵ-DC for $\epsilon = 10$. In particular, any viable and dynamic strategy must satisfy the constraints, $(P? - Y \leq -7, p)$ and $(Y - P? \leq 5, \neg p)$, generated by the HPC-15 rules. But that can only be guaranteed if the decision to execute Y, in cases where p is observed to be *false*, can be made within 5 units after the execution of $P?$, thereby violating ϵ-DC for $\epsilon = 10$.

Comin and Rizzi semantics for ϵ-DC. The CR-15 semantics for ϵ-dynamic consistency begins with preliminary notions from the semantics of dynamic consistency that are equivalent to Defns. 9-12 in this paper. From there, they define *difference sets, ϵ-dynamic execution strategies* and, finally, *ϵ-dynamic consistency,* as follows. For comparison purposes, the CR-15 versions of ϵ-dynamic execution strategies and ϵ-DC are prefixed with ***CR***.

Definition 19 (Difference Set)**.** Let $\mathcal{S}$ be any CSTN; and let s_1 and s_2 be any scenarios. The *difference set* for s_1 and s_2, denoted by $\Delta(s_1; s_2)$, is the set of *observation* time-points in $\mathcal{T}_{s_1}^+$ that yield different outcomes in s_1 and s_2, as follows.

$$\Delta(s_1; s_2) = \{P? \in \mathcal{T}_{s_1}^+ \cap \mathcal{OT} \mid s_1(p) \neq s_2(p)\}$$

Definition 20 (CR-ϵ-dynamic execution strategy)**.** Let $\mathcal{S}$ be any CSTN. Let $\epsilon > 0$ be arbitrary. An execution strategy σ for $\mathcal{S}$ is ***CR-ϵ-dynamic*** if for every pair of scenarios, s_1, s_2, and any time-point $T \in \mathcal{T}_{s_1}^+ \cap \mathcal{T}_{s_2}^+$,

$$[\sigma(s_1)]_T \geq \min\{[\sigma(s_2)]_T, M(s_1; s_2) + \epsilon\}$$

where $M(s_1; s_2) = \min\{[\sigma(s_1)]_{P?} \mid P? \in \Delta(s_1; s_2)\}$.

Definition 21 (CR-ϵ-DC)**.** A CSTN is *ϵ-dynamic consistent* if it has a viable and CR-ϵ-dynamic execution strategy.

Alternative Semantics for ϵ-Dynamic Consistency

This section presents an alternative semantics for ϵ-dynamic consistency that continues the approach taken by HPC-15. It is proven to be equivalent to the CR-ϵ-DC property discussed above. The new semantics will be used to prove that a new ϵ-DC-checking algorithm is sound and complete.

Definition 22 (ϵ-History)**.** Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, \mathcal{P} \rangle$ 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 $\mathit{Hist}^\epsilon(t, s, \sigma)$, is the set of observations made *at or before* $t - \epsilon$ according to $\sigma(s)$:

$$\mathit{Hist}^\epsilon(t, s, \sigma) = \{(p, s(p)) \mid P? \in \mathcal{T}_s^+ \;\&\; [\sigma(s)]_{P?} \leq t - \epsilon\}$$

Definition 23 (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: $\mathit{Hist}^\epsilon(t, s_1, \sigma) = \mathit{Hist}^\epsilon(t, s_2, \sigma)$
then: $[\sigma(s_2)]_X = t$.

Definition 24 (ϵ-DC)**.** Let $\epsilon > 0$. A CSTN is *ϵ-dynamically consistent* if it has a viable and ϵ-dynamic execution strategy.

The following sequence of results culminate in a proof that the definitions of CR-ϵ-DC and ϵ-DC are equivalent.

Lemma 1. *Let $\mathcal{S}$ be any CSTN that satisfies property $\text{WD}_{2.1}$ (Defn. 5); and let σ be any execution strategy that satisfies the constraints in $\text{WD}_{2.2}$ (Defn. 6). Then for any scenario s, any time-point $Y \in \mathcal{T}_s^+$, and any q that appears in $L(Y)$, it must be that $Q? \in \mathcal{T}_s^+$ and $[\sigma(s)]_{Q?} \leq [\sigma(s)]_Y - \epsilon$.*

Proof. Since $\mathcal{S}$ satisfies $\text{WD}_{2.1}$, it follows that $L(Y)$ is an honest label and, hence, that $L(Y) \models L(Q?)$. Therefore, since $Y \in \mathcal{T}_s^+$ (i.e., $s(L(Y)) = \mathit{true}$), it follows that $s(L(Y)) = s(L(Q?)) = \mathit{true}$. Thus, $Q? \in \mathcal{T}_s^+$. Next, σ satisfies the $\text{WD}_{2.2}$ constraint, $(Q? - Y \leq -\epsilon, \ell)$, where $\ell = L(Q?) \wedge L(Y) = L(Y)$. Since $s(L(Y)) = \mathit{true}$, Defn. 18 implies that $[\sigma(s)]_{Q?} - [\sigma(s)]_Y \leq -\epsilon$. □

Definition 25 (Commutative Difference Set)**.** For $\mathcal{S}$, s_1 and s_2 as in Defn. 19, the *commutative difference set*, denoted by $\Delta_c(s_1; s_2)$, is given by: $\Delta_c(s_1; s_2) = \Delta(s_1; s_2) \cap \mathcal{T}_{s_2}^+$.

Note that $\Delta_c(s_1; s_2)$ and $\Delta_c(s_2; s_1)$ are both equal to:

$$\{P? \in \mathcal{T}_{s_1}^+ \cap \mathcal{T}_{s_2}^+ \cap \mathcal{OT} \mid s_1(p) \neq s_2(p)\}$$

In relevant situations, replacing $\Delta(s_1; s_2)$ by $\Delta_c(s_1; s_2)$ in Defn. 20 does not change the value of $M(s_1; s_2)$.

Lemma 2. *Let $\mathcal{S}$ be any CSTN that satisfies property $\text{WD}_{2.1}$ (cf. Defn. 5); let s_1 and s_2 be any scenarios; and let σ be any execution strategy that satisfies the constraints in property $\text{WD}_{2.2}$ (cf. Defn. 6). Then $M(s_1; s_2) = M_c(s_1; s_2)$, where:*

$$M_c(s_1; s_2) = \min\{[\sigma(s_1)]_{P?} \mid P? \in \Delta_c(s_1; s_2)\}.$$

Proof. Let $P? \in \mathcal{T}_{s_1}^+ \cap \mathcal{OT}$ be arbitrary such that $s_1(p) \neq s_2(p)$ and $[\sigma(s_1)]_{P?} = M(s_1; s_2)$. Suppose that $P? \notin \mathcal{T}_{s_2}^+$ (i.e., $s_2(L(P?)) = \mathit{false}$). Then there must be some q that appears in $L(P?)$ such that $s_1(q) \neq s_2(q)$. But then Lemma 1 (with $Y = P?$ and $s = s_1$) ensures that $Q? \in \mathcal{T}_{s_1}^+$ and $[\sigma(s_1)]_{Q?} \leq [\sigma(s_1)]_{P?} - \epsilon < [\sigma(s_1)]_{P?} = M(s_1; s_2)$. However, this, together with $s_1(q) \neq s_2(q)$, contradicts the choice of $P?$. Thus, $P? \in \mathcal{T}_{s_2}^+$ and $M(s_1; s_2) = M_c(s_1; s_2)$. □

Next, an *annotated schedule* is the set of execution events *and observations* that have occurred up to a given time.

Definition 26 (Annotated Schedule)**.** Let σ be an execution strategy for a CSTN $\mathcal{S}$. Let s be any scenario. Then the schedule $\sigma(s)$ determines the execution times for all time-points in $\mathcal{S}$. Let $t_1 < t_2 < \ldots < t_k$ be the $k \leq |\mathcal{T}|$ *distinct* execution times determined by $\sigma(s)$. The i^{th} *annotated (possibly partial) schedule* for $\sigma(s)$ is (ψ_i, θ_i), where:

$\psi_i = \{(X, [\sigma(s)]_X) \mid [\sigma(s)]_X \leq t_i\}$; and
$\theta_i = \{(P?, s(p)) \mid P? \in \mathcal{OT} \text{ and } [\sigma(s)]_{P?} \leq t_i\}$.

In addition, $(\psi_1, \theta_1), (\psi_2, \theta_2), \ldots, (\psi_k, \theta_k)$ is called the *sequence of annotated schedules* for $\sigma(s)$, where:

$\emptyset \subset \psi_1 \subset \psi_2 \subset \ldots \subset \psi_k = \sigma(s)$, and
$\theta_1 \subseteq \theta_2 \subseteq \ldots \subseteq \theta_k = s$.

Lemma 3. *If σ is a valid and CR-ϵ-dynamic strategy for a well-defined CSTN $\mathcal{S}$, then for any scenarios s_1 and s_2, the first time at which the sequences of annotated schedules for $\sigma(s_1)$ and $\sigma(s_2)$ differ is at $M_c(s_1; s_2) = M_c(s_2; s_1)$.*

Proof. Let t^* be the first time at which the annotated schedules for $\sigma(s_1)$ and $\sigma(s_2)$ differ.

Case 1: Some observation time-point $P?$ executes at t^* in both schedules, but yields different outcomes in s_1 and s_2. Without loss of generality, assume that $s_1(p) = \mathit{true}$ and $s_2(p) = \mathit{false}$. Since $P?$ is executed in both schedules, $s_1(L(P?)) = s_2(L(P?)) = \mathit{true}$; thus, $P? \in \mathcal{T}^+_{s_1} \cap \mathcal{T}^+_{s_2}$. Thus, $P? \in \Delta_c(s_1; s_2)$. And since t^* is the first time at which the two annotated schedules differ, $t^* = [\sigma(s_1)]_{P?} = M_c(s_1; s_2)$ and $t^* = [\sigma(s_2)]_{P?} = M_c(s_2; s_1)$.

Case 2: All observation time-points that execute at or before time t^* yield the same outcomes in both scenarios, but some time-point Y executes at t^* in one scenario—say, s_1—but at some later time (or not at all) in the other scenario, s_2. Let (ψ^1, θ^1) be the annotated schedule for $\sigma(s_1)$ at t^*. Since $\sigma(s_2)$ may or may not execute any time-points at t^*, let (ψ^2, θ^2) be the *latest* annotated schedule for $\sigma(s_2)$ that occurs at or before t^*. By construction, $\theta^1 = \theta^2$.

Since Y is executed in s_1, it follows that $s_1 \models L(Y)$. However, suppose that $\theta^1 \not\models L(Y)$. Then, there must be some q (or $\neg q$) in $L(Y)$ such that $Q?$ has not yet executed (i.e., $[\sigma(s_1)]_{Q?} > [\sigma(s_1)]_Y$). But that contradicts Lemma 1. Thus, $\theta^1 \models L(Y)$. But then $\theta^2 = \theta^1$ implies that $s_2 \models \theta^2 \models L(Y)$. Thus, $Y \in \mathcal{T}^+_{s_2}$ (i.e., Y must be executed in s_2). Next, since all observations at or before t^* yield the same results in both scenarios, it follows that $M_c(s_1; s_2) + \epsilon > M_c(s_1; s_2) > t^* = [\sigma(s_1)]_Y$; hence, $[\sigma(s_1)]_Y < M_c(s_1; s_2) + \epsilon$. But then σ being CR-ϵ-dynamic requires that $[\sigma(s_1)]_Y \geq [\sigma(s_2)]_Y$, contradicting the choice of Y. □

Theorem 1. *Let $\mathcal{S}$ be a well defined CSTN; and σ any valid execution strategy. Then for each $\epsilon > 0$, σ is CR-ϵ-dynamic (cf. Defn. 20) if and only if σ is ϵ-dynamic (cf. Defn. 23).*

Proof. ($\Rightarrow$) Suppose that σ is CR-ϵ-dynamic. Let s_1 and s_2 be any scenarios; and let $X \in \mathcal{T}^+_{s_1}$ be any time-point for which $\mathit{Hist}^\epsilon(t, \sigma, s_1) = \mathit{Hist}^\epsilon(t, \sigma, s_2)$, where $t = [\sigma(s_1)]_X$. It must be shown that $X \in \mathcal{T}^+_{s_2}$ and $[\sigma(s_2)]_X = t$.

Now, by Lemma 1, any literal in $L(X)$ must have been observed at or before time $t - \epsilon$. Since $s_1 \models L(X)$, it follows that the label $L(X)$ must have been entailed by the set of observations made in scenario s_1 up to time $t - \epsilon$. Since the two histories are the same up to that time, it follows that $L(X)$ must have been entailed by the observations made in scenario s_2 up to time $t - \epsilon$. Thus, $s_2 \models L(X)$ (i.e., $X \in \mathcal{T}^+_{s_2}$).

Next, let t^* be the first time at which the corresponding annotated schedules for s_1 and s_2 differ. By Lemma 3, $t^* = M_c(s_1; s_2) = M_c(s_2; s_1)$.

Case 1: $t < t^*$. Since t^* is the point of first difference, this implies that $[\sigma(s_2)]_X = t$.

$\mathbf{R}^\epsilon_0$:	$P? \xrightarrow[\langle w, \alpha' \rangle]{\langle w, \alpha\rho \rangle} X$ if $w < \epsilon, \rho \in \{p, \neg p\}$
$\mathbf{R}^\epsilon_3$:	$P? \xrightarrow{\langle w, \alpha\beta \rangle} X \xleftarrow[\langle \max\{v, w-\epsilon\}, \alpha\beta\gamma' \rangle]{\langle v, \beta\gamma\rho \rangle} Y$ if $w \leq \epsilon, \rho \in \{p, \neg p\}$
$\mathbf{qLP}^\epsilon$:	$X \xrightarrow{\langle u, \alpha \rangle} W \xrightarrow{\langle v, \beta \rangle} Z$ if $u < \epsilon, v < 0$; $\langle u + v, (\alpha \star \beta)^\dagger \rangle$
$\mathbf{qR}^\epsilon_3$:	$P? \xrightarrow{\langle w, \gamma \rangle} Z \xleftarrow[\langle \max\{v, w-\epsilon\}, (\gamma \star \beta)^\dagger \rangle]{\langle v, \beta\tilde{p}\theta \rangle} Y$ if $w \leq 0$

Table 2: New edge-generation rules for ϵ-DC checking

Case 2: $t^* \leq t < t^* + \epsilon$. In this case, $[\sigma(s_1)]_X = t < t^* + \epsilon = M_c(s_1; s_2) + \epsilon$. Now, since $X \in \mathcal{T}^+_{s_1} \cap \mathcal{T}^+_{s_2}$, and σ is CR-ϵ-dynamic, it follows that $t = [\sigma(s_1)]_X \geq [\sigma(s_2)]_X$ or $t = [\sigma(s_1)]_X \geq t^* + \epsilon$. Since Case 2 assumes that $t < t^* + \epsilon$, that leaves $t = [\sigma(s_1)]_X \geq [\sigma(s_2)]_X$ as the only possibility. But then, $[\sigma(s_2)]_X \leq t < t^* + \epsilon$ implies (by a symmetric application of CR-ϵ-dynamicity) that $[\sigma(s_2)]_X \geq [\sigma(s_1)]_X$, leaving $[\sigma(s_2)]_X = [\sigma(s_1)]_X$ as the only possibility.

Case 3: $t^* + \epsilon < t$ (i.e., $t^* < t - \epsilon$). Since the point of first difference must involve an observation at time t^* that yields different outcomes in the two scenarios (cf. the proof of Lemma 3), this case contradicts that the histories are the same up to time $t - \epsilon$ in s_1 and s_2.

($\Leftarrow$) Suppose that σ is ϵ-dynamic. Let s_1 and s_2 be any scenarios, and X any time-point in $\mathcal{T}^+_{s_1} \cap \mathcal{T}^+_{s_2}$. Let $t = [\sigma(s_1)]_X$. It must be shown that $[\sigma(s_1)]_X \geq [\sigma(s_2)]_X$ or $[\sigma(s_1)]_X \geq t^* + \epsilon$. Suppose that $[\sigma(s_1)]_X < t^* + \epsilon$. In other words, $t - \epsilon < t^*$. But then $\mathit{Hist}^\epsilon(t, \sigma, s_1) = \mathit{Hist}^\epsilon(t, \sigma, s_2)$, which, by ϵ-dynamicity implies $[\sigma(s_2)]_X = t$. □

A New ϵ-DC-Checking Algorithm for CSTNs

This section introduces an ϵ-DC-checking algorithm for CSTNs based on the propagation of labeled constraints. The new algorithm uses a set of six rules: the LP and qR_0 rules from Table 1, along with the four new rules in Table 2, which are variants of the remaining HPC-15 rules. Like the HPC-15 algorithm, the new algorithm applies the rules to all relevant combinations of edges until it finds either (1) a negative self-loop with a consistent label; or (2) that no rule generates a stronger constraint. In the first case, it answers "Not ϵ-DC"; in the second case, "Yes, ϵ-DC". The algorithm is conjectured to be exponential with a pseudo-polynomial factor that depends on the granularity of the time-domain.

Soundness of the ϵ rules. The following lemma extends a result from HPC-15 to explicate the conditions under which a viable and ϵ-dynamic strategy σ satisfies a *lower-bound* constraint $(X \geq \delta, \ell)$: σ can only execute X before δ if some observation made at least ϵ in the past ensures that the current scenario is inconsistent with ℓ.

Lemma 4. *Let σ be a viable and ϵ-dynamic execution strategy; and $(X \geq \delta, \ell)$ a lower-bound constraint, where $\ell \in \mathcal{P}^*$. Then σ satisfies $(X \geq \delta, \ell)$ if and only if for each scenario s, at least one of the following hold, where $t = [\sigma(s)]_X$:*

(p1) $t \geq \delta$;
(p2) $\bigvee_{p_i \in \ell} \; ([\sigma(s)]_{P_i?} \leq t - \epsilon) \wedge (s(p_i) = \mathit{false})$; *or*
(p3) $\bigvee_{\neg q_j \in \ell} ([\sigma(s)]_{Q_j?} \leq t - \epsilon) \wedge (s(q_j) = \mathit{true})$.

Proof. Suppose σ satisfies $(X \geq \delta, \ell)$, but (p1), (p2) and (p3) are all false. Since (p1) is false, $t < \delta$. But then, by Defn. 18, s must be inconsistent with ℓ. Thus, there is some $p \in \ell$ such that $s(p) = \mathit{false}$ or some $\neg q \in \ell$ such that $s(q) = \mathit{true}$. If more than one, choose the one that is executed first in $\sigma(s)$.

Case 1: $p \in \ell$ with $s(p) = \mathit{false}$. Now (p2) being false implies that $[\sigma(s)]_{P?} > t - \epsilon$. Let s' be the same as s, except that $s' \models \ell$. By construction, the first point of difference between $\sigma(s)$ and $\sigma(s')$ is when $P?$ is executed, and $[\sigma(s')]_{P?} = [\sigma(s)]_{P?} > t - \epsilon$. Thus $[\sigma(s')]_X = [\sigma(s)]_X = t < \delta$ and $s' \models \ell$, contradicting that σ satisfies $(X \geq \delta, \ell)$.

Case 2: Some $\neg q \in \ell$ with $s(q) = \mathit{true}$. Similar. □

Satisfying a q-labeled lower-bound constraint is defined analogously, extending a definition from HPC-15.

Definition 27 (Satisfying a Q-labeled Constraint)**.** Let σ be any viable and ϵ-dynamic execution strategy; and let $(X \geq \delta, \ell)$ be a lower-bound constraint, where $\ell \in \mathcal{Q}^*$. Then σ satisfies $(X \geq \delta, \ell)$ if and only if for each scenario s, at least one of the following hold, where $t = [\sigma(s)]_X$:

(q1) $t \geq \delta$;

(q2) $\bigvee_{p_i \in \ell} \ ([\sigma(s)]_{P_i?} \leq t - \epsilon) \wedge (s(p_i) = \mathit{false})$;

(q3) $\bigvee_{\neg q_j \in \ell} ([\sigma(s)]_{Q_j?} \leq t - \epsilon) \wedge (s(q_j) = \mathit{true})$; or

(q4) $\bigvee_{?r_k \in \ell} ([\sigma(s)]_{R_k?} \leq t - \epsilon)$.

Thus, σ satisfies the constraint if $t \geq \delta$; some $p \in \ell$ is known to be false at $t - \epsilon$; for some $\neg q_j \in \ell$, q_j is known to be true at $t - \epsilon$; or for some $?r_k \in \ell$, r_k has been observed by $t - \epsilon$.

Space limitations preclude proving the soundness of all of the new rules in Table 2. Therefore, the following lemma restricts attention to the rule with the most complicated proof.

Lemma 5. *Rule* qR_3^ϵ *from Table 2 is sound for* ϵ*-DC.*

Proof. Let σ be a viable and ϵ-dynamic strategy that satisfies $C_1 : (P? \geq -w, \gamma)$ and $C_2 : (Y \geq -v, \beta\tilde{p}\theta)$, subject to the conditions listed in Table 1, with $w \leq 0$. However, suppose that σ does not satisfy the generated constraint, $C_3 : (Y \geq -m, \ell)$, where $m = \max\{v, w - \epsilon\}$ and $\ell = (\gamma \star \beta)^\dagger$. Thus, for some scenario s, conditions (q1)-(q4) from Defn. 27 must *all* be false for C_3. Since s is fixed, this proof writes $Y, P?$, etc. instead of $[\sigma(s)]_Y, [\sigma(s)]_{P?}$, etc. For example, since (q1) is false for C_3, $Y < -m$. Now, since σ satisfies C_2, one of (q1)-(q4) must be true for C_2.

Case 1: (q1) is true for C_2 (i.e., $Y \geq -v$). However, this is contradicted by $Y < -m = \min\{-v, -w + \epsilon\} \leq -v$.

Case 2: (q2) is true for C_2 (i.e., for some $g \in \beta\tilde{p}\theta$, $G? \leq Y - \epsilon$ and $s(g) = \mathit{false}$). Now, $g \in \ell$ would contradict (q2) being false for C_3. Similarly, $?g \in \ell$ would contradict (q4) being false for C_3. And if g is a child of some $?h \in \ell$, then, by Lemma 1, $H? < G? - \epsilon \leq Y - 2\epsilon$, which contradicts (q4) being false for C_3. Thus, $g \in \tilde{p}\theta$.

Case 2a: $g = p$, $P? \leq Y - \epsilon$ and $s(p) = \mathit{false}$. Now, $Y < -m \leq -w + \epsilon$ implies that $P? < -w$ (i.e., (q1) is false for C_1). Thus, (q2), (q3) or (q4) must be true for C_1. With no loss of generality, assume that (q2) is true for C_1. Then there must be some $k \in \gamma$ such that $K? \leq P? - \epsilon$ and $s(k) = \mathit{false}$. But then Lemma 1 gives that $K? \leq P? - \epsilon < Y - 2\epsilon$. Now, if $k \in \ell$, this contradicts (q2) being false for C_3; if $?k \in \ell$ it contradicts (q4) being false for C_3; and if $k \notin \ell$, k must be the child of some q-literal $?h$ in ℓ, where $H? < K? \leq Y - \epsilon$, which contradicts (q4) being false for C_3.

Case 2b: $g \in \theta$ (i.e., g is a child of $\tilde{p}$). In this case, $P? < G? \leq Y - \epsilon < -w$, which is similar to Case 2a.

Case 3: (q3) is true for C_2. Handled like Case 2.

Case 4: (q4) is true for C_2 (i.e., for some $?r \in \beta\tilde{p}\theta$, $R? \leq Y - \epsilon$). Now $?r \in \ell$ contradicts (q4) being false for C_3. Thus $?r$ must be a child of some $?h \in \ell$, whence $H? \leq R? - \epsilon \leq Y - 2\epsilon$, which contradicts that (q4) is false for C_3. □

INIT: $\pi_0 = \boxdot$; $t_0^* = -\epsilon$; $\mathcal{T}_0 = \mathcal{T}_0^\epsilon = \emptyset$; $i = 0$.
WHILE (true)
$\mathcal{T}_{\pi_i} = \{X \in \mathcal{T} - \mathcal{T}_i \mid \pi_i \models L(X)\}$
$t_{i+1}^* = \min\{ELB(X, \pi_i) \mid X \in \mathcal{T}_{\pi_i} \cap \mathcal{OT}\}$
$\mathcal{T}_{i+1} = \mathcal{T}_i \cup \{X \in \mathcal{T}_{\pi_i} \mid ELB(X, \pi_i) \leq t_{i+1}^*\}$
$\mathcal{T}_{i+1}^\epsilon = \{X \in \mathcal{T}_{\pi_i} \mid ELB(X, \pi_i) \in (t_{i+1}^*, t_{i+1}^* + \epsilon)\}$
For each $X \in \mathcal{T}_{i+1}^\epsilon \cup \mathcal{T}_{i+1} - \mathcal{T}_i, \sigma(X) := ELB(X, \pi_i)$
$\pi_{i+1} = \pi_i$ plus observation(s) made at time t_{i+1}^*, if any
IF ($t_{i+1}^* = \infty$) HALT
ELSE continue with $i := i + 1$

Table 3: The earliest-first execution strategy, σ

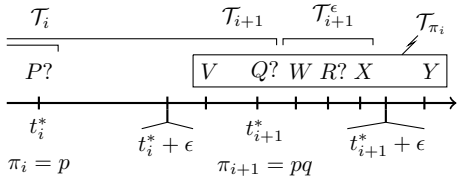

Figure 2: One iteration of the earliest-first strategy

The earliest-first execution strategy. This section presents an iterative execution strategy that executes each time-point as soon as it clears its relevant lower-bound constraints (i.e., at the earliest possible time). While sharing some elements, it is a *significant* extension of the earliest-first strategy presented in HPC-15. Pseudo-code for the earliest-first strategy is given in Table 3; and one iteration of the strategy is illustrated in Figure 2. That example begins with the most recent observation time-point $P?$ having just executed at time t_i^*. The set $\mathcal{T}_i = \{P?, ...\}$ contains all of the time-points that have executed up to that time. The execution of $P?$ generates the observation $p = \mathit{true}$, which is reflected in the *current partial scenario* (CPS) $\pi_i = p$. Crucially, the information in π_i is used only to determine execution times occurring at least ϵ after the information in π_i has become available (i.e., *after* $t_i^* + \epsilon$). In preparation for the next iteration, $\mathcal{T}_{\pi_i} = \{V, Q?, W, R?, X, Y\}$ is the set of as-yet-unexecuted time-points whose labels are entailed by π_i. An *effective lower bound* (ELB) (cf. Defn. 29, below) is computed for each time-point in $\mathcal{T}_{\pi_i}$, represented by the tick-marks in the figure. The time of the next observation event, t_{i+1}^*, is the minimum ELB among *observation* time-points in $\mathcal{T}_{\pi_i}$. In this case, $Q?$ will be the next OTP to execute. $\mathcal{T}_{i+1} = \mathcal{T}_i \cup \{V, Q?\}$ augments $\mathcal{T}_i$ to include the

time-points in $\mathcal{T}_{\pi_i}$ that will execute no later than t^*_{i+1}; and $\mathcal{T}^\epsilon_{i+1} = \{W, R?, X\}$ contains the time-points in $\mathcal{T}_{\pi_i}$ whose ELB values are in the interval, $(t^*_{i+1}, t^*_{i+1} + \epsilon)$. Although these time-points will execute within ϵ of $Q?$, their execution times are not dependent on the observation resulting from $Q?$; instead, based only on the information available at t^*_i, these time-points will be executed at their ELB values in the interval $(t^*_{i+1}, t^*_{i+1} + \epsilon)$. Each $X \in \mathcal{T}^\epsilon_{i+1} \cup \mathcal{T}_{i+1} - \mathcal{T}_i$ is assigned an execution time: $\sigma(X) = ELB(X, \pi_i)$. (Property (I3) in Lemma 6, below, ensures that σ is well defined.) In preparation for the next iteration, the observation $q = true$, made at time t^*_{i+1}, is used to update the CPS: $\pi_{i+1} = pq$. π_{i+1} will be used only to determine execution events that will occur *after* $t^*_{i+1} + \epsilon$.

ELB values are computed as in HPC-15. First, for a given CPS π, a lower-bound constraint labeled by ℓ is *applicable* if ℓ is not already known to be irrelevant according to π.

Definition 28 ($appl(\ell, \pi)$)**.** A label $\ell \in \mathcal{Q}^*$ (whether on a time-point or constraint) is called *applicable* (or *relevant*) with respect to a current partial scenario $\pi \in \mathcal{P}^*$, if for each p that appears in both ℓ and π, p appears *identically* in both (i.e., as p in both, or as $\neg p$ in both, since q-literals cannot appear in π). In such a case, we say that $appl(\ell, \pi)$ holds.

Definition 29 (ELB)**.** The *effective lower bound* of a time-point X with respect to the CPS $\pi \in \mathcal{P}^*$ is given by:

$$ELB(X, \pi) = \max\{\delta \mid \exists \ell \in \mathcal{Q}^* : (X \geq \delta, \ell) \in \mathcal{C} \text{ and } appl(\ell, \pi)\};$$

Lemma 6 was inspired by the "spreading lemma" from HPC-15, but employs new techniques to accommodate ϵ.

Lemma 6 (ϵ-spreading lemma)**.** *Let $\mathcal{S}$ be a well-defined CSTN whose (propagated) constraint set $\mathcal{C}$ is closed under the six ϵ-DC-checking rules.*[1] *Then the following invariants hold during execution by the earliest-first strategy:*

(I1) *For each $X \in \mathcal{T}_{i+1} \cup \mathcal{T}^\epsilon_{i+1}$, the constraint $(X \geq ELB(X, \pi_i), \pi_i)$ is entailed by constraints in $\mathcal{C}$.*

(I2) *Let $X \in \mathcal{T} - \mathcal{T}_{i+1} - \mathcal{T}^\epsilon_{i+1}$ such that $L(X)$ is consistent with π_i. Then $ELB(X, \pi_i) \geq t^*_{i+1} + \epsilon$.*

(I3) *For each $X \in \mathcal{T}^\epsilon_{i+1}$, $ELB(X, \pi_i) = ELB(X, \pi_{i+1})$.*

(I3) ensures that the ELB values for time-points in $\mathcal{T}^\epsilon_{i+1}$ do not change across subsequent iterations, thereby ensuring that the earliest-first strategy is well defined.

Proof. **(I1).** By construction, for each *observation* time-point $P? \in \mathcal{T}_{\pi_i}$, $ELB(P?, \pi_i) \geq t^*_{i+1}$. In addition, if $R?$ is any OTP whose label is consistent with π_i, but not entailed by π_i, $ELB(R?, \pi_i) \geq t^*_{i+1}$. ($R?$ must be the child of some as-yet-unexecuted $R_1?$ whose label, by label honesty, is consistent with π_i. Since there are only finitely many OTPs, eventually some parent OTP $R_j?$, whose label *is* entailed by π_i, must be reached. By Lemma 1, $R_j?$ is constrained to occur before $R?$. Thus, $ELB(R?, \pi_i) > ELB(R_j?, \pi_i) \geq t^*_{i+1}$.)

Next, let $P?$ be some as-yet-unexecuted OTP whose label is consistent with π_i, and let $(P? \geq ELB(P?, \pi_i), \ell')$ be any strongest lower-bound constraint on $P?$. By Rule qR_0 from Table 1, any occurrence of p in ℓ' can be removed. Next, let $X \in \mathcal{T}_{i+1} \cup \mathcal{T}^\epsilon_{i+1}$; and let $(X \geq ELB(X, \pi_i), \ell)$ be any strongest lower-bound constraint on X. Since $ELB(X, \pi_i) < t^*_{i+1} + \epsilon \leq ELB(P?, \pi_i) + \epsilon$, any occurrence of p in ℓ can be removed by an application of qR^ϵ_3, while leaving the lower-bound on X unchanged. (In Rule qR^ϵ_3, let $w = -ELB(P?, \pi_i)$ and $v = -ELB(X, \pi_i)$. It follows that $v = \max\{v, w - \epsilon\}$.) After removing all occurrences of p from all such constraints, if there is another OTP $Q?$ whose label is consistent with π_i, then, in the same way, all occurrences of q can be removed from all such constraints, and so on, until each lower-bound constraint on any $X \in \mathcal{T}_{i+1} \cup \mathcal{T}^\epsilon_{i+1}$ has a label entailed by π_i.

(I2). As seen, each as-yet-unexecuted OTP $R?$ whose label is consistent with π_i satisfies $ELB(R?, \pi_i) \geq t^*_{i+1}$. And, by construction, each $X \in \mathcal{T}_{\pi_i} - \mathcal{T}_{i+1} - \mathcal{T}^\epsilon_{i+1}$ satisfies $ELB(X, \pi_i) \geq t^*_{i+1} + \epsilon$. Finally, let $Y \in \mathcal{T} - \mathcal{T}_{\pi_i}$ such that $L(Y)$ is consistent with, but not entailed by π_i. Then $L(Y)$ must contain some p whose corresponding OTP $P?$ has not yet been executed, whence $ELB(Y, \pi_i) \geq ELB(P?, \pi_i) + \epsilon$ (by $\text{WD}_{2.2}$). Thus, $ELB(Y, \pi_i) \geq t^*_{i+1} + \epsilon$. Now, each such Y has a constraint $(Y \geq ELB(Y, \pi_i), \ell)$ for some ℓ. By a procedure similar to that used in the proof of (I1), Rules qR^ϵ_0 and qR^ϵ_3 can be used to remove all literals from ℓ corresponding to as-yet-unexecuted OTPs whose labels are consistent with π_i, yielding a lower-bound constraint for Y whose bound is at least $t^*_{i+1} + \epsilon$, and whose label is entailed by π_i.

(I3). Let $X \in \mathcal{T}^\epsilon_{i+1}$ be arbitrary. By construction, $ELB(X, \pi_i) > t^*_i$. And, by (I1), $(X \geq ELB(X, \pi_i), \pi_i)$ is entailed by constraints in $\mathcal{C}$. Now, $\pi_{i+1} \models \pi_i$ implies that $(X \geq ELB(X, \pi_i), \pi_{i+1})$ must also be entailed by constraints in $\mathcal{C}$ which, in turn, implies that $ELB(X, \pi_{i+1}) \geq ELB(X, \pi_i)$. However, $ELB(X, \pi_{i+1}) \leq ELB(X, \pi_i)$, since the second bound is computed over the same or more constraints. Thus, the two bounds must be equal. □

Theorem 2. *The new ϵ-DC-checking algorithm for CSTNs is* complete. *That is, if the algorithm says that a given CSTN is ϵ-DC, then the network is ϵ-DC.*

Proof. Let $\mathcal{S}$ be a CSTN that the ϵ-DC-checking algorithm says is ϵ-DC. Let σ be the earliest-first execution strategy. The goal is to demonstrate that σ is viable and ϵ-dynamic. Now, σ is *ϵ-dynamic* since the information available at each time t^*_i is only used to schedule execution events occurring at or after $t^*_i + \epsilon$. The rest of the proof is a refinement of the completeness proof in HPC-15.

Let s be any scenario. It remains to show that $\sigma(s)$ is a solution to the projection of $\mathcal{S}$ onto s (i.e., the STN $\mathcal{S}(s)$). By definition, each edge in $\mathcal{S}(s)$ has a label entailed by s, and was present in $\mathcal{S}$ *before* any propagation.

First, suppose that $\mathcal{S}(s)$ is inconsistent. Then it must have a negative loop. But the corresponding loop in $\mathcal{S}$ would, by repeated use of the LP rule during constraint propagation, yield a *single-edge* negative loop with a consistent label entailed by s. But that would have caused the algorithm to report "Not ϵ-DC", a contradiction. Thus, $\mathcal{S}(s)$ must be consistent.

Next, suppose that $\sigma(s)$ is *not* a solution for $\mathcal{S}(s)$. For each $X \in \mathcal{T}^+_s$, let $x = [\sigma(s)]_X$ be the value assigned to

[1] I.e., applying any of the ϵ-DC-checking rules to constraints in $\mathcal{C}$ only generates a constraint entailed by a constraint already in $\mathcal{C}$.

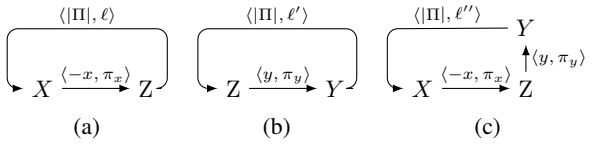

Figure 3: Paths discussed in the proof of Theorem 2

X by $\sigma(s)$. 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 for $\mathcal{S}(s)$, inserting these constraints into $\mathcal{S}(s)$ must create a negative loop (Dechter, Meiri, and Pearl 1991). Without loss of generality, there must be such a loop having exactly one occurrence of Z. Let $\mathcal{L}$ be any such loop.

Case 1 (illustrated in Fig. 3a): $\mathcal{L}$ consists of a lower-bound edge from X to Z, followed by a path Π from Z back to X, where: (1) the label π_x on the lower-bound edge is the CPS when X was scheduled; (2) the lower bound on that edge satisfies $x = ELB(X, \pi_x)$; (3) the edges in Π are original edges from $\mathcal{S}(s)$ having labels consistent with s; (4) ℓ is the conjunction of those labels; and (5) $|\Pi| < x$. The ϵ-spreading lemma ensures that the constraint, $(X \geq -x, \pi_x)$, was entailed by constraints in the fully propagated CSTN. But then, since π_x and ℓ are consistent with s, constraint propagation also would have yielded a negative loop consisting of the edge, $(\mathrm{Z} - \mathrm{Z} \leq |\Pi| - x, \pi_x \wedge \ell)$, whose label is consistent with s. But then the algorithm would have reported that $\mathcal{S}$ was not ϵ-DC, contradicting the main premise.

Case 2 (illustrated in Fig. 3b): $\mathcal{L}$ consists of an upper-bound edge from Z to Y, followed by a path Π from Y back to Z, where: (1) π_y is the CPS when Y was scheduled; (2) the upper bound on the edge from Z to Y is $y = ELB(Y, \pi_y)$; (3) the edges in Π are original edges from $\mathcal{S}(s)$ whose labels are consistent with s; (4) ℓ' is the conjunction of those labels; and (5) $|\Pi| < -y$. In this case, repeated application of the LP rule during constraint propagation in $\mathcal{S}$ would have yielded $(\mathrm{Z} - Y \leq |\Pi|, \ell')$ (i.e., $(Y \geq -|\Pi|, \ell')$), where ℓ' is consistent with s and, hence, also with π_y. But then, $y = ELB(Y, \pi_y) \geq -|\Pi| > y$, a contradiction.

Case 3 (illustrated in Fig. 3c): $\mathcal{L}$ consists of a lower-bound edge from X to Z, followed by an upper-bound edge from Z to Y, followed by a path Π from Y back to X. Here, $|\Pi| - x + y < 0$ and $\pi_x \wedge \pi_y \wedge \ell''$ is consistent with s. As in Case 1, the ϵ-spreading lemma ensures that the lower-bound constraint $(X \geq x, \pi_x)$ must have been entailed by constraints in the fully propagated CSTN, where $x = ELB(X, \pi_x)$. But then repeated application of the LP rule to the path from Y to X to Z would have yielded the edge, $(\mathrm{Z} - Y \leq |\Pi| - x, \pi_x \wedge \ell'')$ (i.e., $(Y \geq x - |\Pi|, \pi_x \wedge \ell'')$). By construction, its label is consistent with π_y, implying that $x - |\Pi|$ is a relevant lower bound for Y. Hence, $y = ELB(Y, \pi_y) \geq x - |\Pi| > y$, a contradiction. □

Computational complexity. The worst-case complexity of the new algorithm is conjectured to be $f(n)(3^{|\mathcal{P}|})^2 H$, where $f(n)$ is a polynomial in n, $|\mathcal{P}|$ is the number of observation time-points, and H is the temporal horizon in a discrete domain. A more precise characterization is under investigation. Meanwhile, the next section presents an empirical evaluation of the algorithm across a variety of realistic CSTNs.

Empirical Evaluation

This section presents an empirical evaluation of our ϵ-DC-checking algorithm to illustrate its practical performance. The algorithm and procedures necessary for its 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. The implementation includes a CSTN editor and a flexible logging system to verify which rules are applied and when. The goal of the implementation is to provide a tool for evaluating the algorithm's behavior, not necessarily its best performance. The code is freely available (Posenato 2015).

We studied the computation time of the ϵ-DC-checking algorithm against n, the number of time-points in the network.

The generation of CSTN instances was based on random workflow schema generated by the ATAPIS toolset (Lanz and Reichert 2014), ensuring a closer approximation to real-world examples, while also generating networks that are more difficult to check than those generated haphazardly. The toolset produces random workflows according to different input parameters that govern the number of activities, the probability of having parallel branches, the probability of inter-task temporal constraints, and so on.

Benchmarks were generated as follows. First, workflow graphs were randomly generated by setting the number of activities to N, the probability for parallel branches to 0.2, the probability for conditional branches to 0.2, and the maximum duration of activities or delays between activities to W. The values of N and W were varied according to the test type, discussed in more detail below. Second, each workflow graph was translated into an equivalent CSTN as proposed by Combi et al. (2014). It is worth noting that different workflow graphs with the same number of activities may translate into CSTNs of different sizes due to different numbers of connector nodes in the workflows. However, it is not hard to verify that a workflow with N activities translates into a CSTN having between $(2N + 2)$ and $(5N + 2)$ nodes.

Generating ϵ-DC workflows for $N > 40$ activities was problematic for the ATAPIS toolset. Therefore, our evaluation was restricted to workflows with $N \leq 40$. The durations or delays between activities was at most $W = 50$. All edge-weights were at most 10^4. For each test, ϵ was fixed at 1.

For each $N \in \{10, 20, 30, 40\}$, we generated 40 workflows of N activities, subdivided into 20 workflows whose CSTNs were ϵ-DC, and 20 whose CSTNs were not ϵ-DC. Since non-ϵ-DC networks were regularly solved one to two orders of magnitude faster than similarly sized ϵ-DC networks, the rest of this section focuses on the results for the ϵ-DC networks. Moreover, preliminary results suggested an exponential dependence on $|\mathcal{P}|$ = the number of observation time-points in the network. Since the goal of our evaluation was to determine the dependence on n = the number of time-points in the network, we fixed the value of $|\mathcal{P}|$ for the different groups of workflows, as follows.

N:	10	20	30	40
$\lvert\mathcal{P}\rvert$:	3	5	7	9

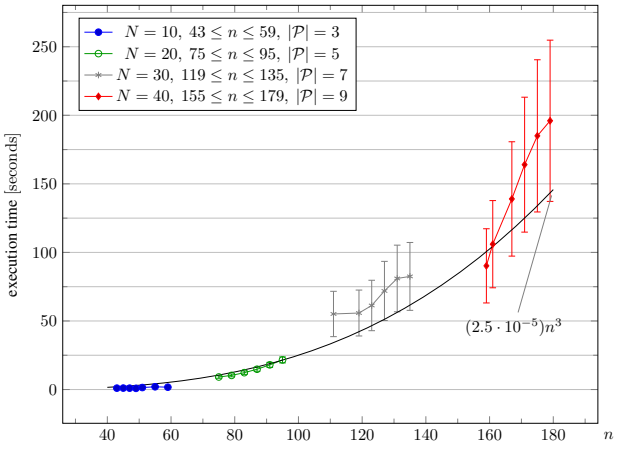

Figure 4: Execution time vs. number of time-points n

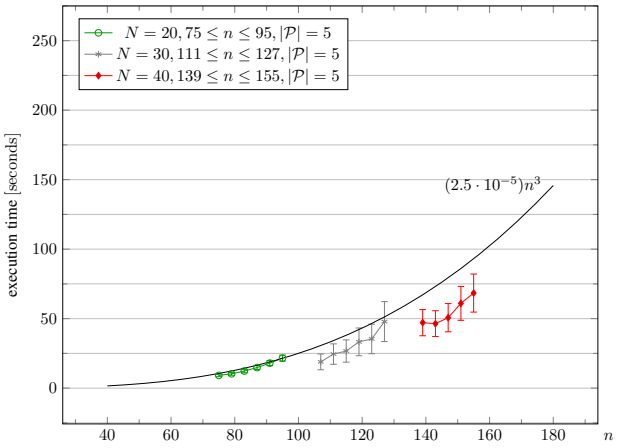

Figure 5: Execution time vs. number of time-points n. Number of proposition is constant.

The first results, shown in Figure 4, demonstrate that the computation time scales well against n, the size of the network. However, they also confirm that the *rate of growth* of the computation time increases sharply as $|\mathcal{P}|$ increases. As previously mentioned, we conjecture that the computation time is exponential with respect to $|\mathcal{P}|$.

The variation in computation time for networks with $N \geq 30$ can be explained by the fact that even if $|\mathcal{P}|$ is fixed, the number of labeled values present on edges in the fully propagated network can vary significantly in response to even small increases in the number of time-points.

The next results, shown in Figure 5, restrict attention to workflows with $|\mathcal{P}| = 5$. The rate of growth of the computation time against n is roughly constant across all three groups of workflows, for $N \in \{20, 30, 40\}$.

We are currently investigating how the computation time of our ϵ-DC-checking algorithm depends on other problem parameters.

Conclusions and Future Work

The most significant contribution of this paper is the new propagation-based ϵ-DC-checking algorithm for CSTNs. The sound-and-complete algorithm is the first ϵ-DC-checking algorithm in the literature to be implemented and evaluated. An initial evaluation suggests that it may be practical for a variety of applications. Future work will focus on (1) more efficiently managing the (possibly exponentially sized) sets of labels on each edge; (2) determining the worst-case complexity of the algorithm; and (3) conducting a more intensive empirical evaluation.

References

Cimatti, A.; Hunsberger, L.; Micheli, A.; Posenato, R.; and Roveri, M. 2014. Sound and complete algorithms for checking the dynamic controllability of temporal networks with uncertainty, disjunction and observation. In Cesta, A.; Combi, C.; and Laroussinie, F., eds., *21st International Symposium on Temporal Representation and Reasoning (TIME-2014), Verona, Italy*, 27–36. IEEE Computer Society.

Combi, C.; Gambini, M.; Migliorini, S.; and Posenato, R. 2014. Representing business processes through a temporal data-centric workflow modeling language: An application to the management of clinical pathways. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on* 44(9):1182–1203.

Comin, C., and Rizzi, R. 2015. Dynamic consistency of conditional simple temporal networks via mean payoff games: a singly-exponential time dc-checking. In Grandi, F.; Lange, M.; and Lomuscio, A., eds., *22st International Symposium on Temporal Representation and Reasoning (TIME 2015)*, 19–28. IEEE CPS.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49(1-3):61–95.

Hunsberger, L.; Posenato, R.; and Combi, C. 2015. A sound-and-complete propagation-based algorithm for checking the dynamic consistency of conditional simple temporal networks. In Grandi, F.; Lange, M.; and Lomuscio, A., eds., *22st International Symposium on Temporal Representation and Reasoning (TIME 2015)*, 4–18. IEEE CPS.

Lanz, A., and Reichert, M. 2014. Enabling time-aware process support with the atapis toolset. In Limonad, L., and Weber, B., eds., *Proceedings of the BPM Demo Sessions 2014*, volume 1295 of *CEUR Workshop Proceedings*, 41–45. CEUR.

Posenato, R. 2015. A CSTN(U) Consistency Check Algorithm Implementation in Java. http://profs.scienze.univr.it/∼posenato/software/cstnu.

Tsamardinos, I.; Vidal, T.; and Pollack, M. E. 2003. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints* 8:365–388.