A Streamlined Model of Conditional Simple Temporal Networks – Semantics and Equivalence Results

Massimo Cairo[1], Luke Hunsberger[2], Roberto Posenato[3], and Romeo Rizzi[4]

1 **Department of Mathematics, University of Trento, Trento, Italy**
massimo.cairo@unitn.it
2 **Department of Computer Science, Vassar College, Poughkeepsie, NY, USA**
hunsberger@vassar.edu
3 **Department of Computer Science, University of Verona, Verona, Italy**
roberto.posenato@univr.it
4 **Department of Computer Science, University of Verona, Verona, Italy**
romeo.rizzi@univr.it

Abstract

A Conditional Simple Temporal Network (CSTN) augments a Simple Temporal Network to include a new kind of time-point, called an observation time-point. The execution of an observation time-point generates information in real time, specifically, the truth value of a propositional letter. In addition, time-points and temporal constraints may be labeled by conjunctions of (positive or negative) propositional letters. A CSTN is called dynamically consistent (DC) if there exists a dynamic strategy for executing its time-points such that no matter how the observations turn out during execution, the time-points whose labels are consistent with those observations have all been executed, and the constraints whose labels are consistent with those observations have all been satisfied. The strategy is dynamic in that its execution decisions may react to observations.

The original formulation of CSTNs included propositional labels only on time-points, but the DC-checking algorithm was impractical because it was based on a conversion of the semantic constraints into an exponentially-sized Disjunctive Temporal Network. Later work added propositional labels to temporal constraints, and yielded a sound-and-complete propagation-based DC-checking algorithm, empirically demonstrated to be practical across a variety of CSTNs.

This paper introduces a streamlined version of a CSTN in which propositional labels may appear on constraints, but not on time-points. This change simplifies the definition of the DC property, as well as the propagation rules for the DC-checking algorithm. It also simplifies the proofs of the soundness and completeness of those rules.

This paper provides two translations from traditional CSTNs to streamlined CSTNs. Each translation preserves the DC property and, for any DC network, ensures that any dynamic execution strategy for that network can be extended to a strategy for its streamlined counterpart.

Finally, this paper presents an empirical comparison of two versions of the DC-checking algorithm: the original version and a simplified version for streamlined CSTNs. The comparison is based on CSTN benchmarks from earlier work. For small-sized CSTNs, the original version shows the best performance, but the performance difference between the two versions decreases as the number of time-points in the CSTN increases. We conclude that the simplified algorithm is a practical alternative for checking the dynamic consistency of CSTNs.

1998 ACM Subject Classification G.2.2 Graph Theory, I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases Conditional Simple Temporal Networks, Dynamic Consistency, Temporal Constraints

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.10

24th International Symposium on Temporal Representation and Reasoning (TIME 2017).
Editors: Sven Schewe, Thomas Schneider, and Jef Wijsen; Article No. 10; pp. 10:1–10:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Dechter et al. [9] defined a *Simple Temporal Network (STN)* as a pair $(\mathcal{T},\mathcal{C})$, where $\mathcal{T}$ is set of real-valued variables, called time-points; and $\mathcal{C}$ is a set of binary difference constraints (a.k.a. temporal constraints) on those time-points. The *Simple Temporal Problem (STP)* is that of determining whether any given STN is *consistent* (i.e., whether there exists a complete assignment to the time-points in $\mathcal{T}$ that satisfies all of the constraints in $\mathcal{C}$). Typically, an STN includes a special time-point, Z, whose value is fixed at zero. Binary constraints involving Z correspond to unary constraints since $X \leq \delta$ is equivalent to $X - Z \leq \delta$; and $X \geq \delta$ is equivalent to $Z - X \leq -\delta$. If an STN does not have a Z time-point, one can be inserted without affecting the consistency of the network [11].

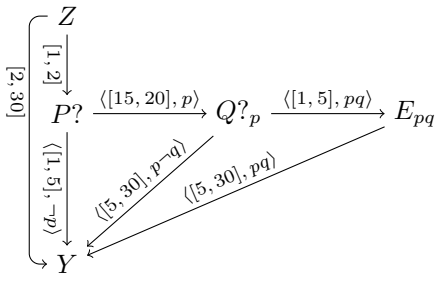
Tsamardinos et al. [18] introduced *Conditional Simple Temporal Networks (CSTNs),* augmenting STNs to include propositional letters, observation time-points, and propositional labels on time-points. Each observation time-point $P?$ has a corresponding propositional letter p, where the execution of $P?$ non-deterministically generates a truth value for p. In addition, any time-point – whether observational or not – may be labeled by a conjunction of (positive or negative) propositional literals, the idea being that only the time-points whose labels are consistent with the incrementally revealed observations need to be executed; and only the constraints among *those* time-points need to be satisfied. A CSTN is called *dynamically consistent (DC)* if there exists a dynamic *strategy* for executing its time-points such that no matter how the observations turn out during execution, the time-points whose labels are consistent with those observations have all been executed, and the constraints among those time-points have all been satisfied. The strategy is dynamic in that its execution decisions may react to observations in real time. They presented an algorithm for checking the DC property – called a DC-checking algorithm – but it was not practical due to its conversion of the semantic constraints into an exponentially-sized Disjunctive Temporal Network.

Hunsberger et al. [15] generalized CSTNs, allowing propositional labels on both time-points *and constraints.* They then introduced rules for propagating labeled constraints, which they used as the basis for a sound-and-complete DC-checking algorithm that was empirically demonstrated to be practical across a variety of CSTNs. To facilitate proving that their propagation rules were sound and complete, they also defined several properties associated with propositional labels (e.g., label *honesty* and label *coherence*); and they formalized a set of *well-definedness* properties that were implicit in the original formulation of CSTNs.

The motivation for this paper began with the observation that proving the soundness and completeness properties for the propagation-based DC-checking algorithm was unnecessarily complicated by the presence of propositional labels on time-points. As this paper shows, no loss of generality results from streamlining CSTNs by allowing propositional labels on constraints, but not on time-points. The streamlined definition of a CSTN simplifies: (1) the definition of the DC property, (2) the definition of the propagation rules, and (3) the soundness and completeness proofs for those rules. The paper proves the equivalence of the streamlined CSTN and the prior formulation. It also empirically demonstrates that the performance of the correspondingly simpler DC-checking algorithm is similar to that of the original DC-checking algorithm, and that the performance difference between the two algorithms decreases as the number of time-points increases.

2 Background

This section reviews the definitions needed for the more general version of CSTN and dynamic consistency presented by Hunsberger et al. [15].



Time Point	Meaning
Z	Begin
$P?$	Do Blood Test (gen. value for p)
$Q?_p$	Repeat Blood Test (only if $p = \top$; generate value for q)
E_{pq}	Do Treatment (only if $p = q = \top$)
Y	End

Figure 1 The graph of a sample CSTN, discussed in the text.

▶ **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 as $\boxdot$.
- for any label ℓ, and any $p \in P$, 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 P, denoted by P^*, is the set of all consistent labels whose literals are drawn from P.

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

- P is a finite set of propositional letters (or propositions);
- $\mathcal{T}$ is a finite set of real-valued variables, called time-points;
- $\mathcal{C}$ 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 $\ell \in P^*$;
- $L : \mathcal{T} \to P^*$ is a function assigning labels to time-points;
- $\mathcal{OT} \subseteq \mathcal{T}$ is a (finite) set of observation time-points; and
- $\mathcal{O} : P \to \mathcal{OT}$ is a bijection between propositional letters and observation time-points.

For convenience, the observation time-point associated with p may be denoted by $P?$ instead of the more cumbersome $\mathcal{O}(p)$. In a CSTN graph, the time-points serve as the nodes, and each labeled constraint, $(Y - X \leq \delta, \ell)$, is represented by an arrow from X to Y annotated by the labeled value $\langle \delta, \ell \rangle$, as follows: $X \xrightarrow{\langle \delta, \ell \rangle} Y$. (If $\ell = \boxdot$, then the label and angle brackets may be omitted, as follows: $X \xrightarrow{\delta} Y$.) For convenience, an interval constraint such as $(Y - X \in [a, b], \ell)$ may be represented by a single edge from X to Y labeled by $\langle [a, b], \ell \rangle$, although it corresponds to two constraints in the CSTN definition. Finally, since any time-points, X and Y, may participate in multiple constraints of the form, $(Y - X \leq \delta_i, \ell_i)$, each edge in the graph may have multiple labeled values of the form, $\langle \delta_i, \ell_i \rangle$.

Fig. 1 shows the graph of a CSTN for a simple health-care example, originally presented by Hunsberger et al. [15]. It will be used as a running example. The nodes, Z and Y, represent starting and ending time-points, respectively. $P?$ represents the time at which a particular blood test is performed. If this test generates a positive result, represented by $p = \top$, then the test must be repeated at time-point $Q?$, which generates a truth value for q. Since $Q?$ applies only in scenarios where $p = \top$, it is labeled by p. If both tests generate positive results, then an emergency treatment is applied at time-point E, whose label is pq.

The edges in this graph use the compact interval notation. For example, the edge from $P?$ to $Q?$ labeled by $\langle [15, 20], p \rangle$ represents that the difference, $Q? - P?$, must lie within $[15, 20]$

in scenarios where p is true (i.e., the repeated test must be performed between 15 and 20 minutes after the first test). Similarly, the edges from $Q?$ to E, and from E to Y are labeled by pq, indicating that those constraints apply only in scenarios where both p and q are $\top$.

2.1 Well-definedness properties for CSTNs

The following definitions specify properties that any *well defined* CSTN must hold. For example, without the *label coherence* property (Defn. 4), it might happen X is labeled by p, Y is labeled by q, and $\mathcal{C}$ contains the constraint, $(Y - X \leq -2, p)$: $X_p \xrightarrow{\langle -2, p \rangle} Y_q$. Then, in the scenario $p\neg q$, X must be executed and Y must not be executed, but the constraint $(Y - X \leq -2, p)$ must hold, which is absurd. Similar examples can be generated for the other well-definedness properties. In short, CSTNs that are not well defined are of no use.

▶ **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, $\ell \models L(Q?)$ (i.e., ℓ contains all of the literals from the label of the observation time-point for q).

▶ **Definition 4** (WD1: Label coherence)**.** A CSTN holds property WD1 (i.e., has *coherent labels)* if for each labeled constraint, $(Y - X \leq \delta, \ell)$, the label ℓ is satisfiable and entails $L(X) \wedge L(Y)$ (i.e., contains all of the literals from $L(X)$ and $L(Y)$).

▶ **Definition 5** (WD2)**.** A CSTN holds property WD2 if:
(a) for each time-point $T \in \mathcal{T}$, its label $L(T)$ is honest, and
(b) for each $p \in P$ that appears in $L(T)$, $\mathcal{C}$ contains a constraint, $(P? - T \leq -\epsilon, L(T))$, for some $\epsilon > 0$ (i.e., T is constrained to occur after $P?$).

▶ **Definition 6** (WD3: Constraint Label Honesty)**.** A CSTN holds property WD3 if the label on each of its constraints is honest.

▶ **Definition 7** (Well defined CSTN)**.** A CSTN is called *well defined* if it holds properties WD1, WD2 and WD3.

2.2 Dynamic Consistency for CSTNs

▶ **Definition 8** (Scenario)**.** A *scenario* over a set P of propositional letters is a function, $s : P \to \{\top, \bot\}$, that assigns a truth value to each letter in P. Any such function also provides the truth value for any label $\ell \in P^*$, which is denoted by $s(\ell)$. The set of all scenarios over P is denoted by $\mathcal{I}$.

▶ **Definition 9** (Schedule)**.** A *schedule* for a set of time-points $\mathcal{T}$ is a mapping, $\psi : \mathcal{T} \to \mathbb{R}$, 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.

▶ **Definition 10** (Projection)**.** Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P \rangle$ be any CSTN, and s any scenario over P. The *projection* of S onto s – notated $Prj(\mathcal{S}, s)$ – is the STN, $(\mathcal{T}_s^+, C_s^+)$, where:
- $\mathcal{T}_s^+ = \{T \in \mathcal{T} \mid s \models L(T)\}$; and
- $C_s^+ = \{(Y - X \leq \delta) \mid \text{for some } \ell, (Y - X \leq \delta, \ell) \in C \text{ and } s \models \ell\}$.

For convenience, we also define $\mathcal{OT}_s^+ = \mathcal{OT} \cap \mathcal{T}_s^+$ (i.e., the set of observation time-points whose labels are entailed by s).

▶ **Definition 11** (Execution Strategy)**.** Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P \rangle$ be any CSTN. An *execution strategy* for $\mathcal{S}$ is a mapping, $\sigma : \mathcal{I} \to \Psi$, such that for each scenario $s \in \mathcal{I}$, the

domain of the schedule $\sigma(s)$ is $\mathcal{T}_s^+$. If, in addition, for each scenario s, the schedule $\sigma(s)$ is a solution to the projection $Prj(\mathcal{S}, s)$, then σ is called *viable.* In any case, the execution time for the time-point X in the schedule $\sigma(s)$ is denoted by $[\sigma(s)]_X$. In addition, $|\sigma|$ denotes the maximum value assigned by σ to any time-point in $\mathcal{T}$ in any scenario $s \in \mathcal{I}$.

▶ **Definition 12** (History). Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P \rangle$ 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 $\sigma(s)$: $Hist(t, s, \sigma) = \{(p, s(p)) \mid P? \in \mathcal{OT}_s^+ \text{ and } [\sigma(s)]_{P?} < t\}$.

▶ **Definition 13** (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 $X \in \mathcal{T}_{s_1}^+$:

- If $Hist(t, s_1, \sigma) = Hist(t, s_2, \sigma)$, where $t = [\sigma(s_1)]_X$, then $X \in \mathcal{T}_{s_2}^+$ and $[\sigma(s_2)]_X = t$.

▶ **Definition 14** (Dynamic Consistency). A CSTN is *dynamically consistent* (DC) if there exists an execution strategy for it that is both dynamic and viable.

2.3 Motivations for a new definition

The prior definitions of CSTN are convenient for the designer working in some domain. The designer typically knows the scenarios in which each time-point must be executed and can directly represent that information in the time-point labels. Furthermore, if the designer constructs a CSTN that is not well defined, it can be easily remedied by augmenting the labels on time-points and constraints to make them honest and coherent, and by inserting any missing constraints needed for property WD2.b. However, the presence of labels on time-points needlessly complicates the constraint-propagation rules needed for practical DC checking. For example, Hunsberger et al. [15] introduced a *child-of* relation among propositional letters that derives from cases where observation time-points have non-empty labels. The applicability conditions of their propagation rules are littered with special cases to handle the children of propositional letters. As a consequence, proving that the propagation rules are sound requires dealing with these special cases. However, if a CSTN has no labels on its time-points, then these complexities disappear. Indeed, it is trivial to check that all of the well-definedness properties become vacuous if there are no labels on any time-points.

These considerations motivated the search for an equivalent formulation of CSTNs that does not include labels on time-points. This paper presents such a formulation, called *streamlined CSTN,* and proves that it is equivalent to the ordinary CSTN presented above. The paper presents two alternative translations from ordinary to streamlined CSTNs, each of which preserves the most important properties of a CSTN, including dynamic consistency.

A designer working in some domain may continue to reap the benefits of working with CSTNs having labels on time-points, leaving it to the DC-checking algorithm to convert the CSTN into a streamlined version before carrying out any constraint propagation. Thus, the streamlined CSTN simplifies the theoretical foundations of CSTNs while still allowing users to work with the earlier version should they find it useful to do so.

3 Streamlined Model of CSTNs

This section presents the definition for a streamlined CSTN, which simply removes the assignment of labels to time-points. It also specifies the slight modifications to the sequence of definitions needed for defining the dynamic consistency of CSTNs – namely, that for any scenario s, $\mathcal{T}_s^+ = \mathcal{T}$ and $\mathcal{OT}_s^+ = \mathcal{OT}$, since there are no labels on any time-points.

▶ **Definition 15** (CSTN$_\boxdot$). A *Streamlined Conditional Simple Temporal Network* (CSTN$_\boxdot$) is a tuple, $\langle \mathcal{T}, \mathcal{C}, \mathcal{OT}, \mathcal{O}, P \rangle$, where:

- P is a finite set of propositional letters (or propositions);
- $\mathcal{T}$ is a finite set of real-valued variables, called time-points;
- $\mathcal{C}$ is a set of labeled constraints of the form, $(Y - X \leq \delta, \ell)$, where $X, Y \in \mathcal{T}$, $\delta \in \mathbb{R}$, $\ell \in P^*$;
- $\mathcal{OT} \subseteq \mathcal{T}$ is a (finite) set of observation time-points; and
- $\mathcal{O} : P \to \mathcal{OT}$ is a bijection between propositional letters and observation time-points.

As previously noted, there is no need for any of the well-definedness properties (Defns. 3-6) for streamlined CSTNs since they become vacuous if there are no labels on time-points. The existing definitions of *scenarios* and *schedules* (Defns. 8 and 9) apply to CSTN$_\boxdot$ without any changes, but Defns. 10-13 must be slightly modified for CSTN$_\boxdot$, as follows.

- **Projection (Defn. 10).** The same, except that for each scenario s, $\mathcal{T}_s^+ = \mathcal{T}$, since there are no labels on any time-points (equivalently, since $s \models \boxdot$). Similarly, $\mathcal{OT}^+ = \mathcal{OT}$.
- **Execution Strategy (Defn. 11).** The same, except that for each scenario s, the domain of $\sigma(s)$ is $\mathcal{T}_s^+ = \mathcal{T}$.
- **History (Defn. 12).** The same, but replace $P? \in \mathcal{OT}_s^+$ by $P? \in \mathcal{OT}$, since $\mathcal{OT}_s^+ = \mathcal{OT}$.
- **Dynamic Execution Strategy (Defn. 13).** The same, but replace $X \in \mathcal{T}_{s_1}^+$ by $X \in \mathcal{T}$, since $\mathcal{T}_{s_1}^+ = \mathcal{T}$, and delete the (now redundant) requirement that $X \in \mathcal{T}_{s_2}^+$.

4 Equivalence of the CSTN and CSTN$_\boxdot$ Models

This section presents two translations from CSTNs to streamlined CSTNs and proves that each translation preserves the property of dynamic consistency. Furthermore, using either translation, any dynamic execution strategy for a DC CSTN can be extended to a dynamic execution strategy for its streamlined counterpart such that for any scenario, the time-points executed by the strategy for the CSTN are executed at the same times by the corresponding strategy for the streamlined CSTN. The first translation inserts constraints that, for each scenario s, require all time-points whose labels are entailed by s to be executed *at or before* a fixed horizon h, while all time-points whose labels are inconsistent with s are constrained to occur *after* h. The second translation does not add any such constraints and, thus, is much simpler; however, it is less explicit about the time-points that are executed in the original CSTN in any given scenario. Both translations are presented here since they illuminate different aspects of the relationships between the CSTN and CSTN$_\boxdot$ models.

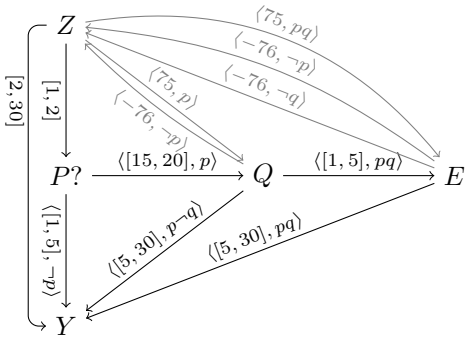
4.1 The First Translation from CSTN to CSTN$_\boxdot$

We begin with some preliminary results.

▶ **Lemma 16** (Upper bound for DC CSTNs). *Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P \rangle$ be any CSTN.*

- *Let $k = |\mathcal{OT}|$ be the number of observation time-points in $\mathcal{S}$;*
- *let $n = |\mathcal{T}|$ the number of time-points;*
- *let $M = \max\{|\delta| \text{ such that some } (Y - X \leq \delta, \ell) \in \mathcal{C}\}$ be the maximum absolute value of any bound on any constraint in $\mathcal{C}$; and*
- *let $h = Mn(2^k)$ be the* horizon.

If $\mathcal{S}$ is DC, then there exists a viable and dynamic execution strategy σ for $\mathcal{S}$ such that for every scenario s, and every time-point X, $[\sigma(s)]_X \leq h$ (i.e., $|\sigma| \leq h$).



The horizon value (cf. Lemma 17) is:
$h = Mn = 15 \cdot 5 = 75$.
$\mathcal{C}'_1 = \{(Q? \leq 75, p), (E \leq 75, pq), (P? \leq 75, \boxdot), (Y \leq 75, \boxdot)\}$.
$\mathcal{C}'_2 = \{(Q? \geq 76, \neg p), (E \geq 76, \neg p), (E \geq 76, \neg q)\}$.
The third and fourth constraints in $\mathcal{C}'_1$ are not shown in the graph since they would be redundant.

Figure 2 The CSTN$_\boxdot$ derived from the sample CSTN using the first method of translation.

Proof. Comin and Rizzi [7] (their Theorem 6) proved a correspondence between CSTNs and Hyper Temporal Networks (HyTNs) such that: (1) the corresponding HyTN has at most $(2^k)n$ time-points; and (2) the CSTN is DC if and only if the corresponding HyTN is consistent. Furthermore, Comin [6] (his Theorem 2.3) showed that an HyTN is consistent if and only if it has no negative cycles (his Defn. 2.5). Now, a negative cycle in an HyTN consists of hyperarcs. A finite cyclic path is obtained from a negative cycle by selecting at most one ordinary arc from each hyperarc such that the selected arcs form a cycle in which each node appears at most once. A negative cycle in an HyTN is characterized by each finite cyclic path having negative length. Therefore, if $\mathcal{S}$ is a DC CSTN, there can be no negative cycle in its corresponding HyTN. Furthermore, inserting constraints of the form $X \leq h$ into the CSTN for each X could not introduce a negative cycle into the HyTN because: (i) all such edges emanate from Z, thus only one can appear in any finite cyclic path; (ii) there can be no more than $(2^k)n$ pre-existing edges in each finite cyclic path; and (iii) the absolute value of each weight on the pre-existing edges can be no more than M. As a result, each finite cyclic path that includes an edge of length $h = Mn(2^k)$ cannot have negative length. ◀

▶ **Lemma 17** (Tighter upper bound; rational weights). *Let $\mathcal{S}$ be a DC CSTN with n time-points and* rational *weights. If M is the maximum absolute value of any* negative *edge in $\mathcal{S}$, then the network obtained by constraining every time-point in $\mathcal{S}$ to occur before time Mn is DC.*

The proof of Lemma 17 is in the Appendix.

▶ **Definition 18** (Reducing a CSTN to a CSTN$_\boxdot$). Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P\rangle$ be a well defined CSTN. The reduction of $\mathcal{S}$ is the CSTN$_\boxdot$, $\mathcal{S}_\boxdot = \langle \mathcal{T}, \mathcal{C} \cup \mathcal{C}'_1 \cup \mathcal{C}'_2, \mathcal{OT}, \mathcal{O}, P\rangle$, where:

- some $h \geq Mn(2^k)$ serves as the *horizon* (cf. Lemma 16);*
- $\mathcal{C}'_1 = \bigcup_{X\in\mathcal{T}} \{(X - Z \leq h, L(X))\} = \bigcup_{X\in\mathcal{T}} \{(X \leq h, L(X))\}$;† and
- $\mathcal{C}'_2 = \bigcup_{X\in\mathcal{T},\rho\in L(X)} \{(Z - X \leq -h - 1, \neg\rho)\} = \bigcup_{X\in\mathcal{T},\rho\in L(X)} \{(X \geq h + 1 > h, \neg\rho)\}$.

The constraints in $\mathcal{C}'_1$ force each X to be executed at or before h in scenarios that entail X's label. And the constraints in $\mathcal{C}'_2$ force each X to be executed after h in scenarios that do not entail X's label.

Fig. 2 shows the streamlined version of the running example from Fig. 1.

The first method of translation generates an equivalent streamlined CSTN, as follows.

* If the weights in $\mathcal{S}$ are rational, then any $h \geq Mn$ can serve as the horizon.
† In $\mathcal{C}'_2$, ρ represents a positive or negative literal, and $\neg\rho$ the literal with the opposite polarity of ρ.

▶ **Theorem 19** (Equivalence of CSTN and CSTN$_\boxdot$ Models). *Let $\mathcal{S}$ be any well defined CSTN. Then $\mathcal{S}_\boxdot$ (i.e., the reduction of $\mathcal{S}$ to a CSTN$_\boxdot$) is equivalent to $\mathcal{S}$ in the sense that $\mathcal{S}$ is DC if and only if $\mathcal{S}_\boxdot$ is DC. Furthermore, in the case where $\mathcal{S}$ and $\mathcal{S}_\boxdot$ are both DC, if σ is any viable and dynamic (V & D) strategy for $\mathcal{S}$ for which $|\sigma| \leq h$, then there is an equivalent V & D strategy $\sigma_\boxdot$ for $\mathcal{S}_\boxdot$ in the sense that for each scenario s and any $X \in \mathcal{T}$:*

- *If $X \in \mathcal{T}_s^+$, then $[\sigma_\boxdot(s)]_X = [\sigma(s)]_X \leq h$; otherwise, $[\sigma_\boxdot(s)]_X \geq h+1 > h$.*

Proof. Let $\mathcal{S}$ be any well defined CSTN; and let $\mathcal{S}_\boxdot$ be the corresponding CSTN$_\boxdot$.

Part 1: $\mathcal{S}$ is DC $\Rightarrow$ $\mathcal{S}_\boxdot$ is DC

By Lemma 16 there exists a V & D execution strategy σ for $\mathcal{S}$ such that $|\sigma| \leq h$. For any such σ, define $\sigma_\boxdot : \mathcal{I} \to \Psi$ as follows. For any scenario $s \in \mathcal{I}$, let $\sigma(s) : \mathcal{T} \to \mathbb{R}$ be the schedule that is the same as $\sigma(s)$ on time-points in $\mathcal{T}_s^+$, but that maps time-points not in $\mathcal{T}_s^+$ to $h+1$ (i.e., just over the horizon). In other words:

- For each $X \in \mathcal{T}_s^+$, let $[\sigma_\boxdot(s)]_X = [\sigma(s)]_X \leq h$. (1)
- For each $X \in \mathcal{T} \backslash \mathcal{T}_s^+$, let $[\sigma_\boxdot(s)]_X = h+1$. (2)

Note that $|\sigma_\boxdot| \leq h+1$. Next, we show that:

- for any $t \leq h+1$, and any scenario s, $\mathit{Hist}(t, s, \sigma) = \mathit{Hist}_\boxdot(t, s, \sigma_\boxdot)$. ($\star$)

To see this, let p be any propositional letter. If $(p, s(p))$ is in $\mathit{Hist}_\boxdot(t, s, \sigma_\boxdot)$, it follows that $[\sigma_\boxdot(s)]_{P?} < t \leq h+1$ and, hence, by (1) and (2), that $[\sigma_\boxdot(s)]_{P?} \leq h$, in which case, $P? \in \mathcal{T}_s^+$ and $[\sigma(s)]_{P?} = [\sigma_\boxdot(s)]_{P?} < t$. Therefore, $(p, s(p))$ is also in $\mathit{Hist}(t, s, \sigma)$. On the other hand, if $(p, s(p))$ is in $\mathit{Hist}(t, s, \sigma)$, it follows that $P? \in \mathcal{T}_s^+$ and thus $[\sigma_\boxdot(s)]_{P?} = [\sigma(s)]_{P?} < t$, in which case $(p, s(p))$ also appears in $\mathit{Hist}_\boxdot(t, s, \sigma_\boxdot)$.

Next we aim to show that $\sigma_\boxdot$ is a *dynamic* strategy. Toward that end, let s_1 and s_2 be any scenarios, let X be any time-point in $\mathcal{T}$, let $t = [\sigma_\boxdot(s_1)]_X$, and suppose that $\mathit{Hist}_\boxdot(t, s_1, \sigma_\boxdot) = \mathit{Hist}_\boxdot(t, s_2, \sigma_\boxdot)$. We must show that $[\sigma_\boxdot(s_2)]_X = t$. First, note that since $t \leq h+1$ and $\mathit{Hist}_\boxdot(t, s_1, \sigma_\boxdot) = \mathit{Hist}_\boxdot(t, s_2, \sigma_\boxdot)$, it follows from ($\star$) that:

- $\mathit{Hist}(t, s_1, \sigma) = \mathit{Hist}_\boxdot(t, s_1, \sigma_\boxdot) = \mathit{Hist}_\boxdot(t, s_2, \sigma_\boxdot) = \mathit{Hist}(t, s_2, \sigma)$. (†)

There are two cases to consider.

1. $X \in \mathcal{T}_{s_1}^+$.
   By (1), $t = [\sigma_\boxdot(s_1)]_X = [\sigma(s_1)]_X \leq h$. Therefore, (†) together with the dynamicity of σ implies that $X \in \mathcal{T}_{s_2}^+$ and $[\sigma(s_2)]_X = t \leq h$, whence $[\sigma_\boxdot(s_2)]_X = [\sigma(s_2)]_X = t$.
2. $X \notin \mathcal{T}_{s_1}^+$.
   In this case, it follows from (2) that $t = [\sigma_\boxdot(s_1)]_X = h+1$. Next, let p be any propositional letter for which $(p, s(p))$ is in the histories, $\mathit{Hist}_\boxdot(t, s_1, \sigma_\boxdot) = \mathit{Hist}_\boxdot(t, s_2, \sigma_\boxdot)$. Then $[\sigma_\boxdot(s_1)]_{P?} < t = h+1$ which, by (1) and (2) above, implies that $[\sigma_\boxdot(s_1)]_{P?} \leq h$ and $P? \in \mathcal{T}_{s_1}^+$. Similarly, $[\sigma_\boxdot(s_2)]_{P?} \leq h$ and $P? \in \mathcal{T}_{s_2}$. Furthermore, by (2), any observation time-point $Q?$ that does not appear in those histories must be executed at time $h+1$, which implies that $Q? \notin \mathcal{T}_{s_1}^+$ and $Q? \notin \mathcal{T}_{s_2}^+$. Thus, those histories contain *exactly* the observation time-points in $\mathcal{T}_{s_1}^+ \cap \mathcal{T}_{s_2}^+$. Since X is not in $\mathcal{T}_{s_1}^+$, it follows that $s_1 \not\models L(X)$. Now, if $X \in \mathcal{T}_{s_2}^+$ (i.e., $s_2 \models L(X)$), then (without loss of generality) there must be some $p \in L(X)$ such that $s_1 \not\models p$, but $s_2 \models p$. Since $L(X)$ is honest (by WD2), $L(X) \models L(P?)$. Therefore, $s_2 \models L(X) \models L(P?)$ and, hence, $P? \in \mathcal{T}_{s_2}^+$, which implies that $[\sigma_\boxdot(s_2)]_{P?} = [\sigma(s_2)]_{P?} \leq h$. But then p must appear in $\mathit{Hist}_\boxdot(h+1, s_2, \sigma_\boxdot)$. However, since p yields different outcomes in s_1 and s_2, that contradicts that the histories for s_1 and s_2 are the same. Therefore, it must be that $X \notin \mathcal{T}_{s_2}^+$, in which case, $[\sigma_\boxdot(s_2)] = h+1 = t$.

By cases 1 and 2, it follows that $\sigma_{\boxdot}$ is dynamic. It remains to show that $\sigma_{\boxdot}$ is viable. Toward that end, let s be any scenario, and let $(Y - X \leq \delta, \ell)$ be any constraint in $\mathcal{S}_{\boxdot}$ whose label ℓ is entailed by s. Now, if this constraint is in $\mathcal{S}$, then by WD1, $\ell \models L(X) \wedge L(Y)$. Hence, $s \models \ell \models L(X) \wedge L(Y)$, which implies that $X, Y \in \mathcal{T}_s^+$. Thus, σ and $\sigma_{\boxdot}$ execute X and Y at the same times in s. And, since $s \models \ell$, this constraint is in $\mathcal{C}_s^+$; thus, the viability of σ ensures that it is satisfied by $\sigma_{\boxdot}(s)$. On the other hand, if this constraint is not in $\mathcal{S}$, then it must be one of the constraints, $(X - Z \leq h, L(X))$ from $\mathcal{C}_1'$, or $(Z - X \leq -h - 1, \neg\rho)$ from $\mathcal{C}_2'$, for some $\rho \in L(X)$. Now if $s \models L(X)$, then (1) requires that $[\sigma_{\boxdot}(s)]_X \leq h$, which implies that the first constraint is satisfied, and $s \models L(X) \models \rho$ implies that the second constraint is trivially satisfied. On the other hand, if $s \not\models L(X)$, then the first constraint is trivially satisfied and, by (2), $[\sigma_{\boxdot}(s)]_X = h + 1$, which implies that the second constraint is satisfied. Therefore, since the choice of constraint was arbitrary, it follows that $\sigma_{\boxdot}(s)$ must be a solution to the projection $Prj_{\boxdot}(\mathcal{S}_{\boxdot}, s)$. And since s was arbitrary, $\sigma_{\boxdot}$ must be viable.

Part 2: $\mathcal{S}_{\boxdot}$ is DC $\Rightarrow$ $\mathcal{S}$ is DC

Let $\sigma_{\boxdot}$ be any V & D strategy for $\mathcal{S}_{\boxdot}$. We will construct a similar V & D strategy σ for $\mathcal{S}$.

First, consider any scenario s, and any time-point X. The viability of $\sigma_{\boxdot}$ ensures that it satisfies the constraints in $\mathcal{C}_1'$ and $\mathcal{C}_2'$. Therefore, if $X \in \mathcal{T}_s^+$ (i.e., if $s \models L(X)$), then $[\sigma_{\boxdot}(s)]_X \leq h$; otherwise $[\sigma_{\boxdot}(s)]_X \geq h + 1 > h$. In short, $[\sigma_{\boxdot}(s)]_X \leq h \Leftrightarrow s \models L(X)$.

Next, define the strategy σ for $\mathcal{S}$, as follows. For each scenario s, and each $X \in \mathcal{T}_s^+$, let $[\sigma(s)]_X = [\sigma_{\boxdot}(s)]_X$ (i.e., $\sigma(s) = \sigma_{\boxdot}(s)|_{\mathcal{T}_s^+}$). Note that, by the preceding remarks, $|\sigma| \leq h$.

Viability of σ. Let s be any scenario, and $Prj(\mathcal{S}, s) = (\mathcal{T}_s^+, \mathcal{C}_s^+)$ the corresponding projection. By WD1, the constraints in $\mathcal{C}_s^+$ only involve time-points in $\mathcal{T}_s^+$. Thus, the endpoints of each constraint in $\mathcal{C}_s^+$ are executed at the same times by σ and $\sigma_{\boxdot}$ in s. Since $\sigma_{\boxdot}$ is viable, it satisfies each constraint in that projection; hence, so does σ. Thus, σ is viable.

Dynamicity of σ. Let s_1 and s_2 be any scenarios, let X be any time-point in $\mathcal{T}_{s_1}^+$, let $t = [\sigma(s_1)]_X \leq h$, and suppose that $Hist(t, s_1, \sigma) = Hist(t, s_2, \sigma)$. Since $X \in \mathcal{T}_{s_1}^+$, $[\sigma(s_1)]_X = [\sigma_{\boxdot}(s_1)]_X$. Next, let p be any letter appearing in $L(X)$; and let $P?$ be the corresponding observation time-point. By WD1, $L(X)$ is honest; hence, $L(X) \models L(P?)$; whence, $s \models L(X) \models L(P?)$. Therefore, $[\sigma(s_1)]_{P?} = [\sigma_{\boxdot}(s_1)]_{P?}$. Next, by WD2.b, $\mathcal{C}$ includes a constraint of the form $(P? - X \leq -\epsilon, L(X))$ and, since $\sigma_{\boxdot}$ is viable, it follows that $[\sigma_{\boxdot}(s_1)]_{P?} < [\sigma_{\boxdot}(s_1)]_{P?} + \epsilon \leq [\sigma_{\boxdot}(s_1)]_X = t$. Then, since $[\sigma(s_1)]_{P?} = [\sigma_{\boxdot}(s_1)]_{P?} < t$, it follows that $(p, s_1(p))$ appears in $Hist(t, s_1, \sigma)$. Since this holds for each p in $L(X)$, it follows that $Hist(t, s_1, \sigma) \models L(X)$. Since $Hist(t, s_2, \sigma) = Hist(t, s_1, \sigma)$, it follows that $Hist(t, s_2, \sigma) \models L(X)$ and, hence, that $s_2 \models L(X)$ (i.e., $X \in \mathcal{T}_{s_2}^+$). Therefore, $[\sigma(s_2)]_X = [\sigma_{\boxdot}(s_2)]_X$. Now, if $Hist_{\boxdot}(t, s_1, \sigma_{\boxdot}) \neq Hist(t, s_1, \sigma)$, there must be some observation time-point $P? \notin \mathcal{T}_{s_1}^+$ that $\sigma_{\boxdot}$ executes in scenario s_1 at some time *before* $t \leq h$. But the lower-bound constraints in $\mathcal{C}_2'$ ensure that this can only happen if $s_1 \models L(P?)$, which contradicts that $P? \notin \mathcal{T}_{s_1}^+$. Thus, $Hist_{\boxdot}(t, s_1, \sigma_{\boxdot}) = Hist(t, s_1, \sigma)$. Similarly, $Hist_{\boxdot}(t, s_2, \sigma_{\boxdot}) = Hist(t, s_2, \sigma)$. But then the dynamicity of $\sigma_{\boxdot}$ ensures that $[\sigma_{\boxdot}(s_2)]_X = t$ and, hence, that $[\sigma(s_2)]_X = t$. ◀

4.2 The Second Translation from CSTN to CSTN$_{\boxdot}$

Unlike the first translation from CSTN to CSTN$_{\boxdot}$, the second translation, defined below, does not insert any extra edges. For convenience, we use the same notation as in the preceding section (i.e., in this section, CSTN$_{\boxdot}$ refers to the following definition).

▶ **Definition 20** (Reducing a CSTN to a CSTN$_\boxdot$)**.** Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P\rangle$ be a well defined CSTN. The reduction of $\mathcal{S}$ is the CSTN$_\boxdot$, $\mathcal{S}_\boxdot = \langle \mathcal{T}, \mathcal{C}, \mathcal{OT}, \mathcal{O}, P\rangle$.

For a well defined CSTN, we can define a partial order among propositional letters, as follows. For any $p, q \in P$, we write $p \prec_L q$ if p (or $\neg p$) appears in $L(Q?)$, where L is the function that assigns labels to time-points. In addition, if, for each observation time-point $P?$, the label $L(P?)$ is honest, then we say that L is honest. Using this notation, it follows that if L is honest and $p \prec_L q$, then $L(Q?) \models L(P?)$. In what follows, it is necessary to show that the $\prec_L$ relation is acyclic for well defined CSTNs for which $\mathcal{S}_\boxdot$ is DC.

▶ **Lemma 21.** *If $\mathcal{S}$ is well defined and $\mathcal{S}_\boxdot$ is DC, then $\prec_L$ is acyclic.*

Proof. Suppose that $p_1 \prec_L \cdots \prec_L p_k = p_1$ is a cycle in $\prec_L$. By WD2.a, L is honest and, therefore, $L(P_k?) \models L(P_{k-1}?) \models \ldots \models L(P_2?) \models L(P_1?)$. Since $p_k = p_1$, it follows that $L(P_1?) = \ldots = L(P_k?)$. For convenience, let $\ell = L(P_1?) = \ldots = L(P_k?)$. By WD2.b, $\mathcal{S}$ (and hence $\mathcal{S}_\boxdot$) contains constraints of the form, $(P_{i+1}? - P_i? \leq -\epsilon_i, \ell)$, for every $i = 1, \ldots, k-1$, forming a negative cycle with the consistent label ℓ. Since such a cycle cannot be satisfied in any scenario s for which $s \models \ell$, $\mathcal{S}_\boxdot$ must not be DC, which is a contradiction. ◀

For a CSTN $\mathcal{S}$, it may be that in some scenarios some propositional variables are not observed because their corresponding observation time-points are not executed. For example, in the CSTN from Fig. 1, $Q?$ is not executed in either of the scenarios $\neg pq$ or $\neg p \neg q$ because $L(Q?) = p$. In general, in such cases, there may be a family of scenarios that are equivalent in that they differ only in the values they assign to propositional letters that are not observed. Below, we define a *canonical scenario* to be a unique representative for such a family of scenarios. The canonical scenario is the unique scenario from the family that assigns a value of $\bot$ (i.e., false) to each propositional letter that is not observed when executing $\mathcal{S}$ in scenarios from that family. For example, the canonical scenario for $\{\neg pq, \neg p \neg q\}$ is $\neg p \neg q$.

▶ **Definition 22** (Canonical Scenario)**.** For any scenario s, define the *canonical* scenario $\hat{s}$ as follows. For any $p \in P$, if $s \models L(P?)$, let $\hat{s}(p) = s(p)$; otherwise, let $\hat{s}(p) = \bot$.

Note that if s and $\hat{s}$ disagree on some p, then $s \not\models L(P?)$ (i.e., s and $\hat{s}$ differ only on variables that cannot be observed in the scenario s). However, the converse need not hold (i.e., it may happen that $s \not\models L(P?)$, yet s and $\hat{s}$ happen to agree on p).

▶ **Lemma 23.** *If $\prec_L$ is acyclic and L is honest, then $s \models L(P?)$ if and only if $\hat{s} \models L(P?)$.*

Proof. There are two cases to consider.

1. $\hat{s} \not\models L(P?) \Rightarrow s \not\models L(P?)$**.**
   Suppose that $\hat{s} \not\models L(P?)$, but $s \models L(P?)$. Then s and $\hat{s}$ disagree on some $q \in L(P?)$. By the honesty of $L(P?)$, it follows that $L(P?) \models L(Q?)$ and, therefore, that $s \models L(P?) \models L(Q?)$. However, since s and $\hat{s}$ disagree on q, the definition of $\hat{s}$ implies that $s \not\models L(Q?)$, which is a contradiction.
2. $\hat{s} \models L(P?) \Rightarrow s \models L(P?)$**.**
   Suppose that $\hat{s} \models L(P?)$, but $s \not\models L(P?)$, where $P?$ (and hence p) is chosen minimally with this property with respect to the ordering $\prec_L$. Let $q \in L(P?)$ be arbitrary. Thus, $q \prec_L p$. By the honesty of $L(P?)$, it follows that $L(P?) \models L(Q?)$. Therefore, $\hat{s} \models L(P?) \models L(Q?)$ and, hence, $\hat{s} \models L(Q?)$. But, then, by the minimality of p, it follows that $s \models L(Q?)$. Since $q \in L(P?)$ was chosen arbitrarily, we have that $s \models L(P?)$, which is a contradiction.

◀

The following lemma states that, if a CSTN $\mathcal{S}$ is well defined, then any scenario s and its corresponding canonical scenario $\hat{s}$ determine the same projection.

▶ **Lemma 24.** *If $\mathcal{S}$ is a CSTN for which $\prec_L$ is acyclic and all labels on time-points and constraints are honest, then for any scenario s and any label ℓ, s and $\hat{s}$ must assign the same truth value to ℓ. As a result, $Prj(\mathcal{S}, s) = Prj(\mathcal{S}, \hat{s})$ (i.e., $\mathcal{T}_s^+ = \mathcal{T}_{\hat{s}}^+$ and $C_s^+ = C_{\hat{s}}^+$).*

Proof. Let ℓ be any label on a time-point or constraint such that $s(\ell) \neq \hat{s}(\ell)$. Then there exists some p that appears in ℓ such that $s(p) \neq \hat{s}(p)$. Then, by the definition of $\hat{s}$, $s \not\models L(P?)$. Therefore, by Lemma 23, $\hat{s} \not\models L(P?)$. But the honesty of ℓ implies that $\ell \models L(P?)$. Therefore, it follows that $s \not\models \ell$ and $\hat{s} \not\models \ell$. But then $s(\ell) = \bot = \hat{s}(\ell)$, which is a contradiction. ◀

It is now possible to apply the obtained results to show the relationship between the dynamic consistency of a CSTN and that of its corresponding streamlined CSTN$_\boxdot$.

▶ **Lemma 25.** *If $\mathcal{S}$ is a well defined CSTN, and $\mathcal{S}_\boxdot$ is DC, then $\mathcal{S}$ is DC.*

Proof. Let $\sigma_\boxdot$ be any viable and dynamic strategy for $\mathcal{S}_\boxdot$. Let the strategy σ for $\mathcal{S}$ be defined as follows. For any scenario $s \in \mathcal{I}$ and any time-point $X \in \mathcal{T}_s^+$, let $[\sigma(s)]_X = [\sigma_\boxdot(\hat{s})]_X$. We need only show that σ is viable and dynamic for $\mathcal{S}$.

Viability. First, note that the conditions of Lemma 21 hold; therefore, $\prec_L$ must be acyclic. Next, since $\mathcal{S}$ is well defined, the labels on all time-points and constraints in $\mathcal{S}$ must be honest; hence, the conditions of Lemma 24 are satisfied. Thus, s and $\hat{s}$ must agree on the truth value of each label on any time-point or constraint in $\mathcal{S}$ (and hence in $\mathcal{S}_\boxdot$).

Suppose that σ violates some constraint, $(Y - X \leq \delta, \ell)$. Then there is some scenario s such that $[\sigma(s)]_Y - [\sigma(s)]_X > \delta$ and $s \models \ell$. But then the definition of σ implies that $[\sigma_\boxdot(\hat{s})]_Y - [\sigma_\boxdot(\hat{s})]_X > \delta$; and Lemma 24 gives that $\hat{s} \models \ell$. Together, these contradict that $\sigma_\boxdot$ is viable.

Dynamicity. Suppose that $X \in \mathcal{T}_{s_1}^+$ (i.e., $s_1 \models L(X)$), $t = [\sigma(s_1)]_X$, and $Hist(s_1, t, \sigma) = Hist(s_2, t, \sigma)$. We must show that $X \in \mathcal{T}_{s_2}^+$ (i.e., $s_2 \models L(X)$) and $[\sigma(s_2)]_X = t$.

Toward that end, let p be any letter that appears in $L(X)$. Since L is honest, it follows that $L(X) \models L(P?)$. Thus, $s_1 \models L(P?)$ (i.e., $P? \in \mathcal{T}_{s_1}^+$); hence, by Lemma 24, $\hat{s}_1 \models L(P?)$. Next, by WD2.b, $\mathcal{S}$ (and hence $\mathcal{S}_\boxdot$) must contain a constraint of the form, $(P? - X \leq -\epsilon, L(X))$. And, since $\sigma_\boxdot$ is viable, it follows that $[\sigma(s_1)]_{P?} = [\sigma_\boxdot(\hat{s}_1)]_{P?} < [\sigma_\boxdot(\hat{s}_1)]_X = [\sigma(s_1)]_X = t$. Since $s_1 \models L(P?)$, it must be that p appears in $Hist(s_1, t, \sigma) = Hist(s_2, t, \sigma)$. And, since p was chosen arbitrarily in $L(X)$, it follows that $Hist(s_1, t, \sigma) = Hist(s_2, t, \sigma) \models L(X)$. Therefore, $s_2 \models L(X)$ (i.e., $X \in \mathcal{T}_{s_2}^+$).

Finally, suppose that $Hist(\hat{s}_1, t, \sigma_\boxdot) \neq Hist(\hat{s}_2, t, \sigma_\boxdot)$. But then there must be some time $t' < t$ at which one of the following holds: (1) one of the schedules, $\sigma_\boxdot(\hat{s}_1)$ or $\sigma_\boxdot(\hat{s}_2)$, executes some observation time-point $Q?$ at t', while the other does not; or (2) both schedules execute some observation time-point $Q?$ at t', but yield different values for q. Without loss of generality, choose t' to be the earliest time at which one of the above conditions hold. Then (1) is impossible, because $Hist(\hat{s}_1, t', \sigma_\boxdot) = Hist(\hat{s}_2, t', \sigma_\boxdot)$ and $\sigma_\boxdot$ is dynamic. To show that property (2) is impossible, first consider the case where q appears in $L(X)$. Then $s_1(q) = s_2(q)$, which implies that $\hat{s}_1(q) = \hat{s}_2(q)$, contradicting the choice of $Q?$. But if q does not appear in $L(X)$, then $\hat{s}_1(q) = \bot = \hat{s}_2(q)$ by definition of $\hat{s}_1$ and $\hat{s}_2$, another contradiction. Therefore, $Hist(\hat{s}_1, t, \sigma_\boxdot) = Hist(\hat{s}_2, t, \sigma_\boxdot)$ which, by the dynamicity of $\sigma_\boxdot$ implies that $[\sigma_\boxdot(\hat{s}_2)]_X = t$, which in turn implies that $[\sigma(s_2)]_X = t$. ◀

Now, it is possible to show the main result of the section.

▶ **Theorem 26.** *Let $\mathcal{S}$ be any well defined CSTN. Then $\mathcal{S}_{\boxdot}$ (i.e., the reduction of $\mathcal{S}$ to a CSTN$_{\boxdot}$) is equivalent to $\mathcal{S}$, in the sense that $\mathcal{S}$ is DC if and only if $\mathcal{S}_{\boxdot}$ is DC.*

Proof. The $\Longrightarrow$ direction is already proven in Theorem 19. Indeed, using the first translation requires *more* constraints to be satisfied in $\mathcal{S}_{\boxdot}$. The $\Longleftarrow$ direction is given by Lemma 25. ◀

5 Empirical Evaluation

When applied to Streamlined Conditional Simple Temporal Networks, the constraint-propagation rules used by the DC-checking algorithm of Hunsberger et al. [15, 12] become simpler. That algorithm checks whether an input network is DC by exhaustively propagating labeled constraints and then verifying that the propagated network does not contain a negative cycle with a consistent label. However, the applicability conditions for the constraint-propagation rules, and the labels generated by those rules, depend on time-point labels and the $\prec_L$ relation. Therefore, if the input CSTN has no labels on its time-points (i.e., if it is a CSTN$_{\boxdot}$), then the algorithm can avoid dealing with such complications.

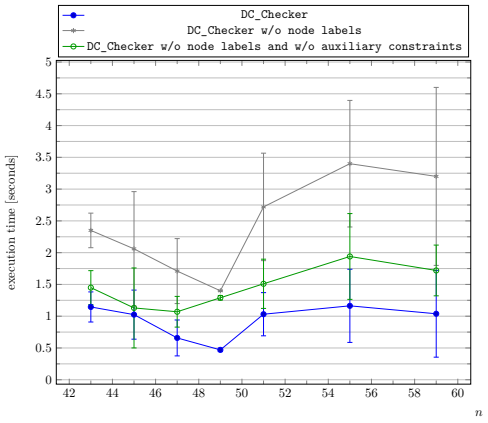
- In this section, the original DC-checking algorithm that applies to any CSTN $\mathcal{S}$ shall be called `DC_Checker`, and the "simplified" algorithm that applies only to a CSTN$_{\boxdot}$ shall be called `DC_CheckerWONodeLabels`.

This section presents an empirical comparison of the performance of `DC_Checker` and `DC_CheckerWONodeLabels` on instances of the benchmarks proposed in prior work [12]. For each CSTN $\mathcal{S}$, `DC_Checker` is run on $\mathcal{S}$, while `DC_CheckerWONodeLabels` is run on the streamlined CSTN $\mathcal{S}_{\boxdot}$, using the simplified propagation rules. Recall that two translations from $\mathcal{S}$ to $\mathcal{S}_{\boxdot}$ were introduced in Section 4, one of which involves the auxiliary constraints in the sets, $\mathcal{C}'_1$ and $\mathcal{C}'_2$. This section reports results from running `DC_CheckerWONodeLabels` on both versions of $\mathcal{S}_{\boxdot}$.

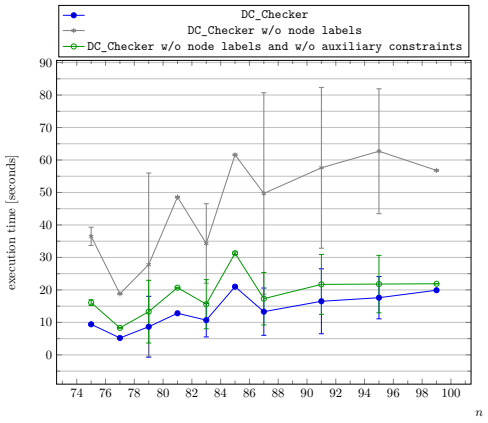
First, we briefly recall that the benchmarks contain CSTN instances obtained from random workflow schemata generated by the ATAPIS toolset [16]. For each $N \in \{10, 20, 30, 40\}$, a class of at least 120 workflow graphs were randomly generated by setting the number of activities to N, the probability 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 50. As a result, all edge-weights were at most 10^4. Then, each workflow graph was translated into an equivalent CSTN as proposed by Combi et al. [5]. 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 n nodes, where $(2N + 2) \leq n \leq (5N + 2)$. There are 4 benchmarks, each containing at least 60 dynamically consistent CSTNs and 60 non-dynamically consistent CSTNs, relating to workflow graphs of the same class. In order to simplify the comparison, for each benchmark, the number of observation time-points in the network, $|\mathcal{P}|$, has been fixed with respect to the class of workflows, as follows.

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

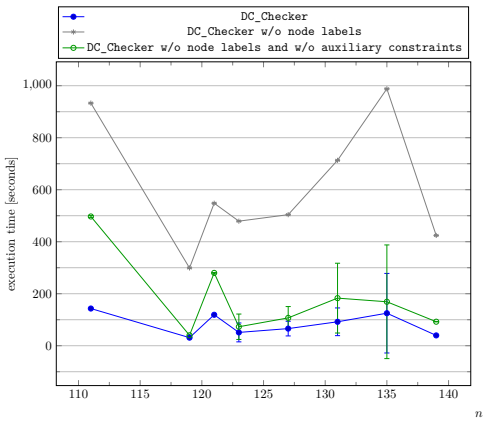
Since non-DC networks were regularly solved one to two orders of magnitude faster than similarly sized DC networks, the rest of this section focuses on the results for the DC networks.



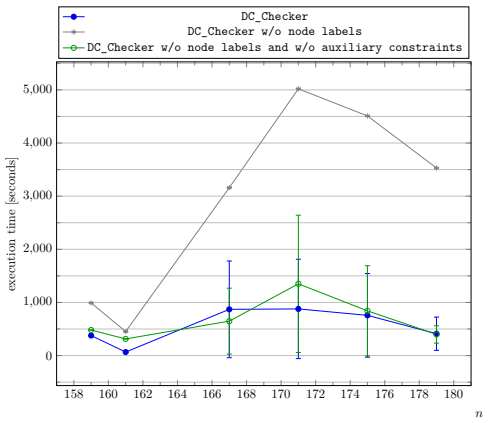
(a) Benchmark $N = 10, |\mathcal{P}| = 3$.



(b) Benchmark $N = 20, |\mathcal{P}| = 5$.



(c) Benchmark $N = 30, |\mathcal{P}| = 7$.
For `DC_Checker w/o node labels` values, the standard deviation has been omitted to have a better scale of the diagram.



(d) Benchmark $N = 40, |\mathcal{P}| = 9$.
For `DC_Checker w/o node labels` values, the standard deviation has been omitted to have a better scale of the diagram.

Figure 3 Execution time vs. number of time-points n.

Algorithms and procedures necessary for this evaluation were implemented in Java and executed on a JVM 8 in a Linux machine with two AMD Opteron 4334 CPUs and 64GB of RAM. The code is freely available [17].

The results shown in Figure 3 demonstrate that, in general, the original CSTN DC-checking algorithm `DC_Checker` has the best performance in almost all instances. Indeed, taking node labels into account allows the algorithm to avoid the propagation of some auxiliary values (in the case of streamlined CSTNs with auxiliary constraints from the sets $\mathcal{C}'_1$ and $\mathcal{C}'_2$) or non-coherent or non-honest ones (in the case of streamlined CSTNs without auxiliary constraints).

We have verified that these kinds of values can be quite numerous and that, for some instances, when they contain the auxiliary constraints from $\mathcal{C}'_1$ and $\mathcal{C}'_2$, the execution time of `DC_CheckerWONodeLabels` can be two or three orders of magnitude greater than the execution time of `DC_Checker` on the corresponding CSTNs.

On the other hand, the performance difference between `DC_CheckerWONodeLabels` and `DC_Checker` decreases as the number of nodes increases. We verified that the original

algorithm continues to propagate fewer labeled values than the streamlined version, but such differences become smaller. Therefore, the time required by the original algorithm to stop non-coherent or non-honest labeled values must become more or less equal to the time required to propagate them as done by the simpler algorithm.

We have also evaluated a different implementation of the streamlined version for checking if it was possible to avoid the propagation of useless labeled values. In this new implementation, part ot the information given by node labels is rebuilt dynamically and exploited to "clean" some labels. We verified that while the number of useless labeled values can be reduced, the computation time still remains the same due to the extra time required by the added code. In other words, the overall performance of that implementation is no better than that of `DC_CheckerWONodeLabels`.

6 Related Work

There are many proposals in the literature for ways of extending the expressiveness of the STN model. Below, we summarize the main results about CSTNs and related models.

Tsamardinos et al. [18] defined the Conditional Simple Temporal Problem (CTP) as that of determining whether a given CSTN admits a viable and dynamic execution strategy. (The CSTN acronym was introduced later.) In their work, propositional labels are associated only with time-points, not constraints. They also informally specified some reasonableness properties that any CSTN ought to satisfy. Although they showed how to solve the CTP by encoding it as a meta-level Disjunctive Temporal Problem (DTP) and feeding it to an off-the-shelf solver, that approach is not practical because the CTP-to-DTP encoding has exponential size and, on top of that, the DTP solver runs in exponential time. To our knowledge, this approach has never been implemented or empirically evaluated.

Later, Hunsberger et al. [14, 15] defined CSTNs (separate from the CTP) and formalized the well-definedness properties for CSTNs. In their work, both time-points (nodes) and constraints (edges) of a CSTN can have propositional labels that specify the scenarios in which they are applicable. (Allowing constraints to be labeled was inspired by the work of Conrad et al. [8], discussed below.) They showed that the labels must satisfy the well-definedness properties in order to guarantee the existence of a dynamic execution strategy. They also presented a sound-and-complete DC-checking algorithm for solving the CTP, and empirically demonstrated its practical performance.

Conrad et al. [8] considered a variant of CSTNs, proposing Drake, a dynamic executive for temporal plans with choice. In their work, the constraints of a temporal plan are labeled as in CSTNs, but the values of propositions (choices) are decided by the executive during run-time, not by the environment.

Cimatti et al. [3, 4] presented a different approach to solving a variety of temporal problems (CSTNs included) in which a temporal network is first translated into an equivalent Timed Game Automaton (TGA) and, then, solved by an off-the-shelf TGA solver. Although this approach is interesting because it shows the relationships between TGAs and a variety of temporal networks – including CSTNs – it has not yet been shown to be practical for solving the CTP.

Comin and Rizzi [7] solved the CTP by converting it into a Mean Payoff Game (MPG). They also introduced a variant of dynamic consistency, called ε-DC, where $\varepsilon > 0$ represents the minimum reaction time of the executive in response to observations. They presented (1) a sharp lower-bounding analysis on the critical value of the reaction time where the CSTN changes from being DC to non-DC, (2) a proof that the CTP is coNP-hard, and (3) the first singly-exponential-time algorithm for solving the CTP.

Hunsberger and Posenato [12] showed how their DC-checking algorithm from earlier work [15] can be extended to check the ε-DC property without incurring any performance degradation. They also introduced four benchmarks for testing DC-checking algorithms.

Hunsberger and Posenato [13] presented another optimization of the approach presented by Cimatti et al. in which the CTP is viewed as a two-player game. Its solution is determined by exploring an abstract game tree to find a "winning" strategy, using Monte Carlo Tree Search and Limited Discrepancy Search to guide its search. An empirical evaluation shows that the new algorithm is competitive with the propagation-based algorithm.

Cairo et al. [1] improved the analysis of the ε-DC property. They showed that if $\varepsilon = 0$ (i.e., if the system can react instantaneously), it is necessary to impose a further condition to avoid a form of instantaneous circularity. In particular, they (1) proposed a new extension of dynamic consistency, called π-DC, suitable for systems that can react instantaneously, (2) showed by a counter-example that π-DC is not equivalent to 0-DC, and (3) proposed a sound-and-complete algorithm for checking the π-DC property having a (pseudo) singly-exponential time complexity in the number of propositional letters.

Cario and Rizzi [2] showed that the CTP is PSPACE-complete.

7 Conclusions and Future Work

This paper presented a new version of CSTNs, named *streamlined Conditional Simple Temporal Networks*, in which propositional labels may appear on constraints, but not on time-points. This change simplifies the definition of the DC property and the specification of propagation rules for the DC-checking algorithm. It also makes proving the soundness and completeness of those rules simpler.

The paper proves that traditional CSTNs can be translated into streamlined CSTNs while preserving the dynamic consistency property. Two translations from CSTNs to streamlined CSTNs were presented. The first generates an equivalent streamlined CSTN in which the information contained in time-point labels is preserved in the form of auxiliary constraints that force time-points in certain scenarios to be executed either before or after a fixed horizon, depending on whether they would be executed or not in the original CSTN. The second translation does not preserve the information in the time-point labels, but provides a simpler, equivalent streamlined CSTN. The drawback of the second translation is that some time-points can be executed in the streamlined CSTN even if they would not be in the original CSTN.

Finally, the paper provided an experimental comparison of two versions of the DC-checking algorithm due to Hunsberger et al. [15]: the original version and a simplified version for streamlined CSTNs. For small CSTNs, the original algorithm shows the best performance; however, the difference in performance between the two versions decreases as the number of time-points increases. During the tests, we verified that the static information given by the time-point labels can limit the propagation of non-coherent/non-honest labels in a significant way making the the DC checking faster. However, that advantage decreases as the number of nodes increases.

It appears that simple heuristics such as one that tries to rebuild dynamically the information given by time-point labels would not be successful for improving the performance of the simplified version of the DC-checking algorithm. Our future work will investigate other methods for improving the performance of the algorithm for streamlined CSTNs in order to make it competitive with the original algorithm on small CSTNs, too.

References

1. Massimo Cairo, Carlo Comin, and Romeo Rizzi. Instantaneous reaction-time in dynamic-consistency checking of conditional simple temporal networks. In *23rd International Symposium on Temporal Representation and Reasoning, TIME 2016*, pages 80–89, 2016. doi:10.1109/TIME.2016.16.
2. Massimo Cairo and Romeo Rizzi. Dynamic controllability of conditional simple temporal networks is PSPACE-complete. In *23rd International Symposium on Temporal Representation and Reasoning, TIME 2016*, pages 90–99, 2016. doi:10.1109/TIME.2016.17.
3. Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Sound and complete algorithms for checking the dynamic controllability of temporal networks with uncertainty, disjunction and observation. In *21st International Symposium on Temporal Representation and Reasoning, TIME 2014*, pages 27–36, 2014. doi:10.1109/TIME.2014.21.
4. Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Dynamic controllability via timed game automata. *Acta Informatica*, 53(6-8):681–722, 2016. doi:10.1007/s00236-016-0257-2.
5. Carlo Combi, Mauro Gambini, Sara Migliorini, and Roberto Posenato. Representing business processes through a temporal data-centric workflow modeling language: An application to the management of clinical pathways. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(9):1182–1203, 2014. doi:10.1109/TSMC.2014.2300055.
6. Carlo Comin. *Complexity in Infinite Games on Graphs and Temporal Constraint Networks*. PhD thesis, University of Trento and Universite Paris-Est, 2017.
7. Carlo Comin and Romeo Rizzi. Dynamic consistency of conditional simple temporal networks via mean payoff games: A singly-exponential time dc-checking. In *22nd International Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 19–28, 2015. doi:10.1109/TIME.2015.18.
8. Patrick R. Conrad and Brian C. Williams. Drake: An efficient executive for temporal plans with choice. *Journal of Artificial Intelligence Research*, 42(1):607–659, 2011. doi:10.1613/jair.3478.
9. Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. doi:10.1016/0004-3702(91)90006-6.
10. Alfonso Gerevini, Anna Perini, and Francesco Ricci. Incremental algorithms for managing temporal constraints. Technical Report IRST-9605-07, IRST, 1996.
11. Luke Hunsberger. Efficient execution of dynamically controllable Simple Temporal Networks with Uncertainty. *Acta Informatica*, 53(2):89–147, 2015. doi:10.1007/s00236-015-0227-0.
12. Luke Hunsberger and Roberto Posenato. Checking the dynamic consistency of conditional simple temporal networks with bounded reaction times. In *26th International Conference on Automated Planning and Scheduling (ICAPS 2016)*, pages 175–183, 2016. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/view/13108.
13. Luke Hunsberger and Roberto Posenato. A new approach to checking the dynamic consistency of conditional simple temporal networks. In *Principles and Practice of Constraint Programming (CP 2016)*, volume 9892 of *LNCS*, pages 268–286, 2016. doi:10.1007/978-3-319-44953-1_18.
14. Luke Hunsberger, Roberto Posenato, and Carlo Combi. The dynamic controllability of conditional stns with uncertainty. In *Workshop on Planning and Plan Execution for Real-World Systems (PlanEx) at ICAPS 2012*, pages 1–8, June 2012. URL: http://arxiv.org/abs/1212.2005.
15. Luke Hunsberger, Roberto Posenato, and Carlo Combi. A sound-and-complete propagation-based algorithm for checking the dynamic consistency of conditional simple temporal

networks. In *22nd International Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 4–18, 2015. doi:10.1109/TIME.2015.26.

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

17 Roberto Posenato. A CSTN(U) consistency check algorithm implementation in Java, June 2017. URL: http://profs.scienze.univr.it/~posenato/software/cstnu.

18 Ioannis Tsamardinos, Thierry Vidal, and Martha E. Pollack. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints*, 8(4):365–388, 2003. doi:10.1023/A:1025894003623.

A Appendix: Proof of Lemma 17

▶ **Lemma 17** (Tighter upper bound; rational weights). *Let $\mathcal{S}$ be a DC CSTN with n time-points and* rational *weights. If M is the maximum absolute value of any* negative *edge in $\mathcal{S}$, then the network obtained by constraining every time-point in $\mathcal{S}$ to occur before time Mn is DC.*

Proof. First, since there are at most $(n^2)(2^k)$ edge weights, each edge weight can be expressed as a fraction involving the least common denominator among the edge weights. As a result, without loss of generality, we may henceforth assume that all edge weights are integers.

Let $\mathcal{C}^*$ be the set of labeled edges obtained by exhaustively applying the sound-and-complete constraint-propagation rules presented by Hunsberger et al. [15]. Given the assumptions that (1) all edge weights are integers, and (2) the network is DC, the constraint propagation must terminate. Hence, $\mathcal{C}^*$ is well defined and contains only finitely many edges. Furthermore, each edge in $\mathcal{C}^*$ can be derived by a finite number of applications of the constraint-propagation rules. For convenience, we shall refer to $\mathcal{C}$ as the set of *original* edges, and $\mathcal{C}^*$ as the set of *derived* edges.

Fix $Z = 0$ and let $\mathcal{U} = \mathcal{T}\backslash\{Z\}$ be the set of as-yet-unexecuted time-points. Prior to executing the time-points in $\mathcal{U}$, there have been no observations and, thus, the *initial partial scenario* is the empty scenario, represented by $\boxdot$. For each $Y \in \mathcal{U}$, its *effective lower bound* (ELB) with respect to the empty scenario, defined by Hunsberger et al., is given by: $ELB(Y, \boxdot) = \max\{\delta \mid (Y \geq \delta, \ell) \in \mathcal{C}^*\}$. Let $\lambda = \min\{ELB(Y, \boxdot) \mid Y \in \mathcal{U}\}$ be the minimum ELB of any as-yet-unexecuted time-point. Let $X \in \mathcal{U}$ be any time-point such that $ELB(X, \boxdot) = \lambda$. (It does not matter if there happen to be multiple such time-points.) We aim to show that $\lambda \leq M$. Therefore, we assume that $\lambda > M$ and seek a contradiction.

By construction, $\lambda \leq ELB(Y, \boxdot)$ for each $Y \in \mathcal{U}$. In addition, the *Spreading Lemma* (from Hunsberger et al.) ensures that for each $Y \in \mathcal{U}$, there is an edge from Y to Z labeled by $\langle -\delta_Y, \boxdot\rangle$, for some $\delta_Y \geq \lambda$.

Given the definition of M, the value, $ELB(X, \boxdot) = \lambda > M$, cannot be due to an *original* edge from X to Z of length $-\lambda < -M$. Instead, it must be due to an edge that has been *derived* by one or more applications of the various constraint-propagation rules. Among all of the derivations used to generate the edges in $\mathcal{C}^*$, generated in some arbitrary order, let D be the *first* derivation that results in an edge from X to Z whose weight equals $-\lambda$.

The following argument focuses on the rule applications in the derivation D that involve the zero time-point Z. (There may be rule applications in D that do not involve Z, but they will not be relevant to the argument that follows.) In this narrow setting, the six constraint-propagation rules presented by Hunsberger et al. ($LP, R_0, R_3^*, qLP, qR_0$ and qR_3^*) can be represented by the three rules shown in Table 1.‡

‡ The labels shown in Table 1 do not play a big role in the proof. However, for completeness, they are

Table 1 Constraint-propgation rules used in the proof of Lemma 17.

(LP/qLP)	$A \xrightarrow{\langle u,\alpha\rangle} B \xrightarrow{\langle v,\beta\rangle} Z$; $A \dashrightarrow Z$ labeled $\langle u+v,\gamma\rangle$	$\alpha\beta$ consistent or ($u < 0$ and $v < 0$); $\gamma = (\alpha \star \beta_p)'$.
(R_0/qR_0)	$P? \xrightarrow{\langle w,\alpha\rangle} Z$; $P? \dashrightarrow Z$ labeled $\langle w,(\alpha_p)'\rangle$	$w < 0$.
(R_3^*/qR_3^*)	$P? \xrightarrow{\langle w,\alpha\rangle} Z \xleftarrow{\langle v,\beta\rangle} C$; $P? \dashrightarrow Z$ labeled $\langle m,\gamma\rangle$	$w < 0$, $m = \max\{w,v\}$, and $\gamma = (\alpha \star \beta_p)'$.

Claim: No finite sequence of rule applications involving any of the six constraint-propagation rules from Hunsberger et al. can generate a *shortest* edge from any time-point $Y \in \mathcal{U}$ to Z whose weight is less than or equal to $-\lambda$.

Proof of Claim. For the base case, we note that the definition of M ensures that no *original* edge in $\mathcal{C}$ can have weight less than or equal to $-\lambda < -M$. For the inductive case, consider an arbitrary edge from Y to Z whose label is $\langle u,\alpha\rangle$. Suppose that this edge is a shortest edge among all edges from Y to Z whose label is α (or more general than α). Finally, suppose that all prior edges encountered during the derivation of this edge satisfy the claim. Note that the final rule application that generates the edge from Y to Z must be one of the three rules shown in Table 1. We address each in turn.

- (LP/qLP). In this case, A in the top row of Table 1 plays the role of Y, and $y = u + v \leq -\lambda < -M$ is the weight of a shortest edge from A to Z among those edges labeled by γ (or some more general label). First consider the possibility that $u \geq 0$. In that case, the weight v of the edge from B to Z satisfies: $v \leq u + v = -\lambda < -M$. By the inductive hypothesis, this cannot be a shortest edge from B to Z labeled by β (or some more general label). But then the same rule application, using a shorter edge from B to Z, would generate a shorter edge from Y to Z whose label is α (or more general than α), contradicting that the first edge from Y to Z was shortest. On the other hand, if $u < 0$, then $-\lambda = u + v < v$. In that case, the Spreading Lemma ensures that there is an edge from B to Z labeled by some $\langle -\delta, \boxdot\rangle$, where $-\delta \leq -\lambda$. But then the same rule application, using this stronger edge from B to Z would generate an edge from Y to Z whose weight is $-\delta + u \leq -\lambda + u < -\lambda$, another contradiction.
- (R_0/qR_0). In this case, $P?$ in the middle row of Table 1 plays the role of Y and $y = w \leq -\lambda < -M$ is the weight of a shortest edge from $P?$ to Z among those labeled by $(\alpha_p)'$ (or some more general label). By the inductive hypothesis, the edge from $P?$ to Z labeled by $\langle w,\alpha\rangle$, which also has the weight $y = w \leq -\lambda < -M$, cannot be a shortest edge from $P?$ to Z labeled by α (or some more general label). But then replacing this

shown in full detail. The notation in the table is a slight simplification of that used by Hunsberger et al. First, for any label ℓ, the label ℓ' is that obtained by removing from ℓ any *children* of any q-literals that appear in ℓ. Second, for any propositional letter p, and any label ℓ, the label ℓ_p is that obtained by removing any occurrence of p or any of its children from ℓ. Finally, the $\star$ operator is a *commutative* operator that extends the conjunction of literals as follows. For any propositional letter p, $p \star \neg p = p? \star p = p? \star \neg p = p? \star p? = p?$. The $\star$ operator is then extended to labels by applying it in pairwise fashion to like literals from each operand. For example, $(abcd) \star (a(\neg b)(c?)(\neg e)) = a(b?)(c?)d(\neg e)$.

edge with a shorter one would generate a shorter edge from $P?$ to Z labeled by $(\alpha_p)'$, a contradiction.

- (R^*_3/qR^*_3). In this case, C in the bottom row of Table 1 plays the role of Y and $y = m = \max\{w, v\} \leq -\lambda < -M$ is the weight of a shortest edge from C to Z among those labeled by γ or some more general label. There are three cases to consider:
  1. $w < v$. Here, $m = \max\{w, v\} = v \leq -\lambda < -M$. By the inductive hypothesis, the edge from C to Z labeled by $\langle v, \beta\rangle$ cannot be a shortest such edge. But then replacing it with a shorter edge, say one labeled by $\langle v - \epsilon, \beta\rangle$, where $\epsilon > 0$, would cause the corresponding application of R^*_3/qR^*_3 to generate a shorter edge from C to Z labeled by γ (or some more general label), a contradiction. (The weight of the resulting edge would be $v - \epsilon'$, where $\epsilon' = \min\{\epsilon, v - w\} > 0$.)
  2. $v < w$. Here, $m = \max\{w, v\} = w \leq -\lambda < -M$. By the inductive hypothesis, the edge from $P?$ to Z labeled by $\langle w, \alpha\rangle$ cannot be a shortest such edge. But then replacing it with a shorter edge, say one labeled by $\langle w - \epsilon, \alpha\rangle$, where $\epsilon > 0$, would cause the corresponding application of R^*_3/qR^*_3 to generate a shorter edge from C to Z labeled by γ (or some more general label), a contradiction. (The weight of the resulting edge would be $w - \epsilon'$, where $\epsilon' = \min\{\epsilon, w - v\} > 0$.)
  3. $w = v$. Here, $m = \max\{w, v\} = w = v \leq -\lambda < -M$. By the inductive hypothesis, neither the edge from $P?$ to Z labeled by $\langle w, \alpha\rangle$, nor the edge from C to Z labeled by $\langle w, \beta\rangle$, can be shortest such edges. Thus, each can be replaced by a corresponding edge with a shorter weight, say, by using the labeled values $\langle w - \epsilon_1, \alpha\rangle$ and $\langle v - \epsilon_2, \beta\rangle$, respectively. But then the corresponding application of R^*_3/qR^*_3 would generate an edge from C to Z whose weight was $v - \epsilon < v$, where $\epsilon = \min\{\epsilon_1, \epsilon_2\}$. That is a contradiction.

Thus, the claim is proven. ◀

From the claim, it follows that no finite sequence of rule applications can generate an edge from X to Z of length $-\lambda < -M$, which contradicts the choice of X. Therefore, it must be that $\lambda \leq M$. Thus, the earliest-first strategy executes X at time λ. Equivalently, we may introduce the constraints, $X - Z \leq \lambda$ and $Z - X \leq -\lambda$ (i.e., $X = \lambda$).

At this point, the time-points X and Z form a *rigid component* [10]. As a result, by re-orienting all edges involving Z to instead involve X (adjusting the weights accordingly), Z can be effectively removed from the network. Afterward, rename X as Z and observe that the resulting network is DC with the same (or smaller) value of M, and one fewer time-point. By induction we get a network where each successive time-point is executed no more than M after the previous one, which gives the desired result. ◀

Note: We conjecture that Lemma 17 also holds for real-valued weights.