Dynamic Controllability Made Simple

Massimo Cairo[1] and Romeo Rizzi[2]

1 **Department of Mathematics, University of Trento, Trento, Italy; and Department of Computer Science, University of Verona, Verona, Italy**
massimo.cairo@unitn.it, massimo.cairo@univr.it

2 **Department of Computer Science, University of Verona, Verona, Italy**
romeo.rizzi@univr.it

Abstract

Simple Temporal Networks with Uncertainty (STNUs) are a well-studied model for representing temporal constraints, where some intervals (contingent links) have an unknown but bounded duration, discovered only during execution. An STNU is dynamically controllable (DC) if there exists a strategy to execute its time-points satisfying all the constraints, regardless of the actual duration of contingent links revealed during execution.

In this work we present a new system of constraint propagation rules for STNUs, which is sound-and-complete for DC checking. Our system comprises just three rules which, differently from the ones proposed in all previous works, only generate unconditioned constraints. In particular, after applying our sound rules, the network remains an STNU in all respects. Moreover, our completeness proof is short and non-algorithmic, based on the explicit construction of a valid execution strategy. This is a substantial simplification of the theory which underlies all the polynomial-time algorithms for DC-checking.

Our analysis also shows: (1) the existence of late execution strategies for STNUs, (2) the equivalence of several variants of the notion of DC, (3) the existence of a fast algorithm for real-time execution of STNUs, which runs in $O(KN)$ total time in a network with $K \geq 1$ contingent links and $N \geq K$ time points, considerably improving the previous $O(N^3)$-time bound.

1998 ACM Subject Classification I.2.4 Knowledge Representation Formalisms and Methods, I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases Simple Temporal Network with Uncertainty, Dynamic controllability

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.8

1 Introduction

A Simple Temporal Network (STN) is a model for representing temporal constraints: it comprises a set of time-points $\{P, Q, R, \dots\}$ and a collection of binary difference constraints of the form $Q \leq P + w$ with $w \in \mathbb{R}$. A planning agent wants to schedule the execution of time-points, i.e., assign a real value to each variable $P, Q, R, \dots$ representing its execution time, so that all the constraints in the network are satisfied.

Simple Temporal Networks with Uncertainty (STNUs) extend STNs to incorporate uncertainty in the duration of some time intervals. In an STNU some of the time-points, called contingent time-points, are not under the control of the planning agent but are executed by the environment. The execution time of a contingent time-point C is regulated by a contingent link (A, l, u, C), whose meaning is that C will be executed some time $\Delta \in [l, u]$ after the time-point A. The value Δ is called the duration of the contingent link, and is unknown to the planner until C is actually executed. An STNU is said to be dynamically controllable (DC) if the agent holds a strategy to execute all the time-points in the network,

such that all constraints are satisfied regardless of the duration of the contingent links. This strategy has to be dynamic, in the sense that the execution time of a time-point X can only depend on the duration of contingent links (A, l, u, C) whose contingent time-point C is executed before X.

Two main problems are considered: checking whether a given network is DC and, if this is the case, executing its time-points in real-time, reacting dynamically to the durations revealed by the environment.

Previous work. Remarkably, it is possible to check in polynomial time whether a given network is DC. The best upper bound on the running time, for a network with N time-points, has been improved from pseudo-polynomial [10] (an "incremental" algorithm proposed in [15], as fixed in [11], also runs in pseudo-polynomial time), to $O(N^5)$ [9], $O(N^4)$ [7] (an incremental algorithm in [12] also runs in $O(N^4)$ time) and finally to $O(N^3)$ [8] (an incremental algorithm in [13] also runs in $O(N^3)$ time). The key tool to achieve a polynomial-time algorithm is propagation of binary constraints, i.e., generation of new constraints from existing ones according to some sound rules.

A system of constraint propagation rules has been proposed first by Morris, Muscettola and Vidal [10], and proven to be sound and complete. Soundness means that the system only generates constraints which must be satisfied by any dynamic execution strategy, as a logical consequence of existing constraints. Completeness means that, if, at some point, no tighter constraints can be generated by the rules, then the resulting network is DC. The system, in the revised and simplified version given in [9], consists of four propagation rules which generates both ordinary, unconditioned constraints (i.e., having the same form as input constraints) and conditional constraints marked with labels. A fifth rule transforms labeled constraints into ordinary constraints, when some conditions apply. We refer to this system of rules as MMV, and provide an illustration it in Table 1. The proof of completeness of MMV relies on the description of an algorithm, sketched in [10], improved and fully described in [5], which executes the time-points in the network in real-time.

Being such a simple and flexible model, STNUs have been extended in several ways in the literature (see, e.g., [14, 6, 2, 3, 1]). All these extension rely on the theory of DC-checking and executions developed for STNUs, and this theory, both for DC-checking and real-time execution, hinges on the system of constraint propagation rules. Indeed, to the best of our knowledge, all the existing DC-checking and real-time execution algorithms for STNUs work by applying propagation rules, implicitly or explicitly, and their correctness proof is based on the soundness and completeness of this rule system.

Issues with MMV system. The MMV system, as introduced in [10] and improved in [9], has still many points of weakness. First of all, the system itself is rather complex. The main reason is that it generates, besides ordinary constraints, also labeled constraints: these labeled constraints are not part of the STNU model, hence their interpretation is not immediately clear and requires further definitions and explanations. Moreover, the system comprises an arguably large number of rules: four constraint-propagation rules and one label-removal rule, which combine labeled and unlabeled constraints in several possible ways. Besides the intrinsic complexity of the system, the related theory is also somewhat intricate. Specifically, the proof of completeness is based on the description of an execution algorithm which, even after the improvement given in [5], is still quite involved, manipulating the network in non-trivial ways during its execution. Instead, as a completeness proof, it would be desirable to a have a simple, static description of an execution strategy which witnesses the dynamic controllability of a network, when no more application of the rules is possible.

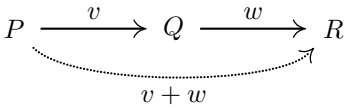
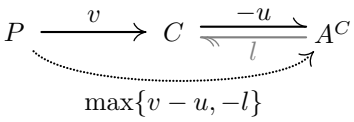
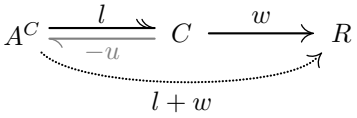
Table 1 The MMV system of constraint-propagation rules [10, 9], in a graphical representation. In each of the rules, the original edges are drawn solid and the generated edge is drawn dotted.

Rule	Graph representation	Applicability conditions
No Case	$P \xrightarrow{v} Q \xrightarrow{w} R$; $P \overset{v+w}{\dashrightarrow} R$	(none)
Upper Case	$P \xrightarrow{v} Q \xrightarrow{C:\, w} A^C$; $P \overset{C:\, v+w}{\dashrightarrow} A^C$	$P \neq C$
Lower Case	$A^C \xrightarrow{c:\, l} C \xrightarrow{w} R$; $A^C \overset{l+w}{\dashrightarrow} R$	$w \leq 0$, $R \neq C$
Cross Case	$A^C \xrightarrow{c:\, l} C \xrightarrow{D:\, w} A^D$; $A^C \overset{D:\, l+w}{\dashrightarrow} A^D$	$w \leq 0$, $C \neq D$
Label Removal	$P \xrightarrow{C:\, z} A^C$; $P \overset{z}{\dashrightarrow} A^C$	$z \geq -l^C$

Our contributions. In this paper, we present a new, sound-and-complete system of constraint-propagation rules, called RUL, which solves all the above problems. Our system contains only three rules (called Relax, Upper and Lower), with simple applicability conditions, which only generate new ordinary (i.e., unlabeled) constraints. This means that, with any application of the rules, the network remains an STNU in all respects, so no further explanation is necessary about the interpretation of generated constraints. We also provide a short, non-algorithmic proof of the completeness of our system, exhibiting a valid execution strategy explicitly. An illustration of the RUL system is given in Table 2 for comparison: the notation is explained in the next sections.

We perform a thorough analysis of our system of rules, proving both soundness and completeness in a simple but rigorous way. The proof of soundness is given with a minimal set of assumptions, which strictly extends its realm of applicability beyond the usual notion of DC in STNUs. In particular, our rules are sound even if the environment reveals the duration of a contingent link (A, l, u, C) right away at time $A + l$, i.e., at the lowest possible execution time of C, instead of its actual (contingent) execution time. This makes the soundness result strictly stronger, since any constraint deduced in this setting must apply also in the traditional setting, where the planner is only provided with less information. Moreover, for

Table 2 The RUL system of constraint propagation rules, in a graphical representation. In each of the rules, the original edges are drawn solid and the generated edge is drawn dotted.

Rule	Graph representation	Applicability conditions
RELAX		(none)
UPPER		(none)
LOWER		$w \leq 0$ for $R \in \mathcal{T}_{\mathrm{X}}$ $w \leq u^R$ for $R \in \mathcal{T}_{\mathrm{C}} \setminus \{C\}$

soundness, we only assume that each duration Δ can take its extremal values l and u; we do not need to take into account intermediate values $l < \Delta < u$. Again, this allows to apply our rules in a strictly more general setting, where the durations Δ are restricted to assume only specific values (if any) beyond the extremal ones. By considering relaxed assumptions, we address in one shot several variants of the notion of DC, and the traditional notion among them as a special case.

As for completeness, we consider the notion of late execution strategy, where each time-point is executed at the latest possible time among all valid dynamic execution strategies which finish within a fixed time horizon. The existence of a valid late execution strategy is non-trivial and, to the best of our knowledge, has not been observed in previous work. The proof of completeness of our rules is obtained by constructing explicitly the late execution strategy of the given STNU, which is used without modifications as a witness of dynamic controllability in all the settings considered for soundness. This not only proves the existence of the late execution strategy, but also proves the equivalence of several variants of the notion of DC for STNUs. Thus, the late executions strategy serves as a unifying theoretical tool for studying STNUs, besides its potential practical usage, being both easier to define and more generally applicable than the early execution strategy. Indeed, despite being possibly more useful in practice, the early strategy requires a more involved construction, and might vary among different but equivalent variants. Actually, there are several notions of "early execution strategies", varying according to the specific variant of controllability, while the late strategy has a unique, simple construction which applies to all the variants and thus proves their equivalence.

Finally, we propose an algorithm for executing the late strategy in real time. Thanks to the simple and explicit definition of late execution strategy, our algorithm is intuitive and amounts to instantiating the definition in a computationally efficient way. With our algorithm, we achieve a running time of $O(KN)$ for executing a network of N time-points and $1 \leq K < N$ contingent links. This is a considerable improvement over the previous best upper bound of $O(N^3)$ time for executing a network, achieved using the early strategy.

Paper outline. We introduce concepts and notations for STNUs in Section 2. Then, in Section 3, we describe our system of rules RUL. We prove soundness in Section 4 and completeness in Section 5. Finally, we show our real-time execution algorithm in Section 6.

2 Preliminaries and notation

An STNU is a tuple $\Gamma = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ where: $\mathcal{T}$ is a finite set of real-valued temporal variables called *time-points* (here denoted with capital letters $P, Q, R, \dots$), $\mathcal{C}$ is a finite set of *constraints* of the form $Q \leq P + w$, for $P, Q \in \mathcal{T}$ and $w \in \mathbb{R}$, and $\mathcal{L}$ is a finite set of *contingent links*, i.e., tuples of the form (A, l, u, C), for $A, C \in \mathcal{T}$ and $l, u \in \mathbb{R}$ with $0 < l \leq u < \infty$. In a contingent link (A, l, u, C), A is the *activation* time-point, l is the *duration lower bound*, u is the *duration upper bound*, and C is the *contingent* time-point. Distinct contingent links have distinct contingent time-points. Given the contingent time-point C of a contingent link $(A, l, u, C) \in \mathcal{L}$ we define $A^C = A$, $u^C = u$, $l^C = l$. The set of contingent time-points is $\mathcal{T}_{\mathrm{C}} \coloneqq \{C \mid (A, l, u, C) \in \mathcal{L}\}$, while the set of *executable* time-points is $\mathcal{T}_{\mathrm{X}} = \mathcal{T} \setminus \mathcal{T}_{\mathrm{C}}$. In the following we assume that an STNU $\Gamma = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ is given.

A *situation* is a function $s \colon \mathcal{T}_{\mathrm{C}} \to \mathbb{R}$, which assigns a *duration* $\Delta_s^C \coloneqq s(C) \in [l^C, u^C]$ to each contingent link $(A^C, l^C, u^C, C) \in \mathcal{L}$. The set of all possible situations is $\Omega_{\mathrm{all}} \coloneqq \prod_{C \in \mathcal{T}_{\mathrm{C}}} [l^C, u^C]$. For increased generality, the following definitions are given with respect to a fixed, non-empty subset of the situations $\Omega \subseteq \Omega_{\mathrm{all}}$. (The usual notions are obtained simply by choosing $\Omega = \Omega_{\mathrm{all}}$.) An *execution strategy* is a function $\sigma \colon (\Omega, \mathcal{T}_{\mathrm{X}}) \to \mathbb{R}$ that assigns an *execution time* $X_s^\sigma \coloneqq \sigma(s, X)$ to each executable time-point X, in each possible situation $s \in \Omega$. Moreover, we define the execution time of a contingent time-point $C \in \mathcal{T}_{\mathrm{C}}$ to be $C_s^\sigma \coloneqq A_s^{C\sigma} + \Delta_s^C$. In general, $P_s^\sigma \in \mathbb{R}$ denotes the execution time of the (executable or contingent) time-point $P \in \mathcal{T}$ in the situation s with the strategy σ. The superscript σ is omitted when clear from the context. Also, in some equations we replace the subscript s with $*$, meaning that the specific equation holds for every situation $s \in \Omega$.

An execution strategy σ is *viable* if $Q_* \leq P_* + w$ for every constraint $Q \leq P + w$ in $\mathcal{C}$. It is *dynamic* if, for any two situations $s, r \in \Omega$ and executable time-point $X \in \mathcal{T}_{\mathrm{X}}$, if $\{\langle C, \Delta_s^C \rangle \mid C_s < X_s\} = \{\langle C, \Delta_r^C \rangle \mid C_r < X_s\}$ then $X_r = X_s$. This definition correctly captures the intuitive notion that the execution time of X can only depend on the duration of contingent links whose contingent time-point is executed before X [4]. An STNU is said to be *dynamically controllable* (DC) if it admits an execution strategy which is both dynamic and viable.

We assume without loss of generality to have at most one constraint $Q \leq P + w_{PQ}$ in $\mathcal{C}$ for any two time-points $P, Q \in \mathcal{T}$: to this end, it is sufficient to take $w_{PQ} \coloneqq \min\{w \mid (Q \leq P + w) \in \mathcal{C}\}$. By convention $w_{PQ} = \infty$ if there are no constraints $(Q \leq P + w) \in \mathcal{C}$.

An STNU is represented as a graph on node set $\mathcal{T}$ with three classes of weighted edges: *ordinary edges* $\mathcal{E}^0 = \{(P, Q, w_{PQ}) \mid w_{PQ} < \infty\}$, *lower bound edges* $\mathcal{E}^- = \{(A^C, C, l^C) \mid C \in \mathcal{T}_{\mathrm{C}}\}$ and *upper bound edges* $\mathcal{E}^+ = \{(C, A^C, -u^C) \mid C \in \mathcal{T}_{\mathrm{C}}\}$. When convenient, this graph-based notation is used to describe the constraints and contingent links in an STNU.

We borrow an example of STNU from [5] to use as a running example for our rule system. The example is shown in Figure 1 with two different graphical notations. In the notation on the left, used by previous work, contingent links are represented with a pair of edges labeled with upper-case and lower-case labels. Since in this work we do not need to generate other labeled edges in any given network, we prefer to use a different notation, as shown on the right. Contingent links are still represented as a pair of parallel edges, one lower bound edge and one upper bound edge: this is convenient in describing the rules, ensuring that rules

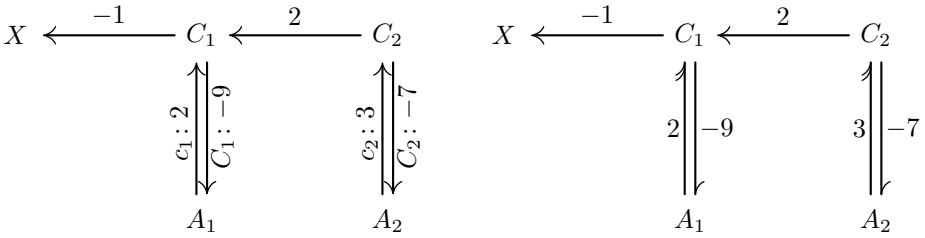


Figure 1 An STNU used as running example (borrowed from [5]). On the left, contingent links are represented as labeled constraints. On the right, they are represented using our notation.

always combine consecutive edges pointing in the right direction. However, the three types of edges, ordinary, lower and upper, are distinguished using a different graphical notation, and not by the addition of labels.

3 The RUL system

We introduce a system of three constraint-propagation rules: Relax, Upper and Lower. Each of these rules takes two consecutive edges (P, Q, v) and (Q, R, w) of the network and, if some conditions are satisfied, generates a new *ordinary* edge from P to R.

The Relax rule takes two ordinary edges (P, Q, v) and (Q, R, w) and generates the ordinary edge $(P, R, v + w)$.

The Upper rule takes an ordinary edge (P, C, v) and an upper bound edge $(C, A^C, -u^C)$ and generates the ordinary edge $(P, A^C, \max\{v - u^C, -l^C\})$.

The Lower rule takes a lower bound edge (A^C, C, l^C) and an ordinary edge (C, R, w) and generates the ordinary edge $(A^C, R, l^C + w)$, if the following preconditions are satisfied: either $R \in \mathcal{T}_{\mathrm{X}}$ and $w \leq 0$, or $R \in \mathcal{T}_{\mathrm{C}} \setminus \{C\}$ and $w \leq u^R$.

This system of constraint-propagation rules is called RUL and is illustrated in Table 2.

Comparison with MMV. The reader can compare the RUL system and the MMV system with the help of Table 1 and Table 2.

The Relax rule in RUL is identical to the No Case rule in MMV, and was renamed only for uniformity.

The Upper rule in RUL can be thought of as a combination of the Upper Case and Label Removal rules in MMV. Indeed, any edge $(P, A^C, \max\{v - u^C, -l^C\})$ obtained by Upper can be obtained with Upper Case and Label Removal in three steps.

1. If $v < u^C - l^C$, then take the unlabeled edge (P, C, v) and transform it into the weaker edge $(P, C, u^C - l^C)$. This is legal since the constraint $C \leq P + v$ subsumes the constraint $C \leq P + (u^C - l^C)$. In general, we end up with the edge $(P, C, \max\{v, u^C - l^C\})$.
2. Apply the Upper Case rule, obtaining the labeled edge $(P, A^C, C\colon\ \max\{v, u^C - l^C\} - u^C) = (P, A^C, C\colon\ \max\{v - u^C, -l^C\})$.
3. Apply the Label Removal rule, valid since $\max\{v - u^C, -l^C\} \geq -l^C$, to obtain the desired edge $(P, A^C, \max\{v - u^C, -l^C\})$.

As for the Lower rule, we have two cases. For $R \in \mathcal{T}_{\mathrm{X}}$, it is identical to the Lower Case rule. However, for $R \in \mathcal{T}_{\mathrm{C}}$, it extends strictly the domain of applicability of Lower Case. This is a crucial addition given by the RUL system with respect to MMV, since the edges generated by Lower cannot in general be obtained using MMV only. It is thanks to this addition that we can avoid generating labeled edges with the Upper Case rule, and still achieve completeness.

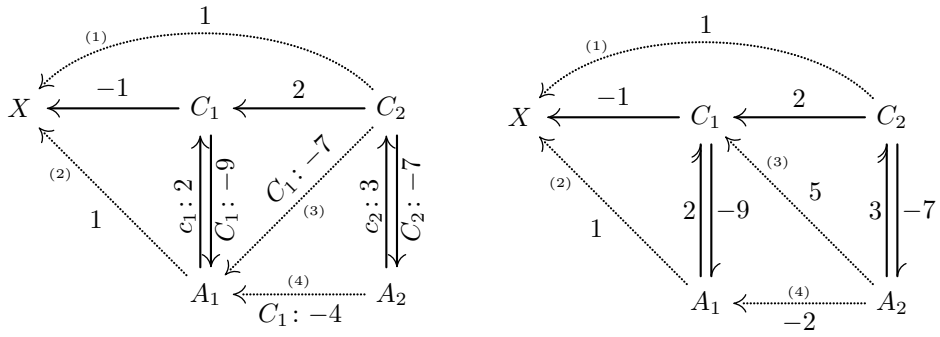


Figure 2 Examples of application of MMV (left) and RUL (right) on the example STNU.

Application examples. In Figure 2, we compare the application of the MMV and RUL systems on the example network.

MMV:

1. The No Case rule is applied to $(C_2, C_1, 2)$ and $(C_1, X, -1)$ to obtain $(C_2, X, 1)$.
2. The Lower Case rule is applied to $(A_1, C_1, c_1 : 2)$ and $(C_1, X, -1)$ to obtain $(A_1, X, 1)$.
3. The Upper Case rule is applied to $(C_2, C_1, 2)$ and $(C_1, A_1, C_1 : -9)$ to obtain the labeled edge $(C_2, A_1, C_1 : -7)$.
4. The Cross Case rule is applied to $(A_2, C_2, c_1 : 3)$ and $(C_2, A_1, C_1 : -7)$ to obtain $(A_2, A_1, C_1 : -4)$.

RUL:

1. The Relax rule is applied to $(C_2, C_1, 2)$ and $(C_1, X, -1)$ to obtain $(C_2, X, 1)$.
2. The Lower rule (in the case $R \in \mathcal{T}_{\mathrm{X}}$) is applied to $(A_1, C_1, 2)$ and $(C_1, X, -1)$ to obtain $(A_1, X, 1)$.
3. The Lower rule (in the case $R \in \mathcal{T}_{\mathrm{C}}$) is applied to $(A_2, C_2, 3)$ and $(C_2, C_1, 2)$ to obtain $(A_2, C_1, 5)$.
4. The Upper rule is applied to $(A_2, C_1, 5)$ and $(C_1, A_1, -9)$ to obtain $(A_2, A_1, \max\{5 - 9, -2\}) = (A_2, A_1, -2)$.

The edges $(C_2, X, 1)$ and $(A_1, X, 1)$ are generated by both systems, respectively using the No Case and Relax rules, and the Lower Case and Lower rules with $R \in \mathcal{T}_{\mathrm{X}}$.

As for the other edges, on one hand, the MMV system generates the labeled edges $(C_2, A_1, C_1 : -7)$ and $(A_2, A_1, C_1 : -4)$ which are not generated using RUL. On the other hand, the RUL system generates the edge $(A_2, C_1, 5)$, with the Lower rule in the case $R \in \mathcal{T}_{\mathrm{C}}$, which cannot be generated with MMV, and only gets a value -2 on the edge from A_2 to A_1, the strongest bound that can be given unconditionally, instead of the labeled edge $(A_2, A_1, C_1 : -4)$. Observe, however, that the edge $(A_2, A_1, C_1 : -4)$ could be obtained, *in a single step*, from the edges $(A_2, C_1, 5)$ and $(C_1, A_1, C_1 : -9)$, with an application of the Upper Case rule, so it is somewhat implicit in the edge $(A_2, C_1, 5)$. In general, when replacing the MMV system the RUL system, the labeled edges, with head in an activation time-point, are replaced with unlabeled edges with head in the corresponding contingent time-point.

4 Soundness

We commit to prove soundness of all three rules comprising RUL, in the most general sense. Specifically, we partially relax the following assumptions: (1) that $\Omega = \Omega_{\mathrm{all}}$, i.e., that every possible situation $s \in \Omega_{\mathrm{all}}$ can be realized by the environment, and (2) that the duration Δ_s^C

is revealed only at time C_s. Proving soundness under our weaker assumptions is a stronger result, which not only implies soundness in the usual sense, but also helps applying the model of STNUs to a wider range of scenarios.

To express our assumptions precisely, the following notion is needed.

▶ **Definition 1.** Given a situation s, a contingent time-point $C_0 \in \mathcal{T}_{\mathrm{C}}$ and a value $d \in [l^{C_0}, u^{C_0}]$, define the situation $s[d/\Delta^{C_0}] = r$ where $\Delta_r^{C_0} = d$ and $\Delta_r^C = \Delta_s^C$ for every $C \in \mathcal{T}_{\mathrm{C}} \setminus \{C_0\}$.

We first state a condition on the set Ω.

▶ **Definition 2.** The set $\Omega \subseteq \Omega_{\mathrm{all}}$ is *extremal-closed* if, for every $s \in \Omega$, also $s[u^C/\Delta^C] \in \Omega$ and $s[l^C/\Delta^C] \in \Omega$.

For soundness, we need to assume that Ω is extremal-closed. This means that, even if not all situations may be realized by the environment, each duration Δ^C can at least assume the extremal values l^C and u^C, and do so independently of the other durations. Observe that this is the case for $\Omega = \Omega_{\mathrm{all}}$, where all the situations are possible. However, there are other, interesting examples of sets Ω that are extremal-closed: e.g., $\Omega = \prod_{C \in \mathcal{T}_{\mathrm{C}}} \{l^C, u^C\}$, where each duration Δ^C can *only* assume the extremal values l^C and u^C. Hence, with this assumption, we strictly increase the generality with respect to the case $\Omega = \Omega_{\mathrm{all}}$ addressed by previous work.

A relaxed notion of dynamic execution strategy is also introduced, where the duration of a contingent link $(A, l, u, C) \in \mathcal{L}$ in the situation s is revealed at time $A_s + l$ instead of C_s.

▶ **Definition 3.** An execution strategy σ is *upfront-dynamic* if, for any two situations $s, r \in \Omega$ and executable time-point $X \in \mathcal{T}_{\mathrm{X}}$, we have that $X_r = X_s$ if $\{\langle C, \Delta_s^C\rangle \mid A_s^C + l^C < X_s\} = \{\langle C, \Delta_r^C\rangle \mid A_r^C + l^C < X_s\}$.

By Lemma 4, working with upfront-dynamic strategies yields no loss in generality with respect to dynamic execution strategies.

▶ **Lemma 4.** *Any dynamic execution strategy is upfront-dynamic.*

Proof. The statement is clear if we consider that $C_* \geq A_*^C + l^C$, so upfront-dynamic strategies can only have more information at any time, whence more freedom, with respect to dynamic strategy.

A formal proof is provided for reference. The proof is by contradiction: assume σ is dynamic but not upfront-dynamic. Fix any two scenarios $s, r \in \Omega$ and suppose $X_s \neq X_r$ for some $X \in \mathcal{T}_{\mathrm{X}}$. Assume without loss of generality that $X_s = t < X_r$ and that t is minimal, i.e., $Y_s = Y_r$ for any $Y \in \mathcal{T}_{\mathrm{X}}$ such that either $Y_s < t$ or $Y_r < t$.

Since σ is dynamic, we have $\{\langle C, \Delta_s^C\rangle \mid C_s < t\} \neq \{\langle C, \Delta_r^C\rangle \mid C_r < t\}$. Consider any C in the difference. Either $A_s^C < t$ or $A_r^C < t$, then actually[1] $A_s^C = A_r^C$, by minimality of t. Thus, $\Delta_s^C \neq \Delta_r^C$, so $\{\langle C, \Delta_s^C\rangle \mid A_s^C + l^C < X_s\} \neq \{\langle C, \Delta_r^C\rangle \mid A_r^C + l^C < X_s\}$, closing the proof. ◀

Lemma 5 is the last ingredient needed for our soundness proof.

▶ **Lemma 5.** *Consider an upfront-dynamic execution strategy σ and situations $s, r \in \Omega$, where $r = s[d/\Delta^{C_0}]$ for some contingent time-point $C^0 \in \mathcal{T}_{\mathrm{C}}$ and duration $d \in [l^{C^0}, u^{C^0}]$. Then, $P_r = P_s$ for any $P \in \mathcal{T} \setminus \{C_0\}$ such that $P_s \leq A_s^{C_0} + l^{C_0}$. In particular, $A_s^{C_0} = A_r^{C_0}$.*

[1] For simplicity, here we are assuming $A^C \in \mathcal{T}_{\mathrm{X}}$, as common in the context of STNUs. The proof can be easily adapted if this assumption does not hold.

Proof. By contradiction. Fixed s and r, choose P so that $P_s \leq A_s^{C_0} + l^{C_0}$, $P_s \neq P_r$, and P_s is smallest possible. Then $A_s^C = A_r^C$ for every $C \in \mathcal{T}_{\mathrm{C}}$ such that $A_s^C < P_s$. Also, $\Delta_r^C = \Delta_s^C$ for every $C \in \mathcal{T}_{\mathrm{C}} \setminus \{C_0\}$. If $P \in \mathcal{T}_{\mathrm{C}} \setminus \{C_0\}$, then $P_s = A_s^P + \Delta_s^P = A_r^P + \Delta_r^P = P_r$. Otherwise, if $P \in \mathcal{T}_{\mathrm{X}}$, then $P_s = P_r$ by Definition 3, since $A_s^{C_0} + l^{C_0} \geq P_s$. ◀

Soundness is proven in Lemma 6, where each of the three rules comprising RUL is proven to be sound, even for upfront-dynamic execution strategies, and for any extremal-closed $\Omega \subseteq \Omega_{\mathrm{all}}$.

▶ **Lemma 6.** *Assume Ω is extremal-closed. Let σ be a viable upfront-dynamic execution strategy. Consider an edge (P, R, x) generated using RUL. Then, $R_*^\sigma \leq P_*^\sigma + x$.*

Proof. Relax. Clearly, $Q_* \leq P_* + v$ and $R_* \leq Q_* + w$ imply $R_* \leq P_* + (v + w)$.

Upper. Assume $C_* \leq P_* + v$. Fix any situation $s \in \Omega$: we need to prove $A_s^C \leq P_s^C + \max\{v - u^C, -l^C\}$. If $A_s^C < P_s - l^C$ we are done, hence assume $A_s^C \geq P_s - l^C$. Take $r = s[u^C/\Delta^C]$ and observe that $A_r^C = A_s^C$ by Lemma 5. Moreover, $P_s \leq A_s^C + l^C$, so also $P_r = P_s$ by Lemma 5. Then,

$$\begin{aligned} A_s^C &= A_r^C && \text{by Lemma 5} \\ &= C_r - u^C && \text{as } C_r = A_r^C + \Delta_r^C = A_r^C + u^C \\ &\leq P_r + v - u^C && \text{by assumption } C_* \leq P_* + v \\ &= P_s + v - u^C && \text{as } P_s = P_r \text{ by Lemma 5.} \end{aligned}$$

Lower. Assume $R_* \leq C_* + w$. Fix any situation $s \in \Omega$: we need to prove that $R_s \leq A_s^C + (l^C + w)$. Take $r = s[l^C/\Delta^C]$ and observe that $A_r^C = A_s^C$ by Lemma 5. We now prove that $R_r = R_s$, in two cases.

Case 1: $R \in \mathcal{T}_{\mathrm{X}}$ and $w \leq 0$.

Consider that $R_r \leq C_r + w = A_r^C + l^C + w \leq A_r^C + l^C$ since $w \leq 0$. Hence, $R_s = R_r$ by Lemma 5.

Case 2: $R \in \mathcal{T}_{\mathrm{C}} \setminus \{C\}$ and $w \leq u^R$.

Take $q = r[u^R/\Delta^R]$ and observe that $A_q^R = R_q - u^R \leq C_q + w - u^R \leq C_q = A_q^C + l^C$, whence[2] $A_s^R = A_r^R = A_q^R$ by Lemma 5 and $R_s = A_s^R + \Delta_s^R = A_r^R + \Delta_r^R = R_r$.

Then,

$$\begin{aligned} R_s &= R_r && \text{proven separately in the two cases} \\ &\leq C_r + w && \text{by assumption } R_* \leq C_* + w \\ &= A_r^C + l^C + w && \text{as } C_r = A_r^C + \Delta_r^C = A_r^C + l^C \\ &= A_s^C + l^C + w && \text{as } A_s = A_r \text{ by Lemma 5.} \end{aligned}$$

◀

5 Completeness via late execution strategy

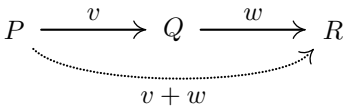
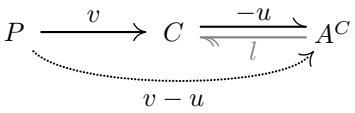
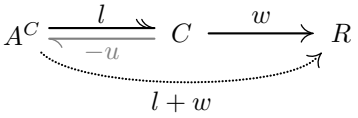
For completeness, we need to assume, as usual in temporal networks, to have a special time-point Z which is due to be executed at time 0 and before any other time-point. Moreover, it will be useful to consider Z as a contingent node for uniformity.

[2] Let us spend some extra words to explain, in a more intuitive way, this step of the proof, which is crucial for our system of rules and arguably is the most involved. In the situation q, where $\Delta_q^C = l^C$ and $\Delta_q^R = u^R$, the activation node A^R for R must be executed before or at time $A_q^C + l_q^C$: this is a direct consequence of the existing constraint $R \leq C + w$ with $w \leq u^R$. Hence, at the time A_q^R, neither the duration Δ^R (trivially) nor Δ^C are known to the planner. Thus, the time A^R cannot change depending on those durations, i.e., $A_s^R = A_r^R = A_q^R$ as claimed.

Table 3 Extra restrictions for the rules RELAX$^+$ and UPPER$^+$.

Rule	Extra restrictions		
RELAX$^+$	$P \in \mathcal{T}_{\mathrm{C}}$	$Q \in \mathcal{T}_{\mathrm{X}}$	$v > 0$
UPPER$^+$	$P \in \mathcal{T}_{\mathrm{C}}$		$v > u^C$

Table 4 RUL$^+$ sub-system of RUL.

Rule	Graph representation	Applicability conditions
RELAX$^+$		$P \in \mathcal{T}_{\mathrm{C}}$ $Q \in \mathcal{T}_{\mathrm{X}}$ $v > 0$
UPPER$^+$		$P \in \mathcal{T}_{\mathrm{C}}$ $v > u^C$
LOWER		$w \leq 0$ for $R \in \mathcal{T}_{\mathrm{X}}$ $w \leq u^R$ for $R \in \mathcal{T}_{\mathrm{C}} \setminus \{C\}$

▶ **Definition 7.** An STNU is *pointed* if it contains distinguished time-points $A^Z \in \mathcal{T}_{\mathrm{X}}$, $Z \in \mathcal{T}_{\mathrm{C}}$, and a contingent link $(A^Z, 1, 1, Z) \in \mathcal{L}$, with $w_{PZ} < 0$ and $w_{ZP} > 0$ for any $P \in \mathcal{T}_{\mathrm{X}} \setminus \{A^Z\}$. If also $w_{ZP} < \infty$ for every $P \in \mathcal{T}_{\mathrm{X}} \setminus \{A^Z\}$, then it is *upper-bounded*.

Any network can be made pointed, without loss of generality, by adding the contingent link $(A^Z, 1, 1, Z) \in \mathcal{L}$ between dummy nodes A^Z and Z, and setting $w_{PZ} = -1$ and $w_{ZP} = \infty$ for every $P \in \mathcal{T}_{\mathrm{X}} \setminus \{A^Z\}$. To make it upper-bounded, it is sufficient to pick a sufficiently large horizon time $h > 0$, and set $w_{ZP} = h$ for every $P \in \mathcal{T}_{\mathrm{X}} \setminus \{A^Z\}$.

The completeness of the RUL system is proven by exhibiting an explicit execution strategy for any upper-bounded network Γ which is closed under them. Actually, Γ only needs to be closed under a subset of the RUL system, called RUL$^+$. In the sub-system RUL$^+$, we only apply the RELAX and UPPER rules when $P \in \mathcal{T}_{\mathrm{C}}$ is contingent. Moreover, we require $v > 0$ and $Q \in \mathcal{T}_{\mathrm{X}}$ for the RELAX rule, and $v > u^C$ for the UPPER rule. The restricted rules, called RELAX$^+$ and UPPER$^+$, are illustrated in Table 3. Together with the unrestricted rule LOWER, they constitute the RUL$^+$ system (Table 4).

The strategy we exhibit is a *late* execution strategy, as opposed to the commonly used early execution strategy.

▶ **Definition 8.** In an upper-bounded network Γ, a viable dynamic execution strategy σ is the *late execution strategy* if, for any other viable dynamic execution strategy τ, $P_s^\sigma \geq P_s^\tau$ for each $s \in \Omega$ and $P \in \mathcal{T}$. (It is implicitly assumed $A_*^Z = -1$ for both σ and τ.) I.e., in the late execution strategy each time-point is executed at the latest possible time over all viable dynamic execution strategies.

It follows from the definition that the late execution strategy, if it exists, is unique. However, it is not at all implied by the definition that a given dynamically controllable network, even if upper-bounded, admits a late execution strategy. Indeed, in principle it could be that the strategy defined as

$$X_s^\sigma := \max\{X_s^\tau \mid \text{dynamic and viable } \tau \text{ with } A_*^Z = -1\},$$

i.e., the only candidate to be the late execution strategy, is not itself dynamic and viable.[3] It is obtained, as a consequence of the analysis of our system of rules, that indeed, for the case of STNUs, the late execution strategy always exists, provided the network is DC and upper-bounded. Working with late execution strategies – it turns out – is very convenient, since we do not have to deal with "wait" (or labeled) edges; this will allow for a short and self-contained non-algorithmic proof of the completeness of RUL.

Notice that a strategy satisfying the following equation would trivially be the late execution strategy, if only it were dynamic and viable:

$$Y_* = \min_{\substack{C \in \mathcal{T}_\mathrm{C} \\ w_{CY} > 0}} C_* + w_{CY} \tag{1}$$

for every $Y \in \mathcal{T}_\mathrm{X} \setminus \{A^Z\}$, and $A_*^Z = -1$.

▶ **Lemma 9.** *Assume Γ is upper-bounded. Then, there exists a unique execution strategy σ satisfying Equation 1. Moreover, σ is dynamic.*

Proof. The reason why the lemma holds is simple: Equation 1 defines the execution time of each executable time-point in terms of time-points with a strictly lower execution time, hence σ is well-defined and unique. Remarkably, the very same observation also shows that σ is dynamic. Notice that boundedness (i.e., $0 < w_{ZP} < \infty$ for every $P \in \mathcal{T}_\mathrm{X} \setminus \{A^Z\}$) is necessary to ensure that the right-hand side of Equation 1 is finite. Also, the assumption $A_*^Z = -1$ is necessary for σ to be unique.

To insist on a formal proof, we can rewrite Equation 1 as follows:

$$Y_s = t \iff \min_{\substack{C \in \mathcal{T}_\mathrm{C} \\ w_{CY} > 0 \\ C_s < t}} C_s + w_{CY} = t \qquad \forall\, Y \in \mathcal{T}_\mathrm{X}, s \in \Omega.$$

In this writing, the dependency on the time $t \in [0, \infty)$ is made explicit and allows for an inductive construction of the solution σ for increasing t, which is thus proven to be unique. To prove that σ is also dynamic, suppose $X_s < X_r$ for some $X \in \mathcal{T}_\mathrm{X}$ and $s, r \in \Omega$. Also, choose a minimal X_s, i.e., assume $Y_s = t \iff Y_r = t$ for every $t < X_s$ and $Y \in \mathcal{T}_\mathrm{X}$. Take $C \in \mathcal{T}_\mathrm{C}$ such that $X_s = C_s + w_{CX}$ and $w_{CX} > 0$ (i.e., the arg min of Equation 1). By definition of σ, we have $X_r \le C_r + w_{CX}$, so $C_s + w_{CX} = X_s < X_r \le C_r + w_{CX}$, whence $C_s < C_r$. By minimality of X_s we have[4] $A_s^C = A_r^C$, so it must be $\Delta_s^C < \Delta_r^C$. This proves that $\{\langle C, \Delta_s^C\rangle \mid C_s < t\} \neq \{\langle C, \Delta_r^C\rangle \mid C_r < t\}$ for every $t \ge X_s$, concluding the proof. ◀

▶ **Lemma 10.** *Assume Γ is upper-bounded and closed under RUL$^+$. Then, the strategy σ defined by Equation 1 is viable.*

[3] In some other temporal network models, such as CSTNs, it is possible to construct examples of upper-bounded networks which admit an early execution strategy but no late execution strategy.

[4] For simplicity, here we are assuming $A^C \in \mathcal{T}_\mathrm{X}$, as common in the context of STNUs. The proof can be easily adapted if this assumption does not hold.

Table 5 Cases in the proof of Lemma 10.

Constraint $R \leq P + w_{PR}$		Case	
$P \in \mathcal{T}_{\mathrm{X}}$			Relax$^+$
$P \in \mathcal{T}_{\mathrm{C}}$	$R \in \mathcal{T}_{\mathrm{X}}$	$w_{PR} > 0$	By-Construction
		$w_{PR} \leq 0$	Lower
	$R \in \mathcal{T}_{\mathrm{C}}$	$w_{PR} > u_R$	Upper$^+$
		$w_{PR} \leq u_R$	Lower

Proof. Fix a situation s. We prove that $R_s \leq P_s + w_{PR}$ by induction on the times P_s and R_s.

The following four cases (By-Construction, Relax$^+$, Lower, Upper$^+$) cover all the possibilities, as illustrated in Table 5.

Case By-Construction: $Y \leq C + w_{CY}$ with $C \in \mathcal{T}_{\mathrm{C}}$, $w_{CY} > 0$ and $Y \in \mathcal{T}_{\mathrm{X}}$.
We have $Y_s \leq C_s + w_{CY}$ by Equation 1.

Case Relax$^+$: $R \leq X + w_{XR}$ with $X \in \mathcal{T}_{\mathrm{X}}$.
Take $C \in \mathcal{T}_{\mathrm{C}}$ such that $X_s = C_s + w_{CX}$ with $w_{CX} > 0$. Then,

$$\begin{aligned} R_s &\leq C_s + w_{CR} && \text{by induction, since } C_s < X_s = C_s + w_{CX} \\ &\leq C_s + w_{CX} + w_{XR} && \text{by Relax}^+ \text{ applied to } C, X, R \\ &= X_s + w_{XR} && \text{by assumption } X_s = C_s + w_{CX}. \end{aligned}$$

Case Lower: $R \leq C + w_{CR}$ with $C \in \mathcal{T}_{\mathrm{C}}$, and either $R \in \mathcal{T}_{\mathrm{X}}$ and $w_{CR} \leq 0$, or $R \in \mathcal{T}_{\mathrm{C}} \setminus \{C\}$ and $w_{CR} \leq u_R$.
Then,

$$\begin{aligned} R_s &\leq A^C_s + w_{A^C R} && \text{by induction, as } A^C_s < C_s = A^C_s + \Delta^C_s \\ &\leq A^C_s + (w_{CR} + l^C) && \text{by Lower applied to } C, A^C, R \\ &\leq C_s + w_{CR} && \text{since } C_s = A^C_s + \Delta^C_s \geq A^C + l^C. \end{aligned}$$

Case Upper$^+$: $C \leq P + w_{PC}$, with $C \in \mathcal{T}_{\mathrm{C}}$ and $w_{PC} \geq u^C$.
Then,

$$\begin{aligned} C_s &\leq A^C_s + u^C && \text{as } C_s = A^C_s + \Delta^C_s \text{ and } \Delta^C_s \leq u^C \\ &\leq P_s + w_{PA^C} + u^C && \text{by induction, as } A^C_s < C_s = A^C_s + \Delta^C_s \\ &\leq P_s + (w_{PC} - u^C) + u^C && \text{by Upper}^+ \text{ applied to } P, A^C, C \\ &= P_s + w_{PC}. \end{aligned}$$

◀

To complete our analysis, we need to show that a DC network admits a closure under RUL$^+$. More precisely:

▶ **Lemma 11.** *If a network admits a viable upfront-dynamic execution strategy, then it admits a closure under RUL.*

This fact is quite intuitive: just continue to apply the rules, as long as possible, until closure is reached. Since the rules are sound, and the network admits a viable and dynamic execution strategy σ, then at some point it should become impossible to deduce stricter and stricter constraints, as they must be satisfied at least by σ. However, to prove that closure is

reached in a finite number of steps requires (explicitly or implicitly) the description of an actual algorithm to perform the propagations, which is beyond the scope of this work. The authors verified that the existing DC checking algorithms can be adapted to use the RUL system, and indeed produce the closure under RUL of a given input network, provided it is DC.

We explore here the consequences of Lemma 11.

▶ **Theorem 12.** *Let Γ be an upper-bounded STNU. The following are equivalent:*
1. *Γ is dynamically controllable (admits a viable dynamic execution strategy),*
2. *Γ admits a viable upfront-dynamic execution strategy,*
3. *Γ admits a closure under the RUL system (without negative self-loops),*
4. *Γ admits a closure under the RUL$^+$ system (with only positive edges from Z).*

Proof. Consider the following implications:
- 1 $\implies$ 2, by Lemma 5,
- 2 $\implies$ 3, by Lemma 11,
- 3 $\implies$ 4, trivial,
- 4 $\implies$ 1, by the construction of the late execution strategy (Lemma 9 and Lemma 10).

◀

▶ **Corollary 13.** *Every dynamically controllable upper-bounded STNU admits a late execution strategy.*

▶ **Corollary 14.** *The property of being dynamically controllable does not change if the duration Δ_s^C of contingent links is revealed at any time between $A_s^C + l^C$ and C_s, nor if the duration Δ_s^C is restricted to assume only the extremal values l^C and u^C.*

Proof. Consider that the property of admitting a closure under the RUL system is unvaried, and the RUL system is sound-and-complete for all these variants. ◀

▶ **Corollary 15.** *DC-checking for an STNU with N time-points, M constraints, and K contingent links admits a certificate of YES of size $O(KN)$, verifiable in $O(K^2N + KM)$ time and logarithmic space.*

Proof. The closure of the network under RUL$^+$ is a certificate of YES. Only $O(KN)$ new edges may be generated, since they all have tail in either a contingent time-point or an activation time-point, and this suffices for the size bound. To verify the closure under the Relax rule, one can guess a constraint $R \leq Q + w$ (either original or in the closure) and the time-point $P \in \mathcal{T}_C$, and verify that the rule does not generate any stronger constraint. This takes $O(K^2N + KM)$ time (a factor $KN + M$ to guess the original or generated constraint, times a factor K to guess the time-point P). Similarly, to verify the closure under the Lower rule, one can guess $C \in \mathcal{T}_C$ and $R \in \mathcal{T}$, in $O(KN)$ total time, while verifying the Upper rule only require $O(K^2) \leq O(KN)$ time to guess $P, C \in \mathcal{T}_C$. Finally, the space required for verification is clearly logarithmic. ◀

▶ **Corollary 16.** *If the network is integer, i.e., $w_{PQ} \in \mathbb{Z} \cup \{\infty\}$ for every $P, Q \in \mathcal{T}$, there there exists a strategy where each execution time X_s satisfies either:*
- *$X_s \in \mathbb{Z}$ is integer, or*
- *$X_s = C_s + k$ for some $C \in \mathcal{T}_C$ with integer $k \in \mathbb{Z}$.*

▶ **Corollary 17.** *If the network is integer, i.e., $w_{PQ} \in \mathbb{Z} \cup \{\infty\}$ for every $P, Q \in \mathcal{T}$, and the durations are constrained to be integers, i.e., $l^C, u^C \in \mathbb{Z}$ and $\Omega = \prod_{C \in \mathcal{T}_C} \{l^C, l^C + 1, \ldots, u^C\}$, then the network admits an integer execution strategy, i.e., where $P_* \in \mathbb{Z}$ for every $P \in \mathcal{T}$.*

Proof. Consider the late execution strategy. ◀

6 Real-time execution

We have shown that a network closed under RUL$^+$ admits a late execution strategy, defined by Equation 1. We next show that the late execution strategy can be also executed in real-time, with a total running time $O(KN)$. We only need to assume that, for every $C \in \mathcal{T}_{\mathrm{C}}$, we have stored the values w_{CX}, for every $X \in \mathcal{T}_{\mathrm{X}}$ such that $w_{CX} > 0$, in a list L_C, sorted by increasing numerical value of w_{CX}. These lists L_C, for $C \in \mathcal{T}_{\mathrm{C}}$, can be compiled in a preprocessing step (by sorting and filtering all the values w_{CX}, $X \in \mathcal{T}_{\mathrm{X}}$), in $O(KN \log N)$ total time. This cost should not be accounted for in the real-time execution running time: it is better regarded as a light postprocessing of the DC-checking step, having lower asymptotic time and space complexity.

The algorithm works by maintaining the value

$$X_\circ := \min_{\substack{C \in \mathcal{X}_{\mathrm{C}} \\ w_{CX} > 0}} C_\bullet + w_{CX}$$

for every $X \in \mathcal{T}_X$, where $\mathcal{X}_{\mathrm{C}}$ is the set of already executed contingent time-points (initially containing only Z), $C_\bullet$ is the execution time of the time-point C, and $X_\circ$ represents the current scheduled time for the time-point X. During the real-time execution of the network, the values $X_\circ$ can only decrease monotonically, and have to be updated whenever a contingent time-point gets executed. The values $X_\circ$ are maintained in a list S sorted by increasing numerical value of $X_\circ$, which can be regarded as an event queue representing the current candidate schedule of executable time-points.

At the beginning $\mathcal{X}_{\mathrm{C}} = \{Z\}$, and the list S is built simply by copying L_Z, setting $X_\circ = w_{XZ}$ for every $X \in \mathcal{T}_{\mathrm{X}}$. The algorithm works by repeatedly executing the time-point $\bar{X}$ which appears first in the list S, at time $\bar{X}_\circ$, unless a contingent time-point is executed before that time. When a contingent time-point $\bar{C}$ is executed, the values $X_\circ$ and the list S need to be updated before resuming the execution. This can be done in $O(N)$ time by merging the new candidate values $\bar{C}_\bullet + w_{\bar{C}X}$ with the list S. More precisely, a new list $S_{\bar{C}}$ is built from $L_{\bar{C}}$, containing the values

$$X_\circ^{\bar{C}} := \bar{C}_\bullet + w_{\bar{C}X}$$

for every $X \in \mathcal{T}_{\mathrm{X}}$ such that $w_{\bar{C}X} > 0$. The list $S_{\bar{C}}$ is constructed preserving the increasing order given by $L_{\bar{C}}$. Observe that the new value of $X_\circ$ is obtained as

$$X_\circ^{\mathrm{new}} := \min\{X_\circ^{\mathrm{old}}, X_\circ^{\bar{C}}\}.$$

To obtain a sorted list with the new values, first merge the lists S and $S_{\bar{C}}$ as in the Merge Sort algorithm. Then, scan the merged list to remove duplicates. For each duplicated time-point X, keep only the first occurrence in the list, i.e., the one with smaller numerical value $\min\{X_\circ^{\mathrm{old}}, X_\circ^{\bar{C}}\}$, and set that value as the new value of $X_\circ$. With this pass we obtain the updated list S^{new}, already sorted, containing the updated values $X_\circ^{\mathrm{new}}$ which take into account the execution of C. Since we pay $O(N)$ time for each of the K contingent time-points and only $O(1)$ for each executable time-point, the total time is $O(KN)$.

7 Conclusions

We have presented a new, sound-and-complete system of constraint propagation rules, called RUL, for checking the dynamic controllability of STNUs. Soundness has been proven, for

each of the rules, in its most general version, and holds even if we take into account upfront-dynamic execution strategies besides dynamic execution strategies. As for completeness, we have proven that closure under a subset of the rules, called RUL^+, is sufficient to guarantee the existence of a viable dynamic execution strategy. Specifically, we considered the strategy which executes each time-point at the latest time allowed by positive edges from contingent nodes. We defined this strategy explicitly, with a single equation; then, we provided a short proof that it is viable, i.e., it satisfies every other constraint in the network, assuming closure under RUL^+. This strategy is the late execution strategy, since moving the execution time of any time-point further in the future would violate at least one constraint. Finally, we showed how to execute the late execution strategy in real-time, paying only $O(KN)$ time in total for a network with N time-points and K contingent links. With the introduction of the RUL system, this paper helps making STNUs not only simpler, and better understood, but also more generally applicable.

In this work, we did not provide any new DC-checking algorithm. However, existing algorithms can be adapted to use the RUL system, and produce the closure under RUL of a given input network, if it is DC, without incurring in any additional complexity cost. Furthermore, the added simplicity of the new rules allows for a cleaner approach to DC-checking, which yields improved algorithms as for both simplicity and efficiency, beyond the scope in this work. We opted for a short and clean description of the RUL system, which is proposed as a new foundation for the theory of DC-checking and real-time execution of STNUs, and their numerous extensions.

References

1. Carlo Combi, Luke Hunsberger, and Roberto Posenato. An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty. In Joaquim Filipe and Ana L. N. Fred, editors, *ICAART 2013 – Proceedings of the 5th International Conference on Agents and Artificial Intelligence, Volume 2, Barcelona, Spain, 15-18 February, 2013*, pages 144–156. SciTePress, 2013.
2. Patrick R. Conrad and Brian Charles Williams. Drake: An efficient executive for temporal plans with choice. *CoRR*, abs/1401.4606, 2014. URL: `http://arxiv.org/abs/1401.4606`.
3. Robert T. Effinger, Brian Charles Williams, Gerard Kelly, and Michael Sheehy. Dynamic controllability of temporally-flexible reactive programs. In Alfonso Gerevini, Adele E. Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*. AAAI, 2009. URL: `http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/739`.
4. Luke Hunsberger. Fixing the semantics for dynamic controllability and providing a more practical characterization of dynamic execution strategies. In Carsten Lutz and Jean-François Raskin, editors, *TIME 2009, 16th International Symposium on Temporal Representation and Reasoning, Bressanone-Brixen, Italy, 23-25 July 2009, Proceedings*, pages 155–162. IEEE Computer Society, 2009. `doi:10.1109/TIME.2009.25`.
5. Luke Hunsberger. Efficient execution of dynamically controllable simple temporal networks with uncertainty. *Acta Inf.*, 53(2):89–147, 2016. `doi:10.1007/s00236-015-0227-0`.
6. Lina Khatib, Paul H. Morris, Robert A. Morris, and Francesca Rossi. Temporal constraint reasoning with preferences. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 322–327. Morgan Kaufmann, 2001.
7. Paul Morris. A structural characterization of temporal dynamic controllability. In Frédéric Benhamou, editor, *Principles and Practice of Constraint Programming - CP 2006, 12th*

International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings, volume 4204 of *Lecture Notes in Computer Science*, pages 375–389. Springer, 2006. doi:10.1007/11889205_28.

8 Paul Morris. Dynamic controllability and dispatchability relationships. In Helmut Simonis, editor, *Integration of AI and OR Techniques in Constraint Programming – 11th International Conference, CPAIOR 2014, Cork, Ireland, May 19-23, 2014. Proceedings*, volume 8451 of *Lecture Notes in Computer Science*, pages 464–479. Springer, 2014. doi:10.1007/978-3-319-07046-9_33.

9 Paul H. Morris and Nicola Muscettola. Temporal dynamic controllability revisited. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 1193–1198. AAAI Press / The MIT Press, 2005. URL: http://www.aaai.org/Library/AAAI/2005/aaai05-189.php.

10 Paul H. Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal uncertainty. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 494–502. Morgan Kaufmann, 2001.

11 Mikael Nilsson, Jonas Kvarnström, and Patrick Doherty. Incremental dynamic controllability revisited. In Daniel Borrajo, Subbarao Kambhampati, Angelo Oddi, and Simone Fratini, editors, *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, ICAPS 2013, Rome, Italy, June 10-14, 2013*. AAAI, 2013. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS13/paper/view/6028.

12 Mikael Nilsson, Jonas Kvarnström, and Patrick Doherty. EfficientIDC: A faster incremental dynamic controllability algorithm. In Steve A. Chien, Minh Binh Do, Alan Fern, and Wheeler Ruml, editors, *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*. AAAI, 2014. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS14/paper/view/7936.

13 Mikael Nilsson, Jonas Kvarnström, and Patrick Doherty. Incremental dynamic controllability in cubic worst-case time. In Amedeo Cesta, Carlo Combi, and François Laroussinie, editors, *21st International Symposium on Temporal Representation and Reasoning, TIME 2014, Verona, Italy, September 8-10, 2014*, pages 17–26. IEEE Computer Society, 2014. doi:10.1109/TIME.2014.13.

14 Francesca Rossi, Kristen Brent Venable, and Neil Yorke-Smith. Uncertainty in soft temporal constraint problems: A general framework and controllability algorithms forthe fuzzy case. *J. Artif. Intell. Res. (JAIR)*, 27:617–674, 2006. doi:10.1613/jair.2135.

15 Julie A. Shah, John Stedl, Brian Charles Williams, and Paul Robertson. A fast incremental algorithm for maintaining dispatchability of partially controllable plans. In Mark S. Boddy, Maria Fox, and Sylvie Thiébaux, editors, *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*, pages 296–303. AAAI, 2007. URL: http://www.aaai.org/Library/ICAPS/2007/icaps07-038.php.