Information Sciences 584 (2022) 784–807

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Adding flexibility to uncertainty: Flexible Simple Temporal Networks with Uncertainty (FTNU)

Roberto Posenato, Carlo Combi

Dipartimento di Informatica, Universita degli Studi di Verona, strada le Grazie 15, I-37134 Verona, Italy

ARTICLE INFO

Article history:
Received 8 January 2021
Received in revised form 26 August 2021
Accepted 2 October 2021
Available online 7 October 2021

Keywords:
Conditional simple temporal constraint network with uncertainty
Dynamic controllability
Guarded constraints
Contingency
Conditional propositions
Temporal constraint networks
Flexible temporal networks

ABSTRACT

A Flexible Simple Temporal Network with Uncertainty (FTNU) represents temporal constraints between *time-points*. Time-points are variables that must be set (*executed*) satisfying all the constraints. Some time-points are *contingent*. It means that they are set by the environment and only observed by the system *executing* the network. The ranges representing temporal constraints associated with contingent time-points (*guarded ranges)* can be shrunk during execution only to some extent to have more flexibility in the execution of the network. Subsets of time-points/constraints may be executed/considered in different contexts according to some observed conditions. The main issue here consists of determining whether all the time-points, under the control of the system, are executable in a way that all the specified constraints are satisfied for any possible occurrence of contingent time-points and any possible context. Such property is called *controllability*. Even though an algorithm was proposed for checking the controllability of such networks, we show that such an algorithm has a limit. Indeed, it does not determine the right bounds for guarded links, and, therefore, it doesn't permit the system to exploit the potential flexibility of the network. We then propose a new constraint-propagation algorithm for checking controllability, prove that such a new algorithm determines the right guarded ranges, and it is sound-and-complete. Thus, it can be used also for executing the network, by leveraging its flexibility.

© 2021 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

In the temporal constraint community, the concept of *controllability*, originally proposed in [1] and then refined in [2], has been extensively studied and applied in different domains [3–5]. Within a set of temporal constraints, each of them specifying an admissible range of temporal distances between two time-point variables, we may distinguish *contingent* and *requirement constraints*. While the last ones represent the usual constraints, connecting time-points both controlled by the system *executing* the network (i.e., assigning the values of such *ordinary time-points*), contingent constraints are between an ordinary time-point (called *activation time-point*) and a contingent one, which will occur within the specified range but will be set by the environment. The system, once executed the activation time-point, can only observe the occurrence of the corresponding contingent time-point. According to this interpretation, ranges of contingent constraints cannot be restricted during any execution of the network. On the other side, ordinary time-points will be executed by the system to properly manage the occurrence of contingent time-points, i.e., to satisfy all the given constraints. A network is *controllable* if such an execution exists. Recently, some research efforts highlighted the lack of flexibility in specifying contingent constraints. Indeed, in many real-world contexts, contingent constraints could be restricted to some extent, still preserving the property of not being under the control of the system [6–8]. Thus, the concept of *guarded link* was proposed. It has an

https://doi.org/10.1016/j.ins.2021.10.008
0020-0255/© 2021 The Author(s). Published by Elsevier Inc.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

unshrinkable core, i.e., a standard contingent constraint, surrounded by a shrinkable range (*external range*). During an execution, before the activation of a guarded link, the range can be shrunk (preserving the core) if it is necessary to execute the network correctly. Once the activation time-point is executed, the unshrinkable core is expanded to the external range, allowing the environment to use any value in such an expanded range.

Previous contributions introduced guarded links for Simple Temporal Constraint Networks with Uncertainty, i.e., STNPSU [6,8], and Conditional Temporal Constraint Networks with Uncertainty, allowing the representation of different execution scenarios, i.e., CSTNPSU [7]. The proposed algorithms for checking the controllability of such networks mainly verify the controllability of the networks by focusing on unshrinkable core ranges of guarded links. Thus, they are not able to explicitly consider the flexibility introduced by guarded links. Here, we prove that such algorithms were not able to derive the suitable external ranges for guarded links (and thus for the requirement constraints) when there are two or more guarded links.

In this paper, we propose *Flexible simple Temporal Networks with Uncertainty (FTNUs)* and a different algorithm for checking dynamic controllability. The main original contributions of our proposal can be summarized as in the following:

- we introduce FTNUs, which generalize and extend our preliminary proposal of CSTNPSUs [7];
- we propose a sound-and-complete algorithm checking the dynamic controllability of FTNUs. It solves the issue related to the derivation of the suitable external ranges.
- the new checking algorithm we propose is also able to find *maximal allowed unshrinkable ranges* for guarded links before their activation, thus making it applicable at runtime in real-world domains;
- finally, we extensively discuss a real-world scenario taken from a clinical domain, to motivate and build our proposal on solid foundations.

The paper is organized as follows. Section 2 discusses some literature of interest for our work, dealing with flexibility for temporal constraint networks. Section 3 introduces informally FTNUs through the discussion of a motivating scenario from a clinical domain. Section 4 to 5 formally describe our proposal of FTNU and related checking algorithm. Section 6 proposes an analysis and a controllable execution of the motivating scenario presented in Section 3, through `TNEditor`, a software application for designing and checking FTNUs and other kinds of temporal constraint networks. Finally, Section 7 concludes with a summary and sketches some future research directions.

2. Related work

Here we focus on the main contributions dealing with the specification and verification of temporal constraints, and explicitly considering uncertainty and flexibility. After the proposal of Simple Temporal Networks (STNs) [9], where it is possible to represent temporal constraints as duration ranges between time-points and check their consistency, further studies addressed a different kind of uncertainty on top of STNs, either related to temporal constraints or different/alternative execution paths, respectively. It is worth noting that the concept of flexibility is interpreted according to different flavors.

2.1. Flexibility and simple temporal networks

Flexibility in STNs refers to the property of having different possible schedules for a temporal plan, as the allowed delays between time-points are multiple.

In [10,11] an explicit characterization of flexibility for STNs is proposed, by taking into account the extension of ranges of possible values for time-points. Different flexibility metrics are discussed. Intuitively, a flexibility measure has to take into account both the number of possible values for time-points and the dependencies between values of different time-points, that make some time-points strictly bound to other ones, i.e., *rigid* [12]. Thus, the focus is on computing a range for any time-point, such that for every time-point one can freely choose a specific value inside its allowed range, without the risk of violating any existing constraint. Moreover, such ranges have to be independent. Indeed, for any time-point, choosing its value within the derived range has to be without any consequence for the choice of values for other time-points. This notion of independence is the ground for the definition of a flexibility metric for STNs [10].

Flexibility metrics for STNs is also the topic discussed in [13]. Flexibility has been dealt with by considering the measures of an STN solution space (and its related geometric properties). Such an approach also motivated a set of desiderata for general flexibility metrics. Two new geometrically-inspired flexibility metrics are also proposed. The first one captures the proportion of valid schedules that are most at risk of becoming invalidated when perturbed, while the second one evaluates how many schedules can remain consistent even when there is some kind of perturbation.

In Simple Temporal Problems with Preferences (STPPs) [14], the lack of flexibility of hard temporal constraints is solved by adding preferences to the temporal constraints. A preference function associates a preference value to each constraint between time-points. Constraint propagation and consistency checking are suitably extended to deal with such kinds of constraints and, under some general conditions for preference functions, consistency checking is tractable.

Moving to interval-based qualitative temporal constraints, an extension to the well known Allen's interval algebra is proposed in [15] to deal with some kind of flexibility. In this case, the paper focuses on qualitative temporal knowledge expressed through binary relations between intervals and proposes an extension of the classical Interval Algebra (IA) to deal

also with flexibility and uncertainty. To this end, different types of temporal constraints are introduced, i.e., (i) soft constraints, allowing the designer to specify preferences among the possible solutions for a scheduling; (ii) constraints with a priority, specifying the "importance" of having the constraint satisfied; (iii) uncertain constraints, expressed through constraints with priorities related, in this case, to their degree of plausibility. In such a fuzzy approach, the consistency of a constraint network is a gradual notion and a solution is ranked with a degree of satisfaction.

In [16], the authors propose a system for generalizing and robustly executing a plan enriched with temporal constraints, expressed through an STN. When unexpected changes in the represented world happen, the system allows one to select from numerous valid (sub)plan fragments that are consistent with the temporal constraints. Moreover, it can consider repeating parts of a plan, or it could omit the execution of some actions, as necessary. In these flexible changes to the original temporally constrained plan, the possible ambiguities of some original temporal constraints are suitably solved. Such an approach allows avoiding unnecessary replanning and rescheduling.

2.2. Introducing choices to make STNs more flexible

Choices/decisions are introduced in some proposals dealing with temporal planning to allow the system to change the specific set of time-points to schedule, to accommodate some possible unexpected situations.

Drake [17,18] is a system managing the verification and the execution of temporal plans. It allows the representation of choices by *Labeled STNs*, which consist of labeled constraints, according to the values of some discrete decision variables. Labeled STNs are equivalent to Disjunctive Temporal Networks (DTNs) [18,19]. The authors proved that, in general, Drake finds a dispatchable solution, which is more compact by over two orders of magnitude when compared with the equivalent one found by previous methods for DTNs. Drake maintains a high degree of flexibility by deferring choice until execution time. This way, autonomous systems executing the plan may avoid following predefined plans and wait for uncertainty unfolding before making decisions.

In [20,21] STNs are extended to represent decisions. Simple Temporal Networks with Decisions (STNDs) extend STNs by adding decision time-points: when they are executed, a truth-value for an associated Boolean proposition is set. According to this truth value, only a subset of time-points and constraints have to be executed, according to their associated labels. Decisions, as choices, are under the control of the system executing the network. Thus, decisions can be considered as a kind of flexibility mechanism, allowing alternative solutions for scheduling. An STND is consistent if at least a truth assignment to decision time-points allows a consistent execution of the associated STN. In [20] the authors prove that checking whether a given STND is consistent is a problem solvable in singly exponential (with respect to the number of decision time-points) deterministic time. In [21], the authors propose STND-HSCC2 (and implement it with a first experimental evaluation), a hybrid SAT-based consistency checking algorithm for STNDs that rules out inconsistent scenarios as early as possible, and provides all consistent scenarios (i.e., truth value assignments to decision time-points) and the related execution schedules. The algorithm is hybrid because a SAT-solver and a shortest path algorithm work in an intertwined way. Moreover, the authors prove that disjunctive temporal networks (DTNs) and STNDs are equivalent.

2.3. Flexibility and duration uncertainty

Simple Temporal Networks with Uncertainty (*STNUs*, [2]) add uncontrollable, non-shrinkable durations between time-points. The main property of such networks is that of *controllability*, i.e., the capability of executing the temporal network by satisfying all the given constraints for any possible occurrence of the uncontrollable (contingent) time-points.

Further approaches have been proposed introducing some kind of flexibility in temporal constraints. In [22] the notions of controllability of STNUs are extended to handle preferences and propose a model named *Simple Temporal Problems with Preferences and Uncertainty* (STPPUs). Preferences are modeled as functions, assigning a preference value to the range of values associated with the constraint between two time-points. An example of such preference could be the fuzzy one, returning values in the range $[0, 1]$. In the case of STPPUs, the consistency degree of the solutions must be taken into account, to identify the best solution with respect to the preferences, together with the satisfaction of the constraints. Here, the authors show that if all preference functions are semi-convex and a finite number of preferences is considered, testing the dynamic controllability of an STPPU network with respect to a preference threshold α has polynomial time complexity.

Recently, *Simple Temporal Networks with Partially Shrinkable Uncertainty* (*STNPSUs*) have been proposed in [6,8], where uncontrollable durations are made more flexible, i.e., partially shrinkable. Then, such kind of flexible contingent constraints has been studied to support a compact representation of temporal features for business sub-processes. Here, we will deepen in a more general context such concept of flexibility, i.e., referring to the capability of making contingent links partially reducible, still maintaining the property of being controllable.

In [23], STNUs are acknowledged as a way of representing temporal flexible plans. The focus of this contribution is on the problem of assigning bounds to requirement constraints. Such assignment must guarantee the dynamic controllability of the given network, but has also to be optimal with respect to the minimization of the total cost associated with the constraint bounds. This optimization problem is NP-hard, even for a linear cost function on the difference between the lower and upper bounds of the temporal distance between time-points.

2.4. Flexibility and condition-related uncertainty

Besides that related to uncontrollable time-points, another source of uncertainty is connected to conditions, allowing one to represent different execution scenarios depending on the truth values of some observable condition. *Conditional Simple Temporal Networks (CSTNs)* are proposed in [24]. Here, an *observation time-point* is a special type of time-point having associated a proposition. When executed, it allows the environment to set the proposition truth value. Thus, proposition truth values are decided by the environment at runtime. Dynamic consistency analysis guarantees that a strategy exists, which allows the execution of all (relevant) time-points satisfying all (relevant) temporal constraints, for any possible combination of proposition truth values revealed during the execution. The Conditional Temporal Problem (CTP), i.e., determining whether a CSTN is dynamically consistent, was already introduced in [25], where authors propose a solution based on the use of Disjunctive Temporal Networks.

In [26] CTPs are extended to deal with some kind of flexibility, expressed through fuzzy preferences. temporal constraints have been enriched with preferences and simple Boolean conditions are extended to fuzzy rules, activating the occurrence of some events based on fuzzy thresholds. Activation time-points (i.e., observation time-points) for events have associated preferences too. The concept of dynamic consistency has been extended accordingly, together with an algorithm to check it.

When dealing with flexibility for conditions, another approach could consist of extending temporal constraint networks also with decisions. They introduce a kind of flexibility, as different time-points can be selected for execution, according to the scenarios induced by uncontrollable conditions. This way, decisions may accommodate a schedule according to a possibly restricted subset of constraints, leading to a successful execution of the network. In this case, it is necessary to consider the potential interplay that occurs between controllable decisions and uncontrollable conditions. In [20] Conditional Simple Temporal Networks with Decisions (*CSTNDs*) are proposed. A CSTND may contain both contingent propositional time-points (associated with conditions) and controllable propositional time-points (associated with decisions). Decisions concur to define a scenario and their values are decided by the agent during the execution of the network, to dynamically manage different possible executions of the network, induced by uncontrollable observations. Here, the authors prove that checking whether any CSTND is dynamically consistent is a PSPACE-complete problem. Then, algorithms addressing two special classes of CSTNDs are proposed for CSTNDs containing decisions only, and for those where all decisions are set before starting the execution of the network, respectively.

2.5. Adding flexibility for durations and conditions

Let us finally consider contributions where flexibility is added, for temporal networks where both uncontrollable durations and conditions have been considered. Conditional Simple Temporal Networks with Uncertainty (*CSTNUs*) generalize both STNUs and CSTNs and allow the representation of both kinds of uncertainties simultaneously [27–29].

Conditional Simple Temporal Networks with Uncertainty and Decisions (*CSTNUDs*) are introduced in [30,31], where the authors consider the problem of temporal planning with uncertainty on both durations and execution paths. Moreover, they provide the semantics of dynamic controllability as a two-player game. The first player models the controller, while the second player models the environment, which represents the source of uncertainty. Accordingly, they reduce the verification of dynamic controllability to the characterization of a controller for a Timed Game Automaton (TGA). Thus, they provide an encoding from CSTNUDs into TGAs. An experimental evaluation of the proposed algorithm is then provided.

The semantics of dynamic controllability is given again through a two-player game also in [32], where the focus is on temporally-flexible reactive programs. The executive sets values for controllable time-points and makes choices for controllable choice points. Alternatively, the environment sets values for uncontrollable time-points and makes choices for uncontrollable choice (i.e., conditions) points. Obviously, the executive is allowed to observe what happened in the past till the current time, to dynamically schedule future controllable time-points and choices, accordingly. The execution of temporally-flexible reactive programs depends on the runtime state. Exceptions are thrown and caught at runtime in response to violated timing constraints. Successful program executions happen when exceptions are suitably handled. In this case, the concept of dynamic controllability has been extended and consists of guaranteeing that a program execution will complete, despite runtime constraint violations and uncertainty in runtime state. The dynamic controllability problem is framed as an AND/OR search tree over possible program executions. Only a subset of the possible executions that guarantees dynamic controllability is derived and represented as an AND/OR solution subtree.

Eventually, a preliminary proposal extending STNPSUs with conditional constraints introduces *Conditional Simple Temporal Networks with Partially Shrinkable Uncertainty CSTNPSUs*) [7]. In this case flexible contingent constraints are represented through guarded links, but controllability is mainly checked similarly as for CSTNUs, without completely considering the flexibility introduced by guarded links.

3. Introducing FTNUs through a motivating scenario

To motivate our proposal and to exemplify the different constructs of FTNUs, let us introduce a high-level specification of an excerpt of a clinical guideline related to the management of Adult Stroke Emergency [33].

Such guideline collects statements and suggestions from experts on how to deal with patients possibly having had a stroke, who require urgent treatments under specific temporal constraints.

The guideline suggests at the beginning four sequential (possibly complex) activities: *Stroke Recognition*, *Prehospital Management*, *General and Neurologic Assessment*, and *Imaging Computed-Tomography Scan*. Such activities correspond to the first 5 boxes discussed in the considered guideline [33], to acquire all the needed information for the following decision-making actions.

After the computed-tomography (CT) scan, two alternative paths are possible according to whether the *CT Scan* shows a hemorrhage. If there is a hemorrhage, it is necessary to have a consultation with a Neurologist or Neurosurgeon, followed by the appropriate care of the patient with a hemorrhage. Otherwise, the guideline recommends determining whether the patient is eligible for fibrinolytic therapy. If the patient is suitable for fibrinolysis, the drug *Recombinant Tissue Plasminogen Activator (rtPA)* is administered to the patient and, then, some patient management actions after such therapy have to be considered. On the other side, if the patient cannot afford fibrinolysis, a therapy based on Aspirin is administered and, then, stroke-management actions follow. After such fast actions, the patient has to be admitted to the Stroke Unit. Several temporal constraints have to hold when such actions are executed. Indeed, to treat the patient properly, for example, a maximum delay of 3 hours is allowed between the stroke recognition and the end of the fibrinolytic therapy.

The complex activity *General and Neurologic Assessment* is then further detailed. It starts with an *assessment of the vital signs* of the patient. If the patient is hypoxemic, then oxygen has to be provided. Then, glucose is checked and, in case, hypoglycemia is treated. After these two preliminary possible treatments, a neurologic assessment is performed followed by an electrocardiogram.

All these actions need to be represented also with respect to their temporal features. They consist of temporal constraints that have to be satisfied, to guarantee the clinically successful completion of each step of the guideline. The temporal constraints we will consider here have been set according to the guideline specification, and the temporal constraints graphically represented in the diagram in [page S819] [33].

Such temporal constraints will help clinical stakeholders in planning their work, as they can be aware of how long previous steps will take and how much freedom they have for performing their clinically relevant tasks. A further aspect we have to take into account is that activity durations are not completely under the control of the system supervising the execution of the guideline, as these activities are carried out by clinicians and their duration could vary according to patient-specific conditions.

Therefore, we need to formally represent such guidelines by considering both flexible temporal constraints and possibly different sets of actions when executing the guideline.

In this regard, let us now consider the basic features of FTNUs, the temporal networks we propose to model the (temporal) plans subtended by such clinical guideline.

An FTNU is a directed weighted graph, whose nodes represent real-valued variables (*time-points*), usually corresponding to the occurrence of events like the start or end of activities. We say that a time-point is *executed* when a value is assigned to it. An executed time-point cannot be re-executed.

Fig. 1 shows a graphical representation of the FTNU corresponding to the complex activity *General and Neurologic Assessment*, previously described. Time-points may represent either the start or the end of single activities. For example, V_S and V_E represent the start and the end of *vital signs assessment*, respectively. Certain time-points may be defined at design time as *contingent time-points*, meaning that their values are decided by the environment at runtime instead of the system executing the network. In our example, all the time-points corresponding to the end of activities are contingent, as they are executed by the medical stakeholders. Some time-points may be defined as *observation time-points*. Each observation time-point is associated with a propositional letter (boolean variable) that is set by the environment when the time-point is executed. Usually, we use '?' as a suffix, to denote the observation time-point (e.g., $R?$) of a proposition letter (e.g., r). In Fig. 1 time-points $P?$ and $Q?$ represent when hypoxemia and hypoglycemia states for the given patient become known, respectively. Other time-points correspond to either splitting or merging points for alternative execution paths. For example, time-point $\times_p$ represents when condition p ("is the patient hypoxemic?") is checked and different actions may be taken accordingly. Finally, two special time-points represent the beginning (Z) and the ending (E) of the network execution, respectively.

Two different kinds of edges may be specified in an FTNU. Edges $A \xrightarrow{[x,y]} B$, called *requirement links*, represent a lower and an upper bound constraint on the distance between the two time-points it connects; for example, $\mathrm{Z} \xrightarrow{[0,25]} \mathrm{E}$ represents a constraint expressing that E, i.e., the end of the assessments, must occur between 0 and 25 time-units after the occurrence of Z, i.e., the starting time of the assessment: $0 \leqslant \mathrm{E} - \mathrm{Z} \leqslant 25$.

For the second kind of edges, each contingent time-point has *one special* incoming edge, called its *guarded link* and drawn as a double line, that represents the constraint that the environment has to satisfy when it decides to execute the contingent time-point. For example, $V_S \xRightarrow{[[2,4][6,8]]} V_E$ is the guarded link associated with contingent time-point V_E, corresponding to the end of vital signs assessment, where V_S is an ordinary time-point called the *activation time-point*, corresponding in this case to the start of the vital signs assessment. Label $[[2,4][6,8]]$ corresponds to the quasi-contingent duration range $[2,8]$ augmented with two *guards*, the *lower guard*, having value 4, and the *upper guard*, having value 6 [6]. During the execution, the duration range $[x,y]$, where $x \geqslant 2$ and $y \leqslant 8$, allowed for executing the assessment of the vital signs may be modified by the system, complying with the corresponding guards, i.e., $x \leqslant 4$ and $y \geqslant 6$ until the *activation of the link*. The activation

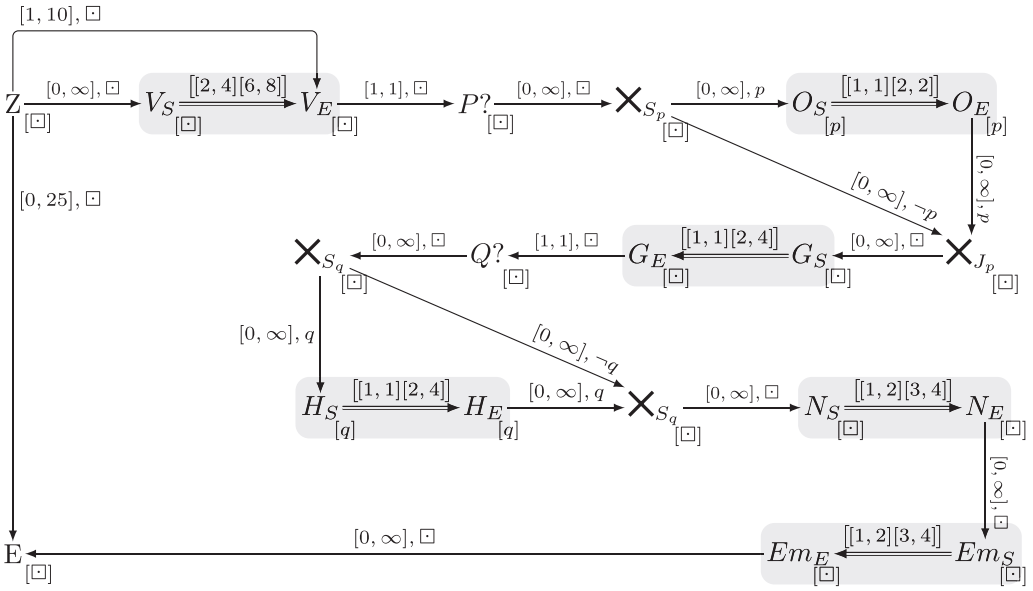


Fig. 1. The FTNU representing an excerpt of the clinical guideline for the management of Adult Stroke Emergency. Different time-points are represented by Uppercase letters (possibly with subscript *S*, for starting time-points of an activity, or *E*, for ending time-points of an activity). Here *V* corresponds to vital sign assessment; *O* corresponds to oxygen provision; *G* corresponds to glucose checking; *H* corresponds to hypoglycemia treatment; *N* corresponds to neurologic assessment; *Em* corresponds to making an electrocardiogram. Special time-points Z and E represent the first and the last time-points to be executed, respectively. Time-points denoted by × represent either splitting points or merging points, according to different execution scenarios. Execution scenarios are represented through labels associated with time-points and edges. More particularly, refer to time-points to be executed in every scenario and to constraints that have to hold in every possible execution. *p* represents the scenario where the patient is hypoxemic, while *q* corresponds to the scenario where the patient has hypoglycemia. Grey parts contain time-points related to the start and end of some external activities, which are not under the control of the system as for their duration (expressed as a flexible contingent constraint). Edges with alphanumeric labels represent temporal constraints between the connected time-points.

of the link triggers just after the execution of the activation time-point V_S. Suppose that the duration range is $[[3,4][6,6]]$ just after the execution of V_S. Then, $[[3,3][6,6]] \equiv [3,6]$ becomes the (pure) contingent range of the link, and it is made available to the physician for executing V_E. That means, once V_S is executed, the physician can assess the vital signs of the patient in a time span from 3 to 6 time-units, which corresponds to set V_E to any value such that $V_E - V_S \in [3,6]$ holds. The specific execution time of V_E is *uncontrollable* since it is decided by the environment, i.e., the physicians, and it will be known only when it happens.

Each node/requirement link may have a propositional label that specifies when the node/requirement link has to be executed/satisfied. In particular, at the start of an execution, all the propositions specified in the network have no values. As soon as an observation time-point is executed, the corresponding proposition is set. As soon as all the propositions present in a label are set, the label assumes a truth value: $\bot$ or $\top$. If the label is associated with a node, the node has to be considered for the execution only when the label value is $\top$. If the label is associated with a constraint, the constraint is ignored only when the label value becomes $\bot$. The full specification of proposition values is called *scenario*.

More formally, given a set $\mathcal{P}$ of propositional letters, a *propositional label* ℓ is any conjunction of literals, where a literal is either a propositional letter $p \in \mathcal{P}$ or its negation $\neg p$. The empty label is denoted by $\boxdot$. The *label universe of* $\mathcal{P}$, denoted by $\mathcal{P}^*$, is the set of all labels whose literals are drawn from $\mathcal{P}$. Two labels $\ell_1, \ell_2 \in \mathcal{P}^*$ are *consistent* if and only if their conjunction $\ell_1 \wedge \ell_2$ is satisfiable and a consistent label ℓ_1 *entails* a consistent label ℓ_2 (written $\ell_1 \supset \ell_2$) if and only if all literals in ℓ_2 appear in ℓ_1 too.

4. Flexible Temporal Constraint Networks

Concerning to the different kinds of flexibility previously introduced, in this paper we will explicitly focus on making contingent constraints more flexible, i.e., shrinkable, to be able to deal with two different uncertainties, namely that coming from uncontrollable durations and that related to uncontrollable conditions (i.e., observations).

Flexible simple Temporal Networks are an extension of CSTNPSUs [7], which are in their turn based on CSTNUs [27–29]. FTNUs extend contingent links for enabling a more flexible temporal constraint management.

4.1. Formal definition

We propose a formal definition of FTNUs that is a specialization of the one proposed in [7]: here it is required also that the set of nodes always contains a special node, Z, that represents the first node to execute.

A *Flexible simple Temporal Network with Uncertainty* (FTNU) is a tuple $(\mathcal{T}, \mathcal{P}, \mathcal{OT}, O, L, \mathcal{C}, \mathcal{G})$, where

- $\mathcal{T}$ is a set of real-valued variables. Such variables are the *time-points* of the network. $\mathcal{T}$ always contains the time-point Z that is assumed to be the first time-point to be executed, i.e., its value is set by the executing agent;
- $\mathcal{P} = \{p, q, r, s, \ldots\}$ is a finite set of propositional letters.
- $\mathcal{OT} \subseteq \mathcal{T}$ is a set of *observation time-points*.
- $O : \mathcal{P} \to \mathcal{OT}$ is a bijection that, given a propositional letter, assigns a unique observation time-point to it. The truth value of a proposition is set by the environment when its observation time-point is executed. Usually, if r is a proposition, its observation time-point has the name $R?$.
- $L : \mathcal{T} \to \mathcal{P}^*$ is a function that assigns a propositional label to each time-point $X \in \mathcal{T}$. A (propositional) label is a conjunction of literals. A true-valued label of a node indicates that the node has to be executed.
- $\mathcal{C}$ is a set of *labeled requirement links*. Each requirement link is denoted as $(u \leqslant Y - X \leqslant v, \alpha)$ or as $X \xrightarrow{[u,v],\alpha} Y$; $X, Y \in \mathcal{T}, u, v \in \mathbb{R}$ with $u \leqslant v$ and $\alpha \in \mathcal{P}^*$. A requirement link has to be satisfied when its label has (will have) a true-value. In other words, a requirement link can be ignored only when its label becomes false during an execution.
- $\mathcal{G}$ is a set of *guarded links*. Each guarded link is represented as $(A, \left[[x, x'][y', y]\right], C)$ or as $A \xrightarrow{\left[[x,x']|[y',y]\right]} C$; A and C are time-points, called *activation* and *contingent* time-points, respectively; $x, y \in \mathbb{R}$ are the external bounds; $x', y' \in \mathbb{R}$ are the *guards*. $[x, y]$ is called *external range* of the guarded link. It must hold $0 < x \leqslant x' \leqslant y' \leqslant y < \infty$, and $L(A) = L(C)$. Moreover, if $(A_i, \left[[x_i, x_i'][y_i', y_i]\right], C_i)$ and $(A_j, \left[[x_j, x_j'][y_j', y_j]\right], C_j)$ are two different guarded links in $\mathcal{G}$, C_i and C_j will be distinct time-points.
- For each labeled constraint $(u \leqslant Y - X \leqslant v, \alpha)$, $\alpha \supset L(Y) \wedge L(X)$. Such a property is called *constraint label coherence* [24].
- For each literal q or $\neg q$ appearing in α, $\alpha \supset L(O(q))$. Such a property is called *constraint label honesty* [24].
- For each $Y \in \mathcal{T}$, if literal q or $\neg q$ appears in $L(Y)$, then $L(Y) \supset L(O(q))$, and $O(q)$ has to occur before Y, i.e., $(\epsilon \leqslant Y - O(q) \leqslant +\infty, L(Y)) \in \mathcal{C}$ for some $\epsilon > 0$. Such a property is called *time-point label honesty* [24].

We assume that the environment/external agent decides the duration of a guarded link once the activation time-point has been executed without any further restriction or condition. Therefore, guarded links can be represented without any propositional label. Indeed, if it is necessary to have different guarded link durations according to some condition, it is always possible to layout different guarded links, one for each possible duration, in different paths that belong to different scenarios, one for any possible condition value. Between two ordinary time-points, it is possible to have more requirement links, each with a different propositional label. Such requirement links represent the different possible constraints that must be satisfied by the two time-point assignments according to the resulting scenario of an execution. For example, let us consider two time-points, X and Y, that have to be executed in any scenario (their label is), and that are involved in two requirement links, $X \xrightarrow{[1,10],p} Y$ and $X \xrightarrow{[5,10],\neg pq} Y$. The requirement link $X \xrightarrow{[1,10],p} Y$ dictates that in all scenarios in which p is $\top$, the values of X and Y must satisfy the constraint $(1 \leqslant Y - X \leqslant 10)$, while the requirement link $X \xrightarrow{[5,10],\neg pq} Y$ requires that in all scenarios in which p is $\bot$ and q is $\top$, the values of X and Y must satisfy the constraint $(5 \leqslant Y - X \leqslant 10)$, a more restrictive constraint.

The three properties about labels–*constraint label coherence, constraint label honesty*, and *time-point label honesty*–are necessary to guarantee that the label specification allows having scenarios in which it is possible to consider time-points and constraints with well-defined labels and without label inconsistencies.

Since Z has to be the first time-point to be executed, we assume that, for each time-point $X \in \mathcal{T}$ such that $X \neq Z$, there is a requirement constraint of the form $Z \xrightarrow{[0,\infty],\boxdot} X$. For sake of clarity, we will not explicitly represent most of such constraints in the graphical representation of FTNUs.

If a time-point X is executed at time t, then such execution is represented by adding $Z \xrightarrow{[t,t],\boxdot} X$ to the network.

4.2. The DC-checking problem

An *execution strategy* is a function that determines a schedule for the time-points of a network. Considering FTNUs, a *dynamic execution strategy* is a function that determines a schedule considering the values that the environment sets to the propositional letters and to the contingent time-points as time passes.

A dynamic execution strategy for an FTNU is *viable* when it guarantees that all relevant constraints will be not violated, no matter which truth values for propositions (*scenario*) and durations for guarded links (*situation*) are incrementally acquired over time.

An FTNU that admits a viable dynamic execution strategy is said *dynamically controllable (DC).* Given an FTNU, the *DC-checking problem* consists of verifying whether it is DC. Usually, such a problem is considered at design time.

There are two important results found in two different kinds of temporal constraint networks, discussed in [29,34], that allow us to characterize better the DC-checking problem for FTNUs.

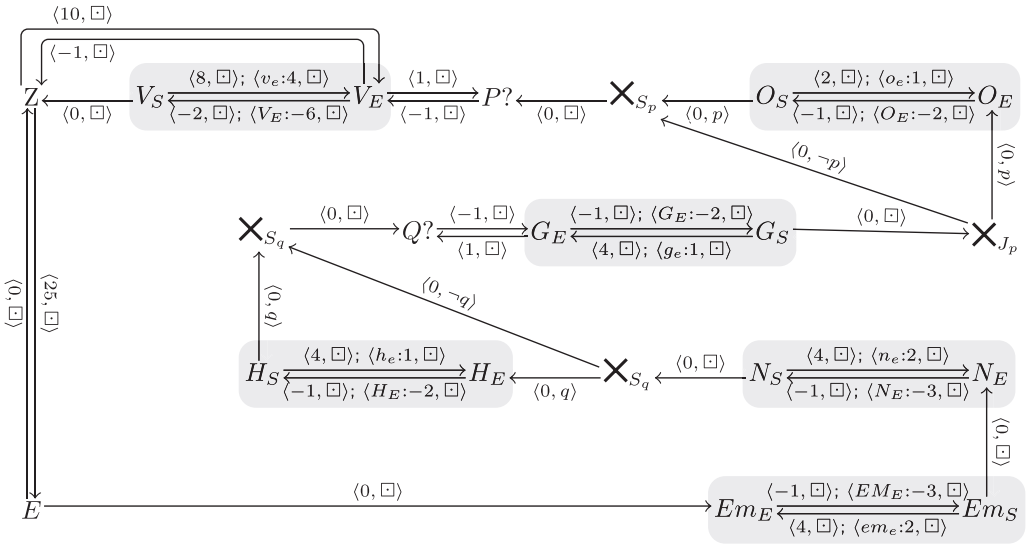


Fig. 2. The distance graph associate to the FTNU depicted in Fig. 1.

The first result is related to Conditional Simple Temporal Networks (CSTNs), temporal constraint networks that can be viewed as FTNUs where there are no guarded links. In [34], the authors showed that the DC-checking problem can be also defined for CSTNs and that it is PSPACE-complete. Therefore, it is straightforward to state that the DC-checking problem in FTNUs is PSPACE-hard.

The second one is related to the Conditional Simple Temporal Networks with Uncertainty (CSTNUs), networks that can be viewed as FTNUs where guarded links have no flexibility (i.e., the external range is equal to the unshrinkable range in each guarded link). In [29], the authors showed that for solving the CSTNU DC-checking problem and for executing a CSTNU instance, it is possible to consider a *streamlined* representation of a CSTNU where nodes are not labeled, i.e., labels are present only on requirement links. Such an alternative representation does not require any transformation. It simply requires ignoring the node labels because it is shown that if the network is represented considering the *constraint label coherence, constraint label honesty, and time-point label honesty* properties, then node labels are redundant, and they can be ignored without loss of generality.[1] Since ignoring node labels makes simpler the analysis of the DC-checking problem, hereinafter we consider only *streamlined FTNUs* for such a goal. Fig. 4a depicts an example of a streamlined FTNU.

To solve the DC-checking problem, an FTNU is represented in an equivalent form, i.e., as a *distance graph*, similarly to the approach in [7]. Given an FTNU, its *distance graph* $\mathcal{D} = (\mathcal{T}, \mathcal{E})$ is a graph having the same set of nodes and edges determined considering the bounds of all requirement/guarded ranges of the original links [6,7,35]. In particular, for each link $S \xrightarrow{[x,y],\ell} T$ of an FTNU instance, in the corresponding distance graph there are two *ordinary edges*: $S \xrightarrow{\langle y,\ \ell\rangle} T$ for constraint $(T - S \leqslant y, \ell)$, and $S \xleftarrow{\langle -x,\ \ell\rangle} T$ for constraint $(S - T \leqslant -x, \ell)$ (note the change of arrow style). For each guarded link between a pair of time-points A and C, in the distance graph there are two *ordinary edges* representing the external bounds as just described and two other edges, called *lower* and *upper-case edges*. Such edges are useful for representing the contingent property of contingent time-points as temporal constraints. In particular, a *lower-case* edge, $A \xrightarrow{\langle c:x',\ \boxdot\rangle} C$, represents the property that the contingent time-point C cannot be forced to be set to an instant greater than the instant x' after A. In other words, it is not possible to have a constraint $A \xleftarrow{\langle -x'',\ \boxdot\rangle} C$, with $x' < x''$, in the network. As regards upper-case edges, an *upper-case* edge, $A \xleftarrow{\langle C:-y',\ \boxdot\rangle} C$, represents the fact that C cannot be forced to be set to an instant less than y' after A. In other words, it is not possible to have a constraint $A \xrightarrow{\langle y'',\ \boxdot\rangle} C$, with $y'' < y'$, in the network. The lower/upper-case edges are necessary to determine the DC property of the network following the approach proposed by Morris et al. [35].

Fig. 2 depicts the distance graph associated with the streamlined FTNU in Fig. 1. To represent a distance graph compactly, we represent different edges from a source X to a destination Y as a single edge with more labeled values separated by ';' (cf. the edges representing contingent links in Fig. 2).

[1] Although the streamlined result allows a simplification of FTNU definition, we prefer to maintain the specification of node labels because node labels are useful for checking the *constraint label coherence, constraint label honesty, and time-point label honesty* properties, and for deciding, at runtime, which nodes to execute in a faster way.

5. A new DC-checking algorithm for FTNUs

The DC-checking problem for FTNUs can be solved by creating an equivalent more explicit network adding new equivalent constraints. If during such a process, a negative cycle is found, the network is not DC for sure, otherwise, at least one solution is present in the resulting network. Such a solution can be calculated in an incremental way, reacting to the outcomes of observations and execution time of contingent time-points.

The new constraints are added using some *propagation* rules. In [7], we extended the constraint-propagation algorithm proposed by Hunsberger and Posenato for the DC-checking problem in CSTNUs [29] to solve the same problem in CSTNPSUs (a well-defined CSTNPSU is equivalent to an FTNU). The algorithm proposed in [7] works on the distance graph of a given FTNU and, applying the *propagation rules* in Table 1, it recursively adds new edges until either a negative cycle is found or no new constraint is added.

The new DC-checking algorithm for FTNUs, Algorithm 1, is a correct extension of the algorithm proposed in [7]: our version considers also another set of rules (see Table 2) because the rules in Table 1 are not sufficient, as we will show in the following.

Algorithm 1. FTNU-DC-Check(G)

Input: $G = (\mathcal{T}, \mathcal{P}, \mathcal{OT}, O, L, \mathcal{C}, \mathcal{G})$: a FTNU instance
Output: the dynamic controllability status of G.

G' = distance graph of G;
$h = M|\mathcal{T}|$, where M is the maximum absolute value of any negative edge;
foreach $X \in \mathcal{T}$ **do**
  Add to G' bounds $\mathrm{Z} \xleftarrow{\langle 0, \boxdot \rangle} X$ and $\mathrm{Z} \xrightarrow{\langle h, \boxdot \rangle} X$;
do
  G' = Apply rM_1 and rM_2 to G'; // Label Modification
  G' = Apply all rules rG_1–rG_9 to G'; // Edge Generation
  if *(there is a negative semi-reducible cycle)* **then return** $\bot$;
while *(rules add/change edges)*;
return $\top$

As regards rules in Table 1, they fall into two main types. The first type consists of rules rG_1–rG_4^* that extend the edge-generation rules proposed in [35] for STNUs; the second type is composed of two label-modification rules, rM_1–rM_2, that "clean" propositional labels considering the edges from observation nodes to Z.

If a rule determines a new edge, the new associated label, e.g., $\ell\ell'$, is the conjunction of the labels of its parent edges, e.g., ℓ and ℓ'. When the conjunction determines an unsatisfiable label (e.g., $p\neg rq\neg q$), then the new edge is not generated (or kept). The $\aleph$ symbol represents a conjunction, possibly empty, of one or more upper-case names of contingent nodes.

A generated edge containing a labeled value having a non-empty $\aleph$ is called *conjunct-upper-case edge* and represents an extension of the *wait* constraint introduced by Morris [2]. For example, the conjunct-upper-case edge $\mathrm{Z} \xleftarrow{\langle \aleph: -v, \boxdot \rangle} Y$ stands for the following *extended wait* constraint: *as long as all the contingent time-points in* $\aleph$ *do not occur, Y must be executed at or after v time units from the beginning of the execution.* If any contingent time-point in $\aleph$ occurs at time $t < v$, then the constraint is satisfied for any occurrence of Y at a time $\geqslant t$, i.e., all the contingent time-points specified in $\aleph$ are in a disjunctive condition for disabling the wait. Thus, all rules only determine new ordinary or *conjunct-upper-case edges*.

Rule rG_4^*, as well as rM_1 and rM_2, can handle also a new kind of labels, named *q-labels*, that can contain special literals, named *q-literals* [29]. If $q \in \mathcal{P}$, then $?q$ is a *q-literal*. $?q$ is $\top$ only when the value of proposition q is not set, i.e., *unknown*. The concept of the propositional label is extended to q-literal as follows: "a *q-label* is a conjunction of literals and/or q-literals". $\mathcal{Q}^*$ denotes the set of all q-labels [29]. To represent the conjunction of literals and q-literals, we prefer to introduce a new operator, called $\bigstar$ operator. Informally, if constraint C_i has label q, and constraint C_j has label $\neg q$, then *both* C_i and C_j must hold as long as q is *unknown*, which is represented by $q\bigstar\neg q = ?q$.

In the following, we summarize the main scope of each rule to give an idea of how the set can help to find a more explicit constraint network. In such a summary, we will present some example using the FTNU depicted in Fig. 3a: it is a modified excerpt of Fig. 2 where we require that 1) observation time-point $P?$ must be executed at most 3 time units before contingent time-point V_E and not after it, and 2) the overall execution time (i.e., the constraint between Z and V_E) may have different values according to observation p. Such modifications are set to show the effect of some rules more effectively.

Rule rG_1 is a specialization of the standard rule that determines the minimal distance between two time-points. It considers the new distance only when labeled values have consistent propositional labels. Since positive distances to Z are meaningless, the rule does not generate edges with positive weights. Fig. 3b depicts the new values determined by

Table 1
Edge-generation rules (1) of FTNU-DC-Check algorithm.

Rule	Conditions	Pre-existing and Generated Edges
rG_1	$u - v < 0$, $\alpha\beta \in \mathcal{P}^*$	$\mathrm{Z} \xleftarrow[\langle \aleph{:}u - v, \alpha\beta\rangle]{\langle \aleph{:}-v, \beta\rangle} Y \xleftarrow{\langle u, \alpha\rangle} X$
rG_2	$x - v < 0$, $C \notin \aleph$, $\beta \in \mathcal{P}^*$	$\mathrm{Z} \xleftarrow[\langle \aleph{:}x - v, \beta\rangle]{\langle \aleph{:}-v, \beta\rangle} C \xleftarrow{\langle c{:}x, \boxdot\rangle} A$
rG_3	$\beta \in \mathcal{P}^*$	$\mathrm{Z} \xleftarrow[\langle C\aleph{:}-y' - v, \beta\rangle]{\langle \aleph{:}-v, \beta\rangle} A \xleftarrow{\langle C{:}-y', \boxdot\rangle} C$
rG_4^*	$m = \max\{-v, -w - x\}$, $C \notin \aleph\aleph_1$, and $\beta, \gamma \in \mathcal{Q}^*$.	$Y \xrightarrow[\langle \aleph\aleph_1{:}m, \beta\star\gamma\rangle]{\langle C\aleph{:}-v, \beta\rangle} \mathrm{Z} \xleftarrow{\langle \aleph_1{:}-w, \gamma\rangle} A \underset{\langle -x, \boxdot\rangle;\ \langle C{:}-y', \boxdot\rangle}{\overset{\langle y, \boxdot\rangle;\ \langle c{:}x', \boxdot\rangle}{\rightleftarrows}} C$
rM_1	$\beta \in \mathcal{Q}^*$, $\tilde{p} \in \{p, \neg p, ?p\}$	$\mathrm{Z} \xleftarrow[\langle \aleph{:}-w, \beta\rangle]{\langle \aleph{:}-w, \beta\tilde{p}\rangle} P?$
rM_2	$m = \max\{-v, -w\}$ $\beta, \gamma \in \mathcal{Q}^*$, $\tilde{p} \in \{p, \neg p, ?p\}$	$Y \xrightarrow[\langle \aleph\aleph_1{:}m, \beta\star\gamma\rangle]{\langle \aleph{:}-v, \beta\tilde{p}\rangle} \mathrm{Z} \xleftarrow{\langle \aleph_1{:}-w, \gamma\rangle} P?$

$v > 0, w > 0, \mathrm{Z}, A, C, X, Y \in \mathcal{T}$; C is contingent; $P? \in \mathcal{OT}$; each of $\aleph$ and $\aleph_1$ is a conjunction of one or more upper-case names of contingent nodes, possibly empty.

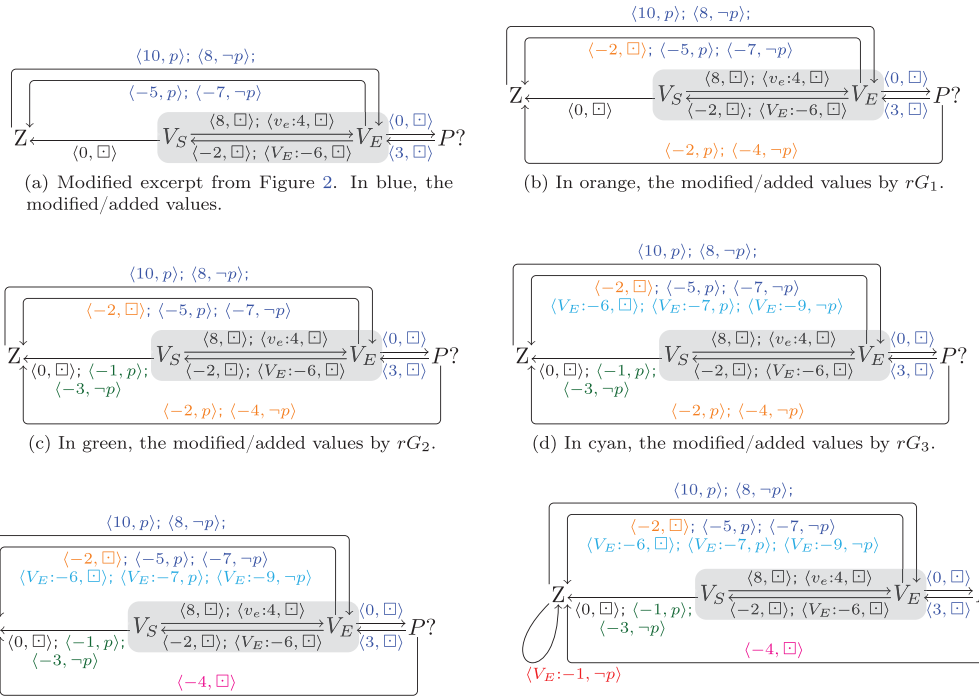
Table 2
(Additional) edge-generation rules (2) of FTNU-DC-Check algorithm.

rG_5	$u + v \geq 0, \alpha\beta \in \mathcal{P}^*$	$\mathrm{Z} \xrightarrow[\langle \urcorner{:}v + u, \alpha\beta\rangle]{\langle \urcorner{:}v, \beta\rangle} Y \xrightarrow{\langle u, \alpha\rangle} X$
rG_6	$c \notin \urcorner$	$\mathrm{Z} \xrightarrow[\langle c\urcorner{:}x' + v, \beta\rangle]{\langle \urcorner{:}v, \beta\rangle} A \xrightarrow{\langle c{:}x', \boxdot\rangle} C$
rG_7	$v - y' \geq 0$, $c \notin \urcorner$	$\mathrm{Z} \xrightarrow[\langle \urcorner{:}v - y', \beta\rangle]{\langle \urcorner{:}v, \beta\rangle} C \xrightarrow{\langle C{:}-y', \boxdot\rangle} A$
rG_8	$C \notin \aleph, c \notin \urcorner, \aleph$ and $\urcorner$ have no common names, $[(\alpha\beta \in \mathcal{P}^*)$ or $(\alpha \in \mathcal{Q}^\star \setminus \mathcal{P}^\star$, $\beta \in \mathcal{P}^\star$ and α contains $\beta.)]$.	$\mathrm{Z} \xleftarrow{\langle \aleph{:}-v', \alpha\rangle} A \underset{\langle -x, \boxdot\rangle;\ \langle C{:}-y', \boxdot\rangle}{\overset{\langle v - v', \boxdot\rangle\ ;\ \langle c{:}x', \boxdot\rangle}{\rightleftarrows}} C$ with $\mathrm{Z} \xrightarrow[\langle \urcorner{:}v, \beta\rangle]{} C$
rG_9	$C \notin \aleph, c \notin \urcorner, \aleph$ and $\urcorner$ have no common names, $[(\alpha\beta \in \mathcal{P}^*)$ or $(\alpha \in \mathcal{Q}^\star \setminus \mathcal{P}^\star$, $\beta \in \mathcal{P}^\star$ and α contains $\beta.)]$.	$\mathrm{Z} \xrightarrow{\langle \urcorner{:}v, \beta\rangle} A \underset{\langle v - v', \boxdot\rangle\ ;\ \langle C{:}-y', \boxdot\rangle}{\overset{\langle y, \boxdot\rangle;\ \langle c{:}x', \boxdot\rangle}{\rightleftarrows}} C$ with $\mathrm{Z} \xleftarrow[\langle \aleph{:}-v', \alpha\rangle]{} C$

$v \geq 0, v' \geq 0, 0 < x \leq x' \leq y' \leq y, -\infty < u < \infty, \mathrm{Z}, A, C, X, Y \in \mathcal{T}$; C is contingent; $P? \in \mathcal{OT}$; each of $\aleph$ and $\aleph_1$ is a conjunction of one or more upper-case names of contingent nodes, possibly empty; each of $\urcorner$ and $\urcorner_1$ is a conjunction of one or more lower-case names of contingent nodes, possibly empty.

the application of rG_1 to triples (V_E, V_S, Z), and $(P?, V_E, Z)$. Even though value $\langle -2, \boxdot\rangle$ is redundant, we reported it only for showing the bare application of the rule. Value $\langle -2, \boxdot\rangle$ is redundant because $\langle -5, p\rangle$ and $\langle -7, \neg p\rangle$ represent all possible scenarios, and they are more restrictive than $\langle -2, \boxdot\rangle$. An efficient implementation of the rule should add only significative values to an edge. Rule rG_1 determines that $P?$ has to occur at least 2 time units after Z in case p is true, 4 time units otherwise. This is apparently quite strange because the value of p is revealed only after the execution of $P?$. Rule rM_1 will fix these values removing the literal p and $\neg p$ as shown in Fig. 3e.

Rule rG_2 is a specialization of rule rG_1 when one edge is a lower-case one. The interesting thing about this rule is that the new edge is a new ordinary/conjunct-upper-case one meaning that lower-case edges are not propagated. The rule determines the minimal distance of the activation time-point from Z considering the minimal distance of the corresponding contingent time-point and the minimal duration of the contingent link. Fig. 3c shows the new values determined by applying rG_2.



(a) Modified excerpt from Figure 2. In blue, the modified/added values.

(b) In orange, the modified/added values by rG_1.

(c) In green, the modified/added values by rG_2.

(d) In cyan, the modified/added values by rG_3.

(e) In magenta, the modified values by rM_1.

(f) In red, the modified/added values by rG_1.

Fig. 3. Modified excerpt from Fig. 2 for showing some rule propagations.

Such values are added to the edge (V_S, Z). Let us consider the value $\langle -3, \neg p\rangle$. It is the result of the combination of the minimal contingent duration $\langle v_e{:}4,\ \boxdot\rangle$ and the minimal distance of V_E $\langle -7, \neg p\rangle$. In other words, since in scenario $\neg p$, the contingent time-point V_E has to occur at least 7 time units after the start Z and the minimal duration of the contingent link is 4 time unit, the activation time point has to occur at least 3 time unit after the start in scenario $\neg p$. The new values $\langle -1, p\rangle$ and $\langle -3, \neg p\rangle$ make the value $\langle 0,\ \boxdot\rangle$ redundant (and it should be removed in an efficient implementation of the rules).

Rule rG_3 is still a specialization of rule rG_1. In this case, the rule dictates how to combine an upper-case edge with an ordinary/conjunct-upper-case edge to generate a new conjunct-upper-case one. If the new edge is generated, then its labeled value contains a conjunct-upper-case label augmented by the contingent name associated with the upper-case edge. Fig. 3d depicts the new values for the edge (V_E, Z) determined by applying rG_3. They represent the latest execution times for the contingent time-point V_E in different scenarios when the guarded link duration assumes its upper-guard value. Also in this case we depicted the redundant value $\langle V_E : -6,\ \boxdot\rangle$, determined considering the redundant value $\langle 0,\ \boxdot\rangle$ in (V_C, Z).

The scope of rG_4^* is to simplify an already present edge removing a contingent name from its labeled value. In short, it checks if a contingent time-point C, whose name is present in a conjunct-upper-case edge $\mathrm{Z} \xleftarrow{\langle C\aleph:-v,\ \beta\rangle} Y$, can occur before the time v. If no, then the contingent time-point name can be dropped by the conjunct-upper-case label $C\aleph$ because it is meaningless. Such simplification is realized by adding the edge $\mathrm{Z} \xleftarrow{\langle \aleph\aleph':-v,\ \beta\bigstar\gamma\rangle} Y$. If C can occur after time v, then the added edge represents the fact that, in any case, it is necessary to wait till the occurrence of C for evaluating the original constraint $\mathrm{Z} \xleftarrow{\langle C\aleph:-v,\ \beta\rangle} Y$. Since the occurrence of C depends also on the constraint of the associated activation time-point A to Z, the new edge has to consider also the labeled value of this last constraint, from which the presence of $\aleph'$ and γ.

The second type of propagation rules, composed of rM_1 and rM_2, has the scope to manage the uncertainty related to observation nodes in a similar way as the rule rG_4^* manages the uncertainty associated with the contingent links.

Rule rM_1 simplifies the propositional label of edges from observation time-points to Z. For example, let assume that there is the edge $\mathrm{Z} \xleftarrow{\langle \aleph:-w,\ \beta\tilde{p}\rangle} P?$, where $\tilde{p} \in \{p, \neg p, ?p\}$. Such an edge represents that $P?$ can be executed at time $t \geqslant w$ after Z when $\beta\tilde{p}$ is not false. Since $\tilde{p}$ can be evaluated only when $P?$ is executed, it cannot contribute to the truth-value of label $\beta\tilde{p}$. Therefore, the literal $\tilde{p}$ can be removed from the label. Edges like $\mathrm{Z} \xleftarrow{\langle \aleph:-w,\ \beta\tilde{p}\rangle} P?$ are possible because of results of the application of other propagation rules. Fig. 3e depicts the new values for the edge $(P?, \mathrm{Z})$ determined by applying rM_1. The values $\langle -2, p\rangle$ and $\langle -4, \neg p\rangle$ become $\langle -2,\ \boxdot\rangle$ and $\langle -4,\ \boxdot\rangle$. Since -4 is more restrictive of -2, only $\langle -4,\ \boxdot\rangle$ is considered.

Rule rM_2 is the generalization of rM_1 to other time-points. In summary, if an edge from Y to Z has a label containing a proposition letter p that cannot have a truth-value at the time represented by the constraint, then a new constraint can be added where the label does not depend on the proposition p.

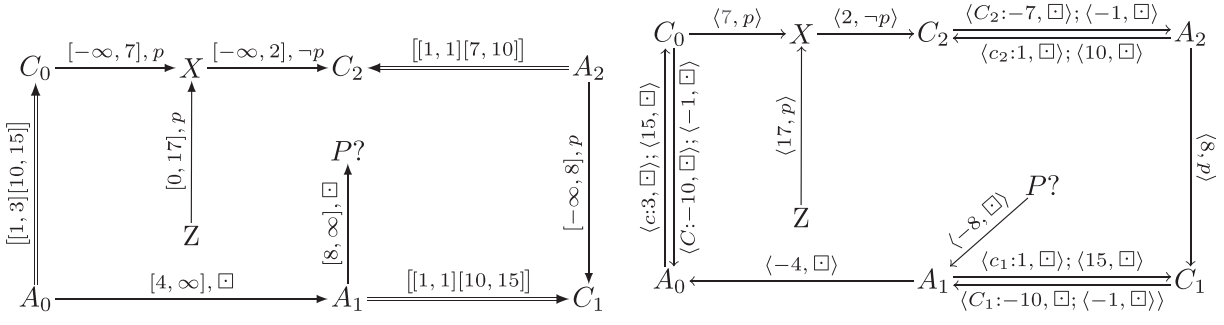
As introduced at the start of the section, the algorithm applies the rules and checks whether newly added edges determine a *negative semi-reducible cycle* in the graph [36]. A cycle in a distance graph is called *semi-reducible* when it is a path made of ordinary or conjunct-upper-case edges only, and it starts and ends at the same time-point [36]. A *negative* semi-reducible cycle is a semi-reducible cycle that has the sum of all edge weights (considered without their labels) negative. A negative semi-reducible cycle represents the set of constraints (the ones making the cycle) that cannot be all satisfied in at least one possible execution of the network. Therefore, when a negative semi-reducible cycle is found, the algorithm returns that the network is not DC. Fig. 3f depicts a negative semi-reducible cycle on Z determined by applying rG_1. The rule applied on $Z \xleftarrow{\langle V_E:-9,\neg p\rangle} V_E \xleftarrow{\langle 8,\neg p\rangle} Z$, determines the negative loop in Z with value $\langle V_E:-1,\neg p\rangle$. Such a value means that in scenarios $\neg p$, when contingent time-point V_E occurs at its upper-guard time (that must be always guaranteed), a constraint is violated. The negative semi-reducible cycle is already present in the original graph (i.e., Z, V_E, V_S, Z), but only applying the rules it becomes explicit.

Example 1. To better illustrate the importance of all propagation rules, let us consider the streamlined FTNU depicted in Fig. 4a and its distance graph depicted in Fig. 4b. The FTNU contains three contingent links that have to occur satisfying some temporal constraints that depend on the condition p. After executing the DC-checking algorithm that considers only rules in Table 1, the graph is updated with new edges and values as partially shown in Fig. 5a, where we reported the most significant new values/edges. The new graph contains the negative semi-reducible cycle $Z - X - Z$. Therefore, the corresponding FTNU cannot be DC. To verify the uncontrollability, let us consider an execution where contingent time-points C_1 and C_2 occur 10 and 7 time-units after their activation time-point, respectively (10 and 7 are the upper guards of the two guarded links), and contingent time-point C_0 occurs 3 time-units after its activation time-point (3 is the lower guard of the guarded link). In such an execution, any possible value of X violates a constraint at least. This unsuccessful execution occurs for any possible schedule of other controllable time-points, i.e., A_0, A_1, A_2, and $P?$. For example, considering the schedule $A_0 = 0, C_0 = 3, A_1 = 4, A_2 = 6, P? = 12, C_2 = 13$, and $C_1 = 14$ (the values of contingent time-points C_i are decided by the environment), X should be executed within 10 for satisfying the constraint $C_0 \xrightarrow{\langle 7,\, p\rangle} X$, but in this way the constraint $X \xrightarrow{\langle 2,\, \neg p\rangle} C_2$ would be violated in case $p = \bot$ at time 12. If X is executed at 12, then it violates the constraint $C_0 \xrightarrow{\langle 7,\, p\rangle} X$ in case that $p = \top$. Moreover, delaying the execution of non-contingent time-points does not solve the issue. In this case, for example, delaying the execution of A_0 would push forward the execution of A_1 and, then, of the other ones.

5.1. Deriving the right ranges

The algorithm proposed in [7] is sound and correct but it is easy to verify that it lacks the ability to determine the right external ranges of guarded links. The right external ranges are not necessary for checking the DC property, but without them, a DC network can be always successfully executed considering *only the cores* of guarded links. In other words, it is not possible to consider the possible flexibility represented by the guarded links. Therefore, a more meaningful approach consists of having an algorithm that checks the DC property and, in the case of a DC instance, finds the right guarded ranges to execute the network exploiting the flexibility of such guarded links.

For example, let us consider the FTNU of Fig. 4b. If we replace the constraint $C_0 \xrightarrow{\langle 7,\, p\rangle} X$ with $C_0 \xrightarrow{\langle 8,\, p\rangle} X$, it is possible to verify that the network becomes DC if all guarded ranges are slightly restricted. Indeed, in the execution shown in Example 1 where the guarded links are restricted to their core (C_0 occurs at its minimum while C_1 and C_2 occur at their maximum), X can be executed at 11 satisfying both constraints $C_0 \xrightarrow{\langle 8,\, p\rangle} X$ and $X \xrightarrow{\langle 2,\, \neg p\rangle} C_2$ before knowing the value of p. Fig. 5b depicts the distance graph after the successful DC checking of the modified network. The algorithm, using only rules in Table 1, determines correctly that the new network is DC but does not determine the right guarded ranges for the guarded links and, therefore, the



(a) Non-DC (streamlined) FTNU (b) Distance Graph of (a)

Fig. 4. Non-DC FTNU and corresponding distance graph.

network admits unsuccessful executions. For example, a possible wrong execution consists of executing A_0 at 0 (i.e., adding $Z \xrightarrow{\langle 0,\ \boxdot\rangle} A_0$). Then, a new DC check determines that the guarded range of $A_0 - C_0$ remains $[[1,3][10,15]]$ and, therefore, it is possible to assign the maximal range, $[[1,1][15,15]]$, as the range for the execution of the contingent time-point C_0. The propagation of such a new range determines a negative cycle, i.e., the new range makes the network not DC!

To verify the DC of an FTNU and derive the right guarded ranges, it is necessary to consider additional rules. In these rules, we have to consider a new kind of edge that we name *conjunct-lower-case edge*. Such a kind of edge stems from the lower-case edges, originally proposed by Morris [35]. We extend the original meaning and use of such constraints to a more general case. In the original proposal, a lower-case edge was used to forbid the increase of the lower bound between the activation point and the contingent one of a contingent link. We extend such meaning according to two different perspectives:

1. a lower-case edge of a contingent link is extended to the origin Z, i.e., it represents the upper bound to the occurrence of a contingent time-point C in the case the environment decides to set C using the lower guard value of the guarded link. Then, such an upper bound can be forward propagated to other time-points. Such a new kind of constraint assumes an important meaning when it is propagated to the activation time-points of other guarded links.
2. as we have to consider possibly multiple interdependent guarded links, the lower-case label has been extended from a single lower-case letter to a conjunction of lower-case letters.

Intuitively, a conjunct-lower-case edge from Z to an activation time-point A, $Z \xrightarrow{\langle \urcorner : v,\ \ell\rangle} A$, specifies that A must be executed before or at v in scenario ℓ when each contingent time-point D_i referenced in the conjunct-lower-case label $\urcorner$ ($d_i \in \urcorner$) occurs at its *lower guard time*, i.e., at time $A_i + x'_i$, where A_i and x'_i is in $(A_i,\ [[x_i,\ x'_i]\ [y'_i, y_i]],\ D_i)$. The proposed rules allow the meaningful propagation and integration of such kind of constraints with the other ones, to check the DC property, and to derive the right bounds for the guarded links.

The new rules are of two types.

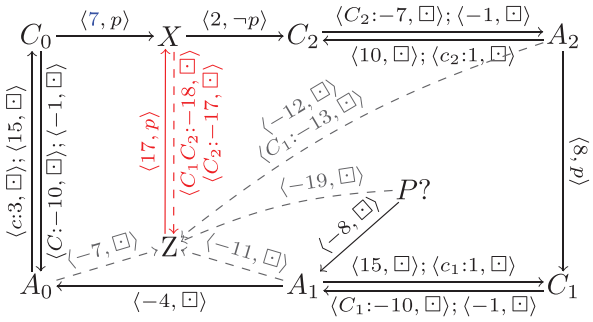
The first one consists of rules rG_5, rG_6, and rG_7 (cf. Table 2). Such rules determine, for each time-point, the conjunct-lower-case edge from Z in a similar way as done for conjunct-upper-case ones by rules rG_1–rG_3 in Table 1.

The second type consists of rules rG_8 and rG_9 that, for each guarded link, determine the right external range (i.e., bounds x and y in guarded links like $(A_i,\ [[x, x']\ [y', y]],\ C_i)$ combining the values of the conjunct-lower-case edges and conjunct-upper-case ones related to the activation time-point and the contingent one of the considered guarded link, respectively.

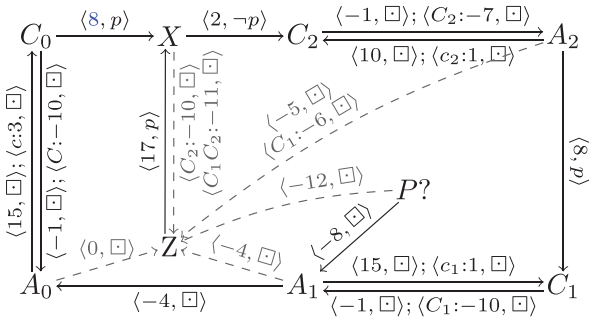
As an example for the rule rG_9, let us consider the conjunct-lower-case edge $Z \xrightarrow{\langle de:10,\ p\rangle} A$ and the conjunct-upper-case edge $Z \xleftarrow{\langle F:-15,\ p\rangle} C$. The edge $Z \xrightarrow{\langle de:10,\ p\rangle} A$ represents an upper bound for executing A in scenarios where p is $\top$ and when contingent time-points D and E occur at their lower guarded time. The edge $Z \xleftarrow{\langle F:-15,\ p\rangle} C$ represents a lower bound for executing C in scenarios where p is $\top$ and when contingent time-point F occurs at its upper guarded time. Let us suppose that the guarded link between A and C is $(A,\ [[1,7]\ [10,20]],\ C)$. In an execution in which 1) proposition p assumes $\top$ value, 2) contingent time-points D and E occurs at their lower guarded time while 3) contingent time-point F occurs at its upper guarded time, A must be scheduled at an instant t such that $t \leqslant 10$ while C must occur at a time $t' \geqslant 15$. In $(A,\ [[1,7]\ [10,\ 20]],\ C)$ the lower bound is 1 that can be increased to 7 at maximum. If we let the lower bound to 1, in the case A is executed at 10, the environment can decide to execute C at 11 violating the constraint $Z \xleftarrow{\langle F:-15,\ p\rangle} C$ in the case F does not occur before or at 11 (and when p is $\top$). To avoid such a possibility, it is sufficient to set the lower external bound of the guarded link to $10 - 15 = -5$, as done by the rule setting the edge $A \xleftarrow{\langle -5,\ \boxdot\rangle} C$, to have the guarantee that any allowed duration for the updated guarded link $(A,\ [[5,7]\ [10,20]],\ C)$ does not violate the constraint $Z \xleftarrow{\langle F:-15,\ p\rangle} C$.

An alternative interpretation of conjunct-lower-case/upper-case edges. Both conjunct-lower-case edges and conjunct-upper-case ones can be viewed as special upper/lower bounds edges for executing time-points. Indeed, given a time-point A, the conjunct-lower-case edge $Z \xrightarrow{\langle \urcorner : v,\ \ell\rangle} A$ represents the *lowest upper bound* for A that can be set between Z and time-point A and has to be dynamically verified when all the contingent time-points referenced in $\urcorner$ are executed at their lower guarded time. If a requirement constraint $Z \xleftarrow{\langle -w,\ \ell\rangle} A$ with $w > v$ is added to the FTNU, then the network cannot be DC because this new constraint is not satisfied at least in a possible execution of the network, i.e., in an execution where contingent time-points in $\urcorner$ occurs at their lower guarded time in scenario ℓ. Such an execution is possible because all contingent time-points can always occur at their lower guarded time.

Analogously, given a time-point A, the conjunct-upper-case edge $Z \xleftarrow{\langle \aleph : -v,\ell\rangle} A$ represents the *greatest lower bound* of A that can be set between Z and A and has to be dynamically verified when all the involved contingent time-points are executed at their *upper guarded time* (it is analogous to the lower guarded time). If an ordinary constraint $Z \xrightarrow{\langle w,\ell\rangle} A$ with $w < v$ is added to the FTNU, then the network cannot be DC.



(a) Most important generated edges (dashed) in distance graph of Figure 4b. In red, a negative semi-reducible cycle.



(b) DC distance graph of fixed Figure 4b where $(X - C_0 \leq 7, p)$ is replaced by$(X - C_0 \leq 8, p)$

Fig. 5. An example of a non dynamic-controllable FTNU instance, (a), and of a dynamic controllable one, (b).

5.2. Soundness

In this section, we show the soundness of rules rG_5–rG_7 and rules rG_8–rG_9 (cf. Table 2).

The proofs for rules rG_5–rG_7 are similar to the proofs given for rules rG_1–rG_3 in [29]. Therefore, we preliminarily recall some necessary definitions given in [29] adjusted for FTNUs.

We already introduced that a *scenario* s is an assignment of a truth value for each proposition in $\mathcal{P}$; a *partial scenario* is a scenario limited to a subset of the propositions.

A *situation* ω enumerates a value (*duration*) for each guarded link, while a *partial situation* enumerates a duration for some guarded links. We represent the duration of the i-th guarded link in situation ω as $\omega[i]$.

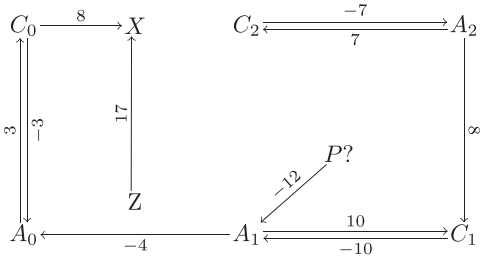
As in [27–29], by *drama*, we denote a scenario & situation pair like (s, ω) that specifies all the environment choices in the execution of the network.

Given a drama (s, ω) relative to an FTNU instance $\mathcal{S}$, the "application" of the drama (s, ω) to $\mathcal{S}$ determines a projection of $\mathcal{S}$, $Prj(\mathcal{S}, s, \omega)$, that results to be a Simple Temporal Network (STN [9]). The application consists of:

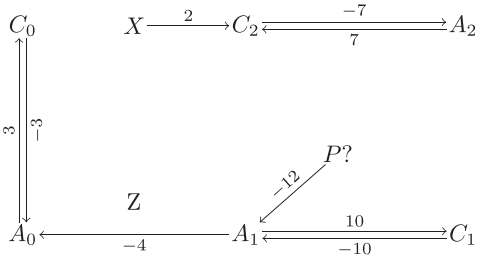
- removing all the constraints whose labels are false under s;
- removing all other labels (that must be true) preserving the values;
- for each guarded link in the original FTNU, replacing all four edges representing it in the distance graph $\mathcal{S}$ by two ordinary edges representing the duration specified by ω, i.e., $A_i \xrightarrow{\omega[i]} C_i$, and $A_i \xleftarrow{-\omega[i]} C_i$.

For example, Fig. 6 depicts two projections of DC FTNU associated with the distance graph in Fig. 5b: one on the drama $(s_1, \omega_1) = (\{p = \top\}, \{\omega[0] = 3, \omega[1] = 10, \omega[2] = 7\})$ and the other one on the drama $(s_2, \omega_1) = (\{p = \bot\}, \{\omega[0] = 3, \omega[1] = 10, \omega[2] = 7\})$.

A *schedule* is a function, $\psi : \mathcal{T} \to \mathbb{R}$ that associates a real value, assumed as the *execution* time, to each time-point in $\mathcal{T}$. For historical reasons, the value determined by a schedule ψ for a time-point $X \in \mathcal{T}$ is denoted as $[\psi]_X$. We denote by Ψ the set of schedules for $\mathcal{T}$.



(a) FTNU projection onto the drama $(s_1, \omega_1) = (\{p = \top\}, \{\omega_1[0] = 3, \omega_1[1] = 10, \omega_1[2] = 7\})$



(b) FTNU projection onto the drama $(s_2, \omega_1) = (\{p = \bot\}, \{\omega_1[0] = 3, \omega_1[1] = 10, \omega_1[2] = 7\})$

Fig. 6. Two different projections of FTNU depicted in Fig. 5b.

Let $\mathcal{OT} = \{P_1?, \ldots, P_k?\}$ be a set of observation time-points. A permutation π over $(1, 2, \ldots, k)$ determines an order for the considered observation time-points. For any $P_i \in \mathcal{OT}$, let $\pi(P_i) \in \{1, 2, \ldots, k\}$ be the position of P_i in the permutation π; and let Π_k be the set of all permutations over $(1, 2, \ldots, k)$. This order is necessary to state in which order to propagate the value of some propositions when two or more observation time-points must be executed at the same time. Let $\mathcal{I}$ be the set of all possible scenarios, and Ω the set of all possible situations.

A *π-execution strategy* for an FTNU $\mathcal{S}$ instance with k observation time-points is a function, $\sigma : (\mathcal{I} \times \Omega) \to (\Psi \times \Pi_k)$, such that for each drama $r = (s, \omega) \in \mathcal{I} \times \Omega$, $\sigma(s, \omega)$ is a pair (ψ_r, π_r) such that $\psi_r : \mathcal{T} \to \mathbb{R}$ is a schedule, and $\pi_r \in \Pi_k$ determines an order of dependency among the observation time-points.

To make easier the specification of the following definitions, the definition of π_r is extended to other types of time-points in the following way: $\pi_r(C_i) = 0$ for each contingent time-point C_i, and $\pi_r(Y) = \infty$ for each non-contingent, non-observation time-point Y. In other words, while any contingent time-point C_i is not depending on any other time-point (therefore, $\pi_r(C_i) = 0$), any non-observation and non-contingent time-point Y has to be considered after all the observation time-points executed at the same time (therefore, $\pi_r(X) = \infty$).

For example, limited to the dramas $\mathbf{r_t} = (s_1, \omega_1) = (\{p = \top\}, \{\omega_1[0] = 3, \omega_1[1] = 10, \omega_1[2] = 7\})$ and $\mathbf{r_f} = (s_2, \omega_1) = (\{p = \bot\}, \{\omega_1[0] = 3, \omega_1[1] = 10, \omega_1[2] = 7\})$ that induce the projections depicted in Fig. 6, a π-execution strategy $\boldsymbol{\sigma}'$ could define the two pairs $\mathbf{ST_{r_t}} = (\psi_{r_t}, \pi_{r_t})$ and $\mathbf{ST_{r_f}} = (\psi_{r_f}, \pi_{r_f})$:

$\mathbf{ST}_{\mathbf{r_t}} = (\psi_{r_t}, \pi_{r_t}) =$

time-points	ψ_{r_t}
Z, A_0, X	0
A_1	4
A_2	6
$P?$	12

time-points	π_{r_t}
Z, A_0, A_1, A_2, X	∞
C_0, C_1, C_2	0
$P?$	1

$\mathbf{ST}_{\mathbf{r_f}} = (\psi_{r_f}, \pi_{r_f}) =$

time-points	ψ_{r_f}
Z, A_0, A_2	0
A_1	4
A_2	6
X	11
$P?$	12

time-points	π_{r_f}
Z, A_0, A_1, A_2, X	∞
C_0, C_1, C_2	0
$P?$	1

Given a strategy σ, we define it *viable* if, for each drama $r = (s, \omega)$, the schedule ψ_r is a solution to the STN given by the projection $Prj(\mathcal{S}, s, \omega)$.

To characterize the *dynamic* aspect of the controllability property, it is necessary to introduce the concept of *history* that collects the values of occurred contingent time-points, and the values of the assigned propositions before a given instant t.

Let $\mathcal{S}$ be an FTNU instance with k observation time-points; $r = (s, \omega)$ a drama; σ a π-execution strategy for $\mathcal{S}$; $(\psi_r, \pi_r) = \sigma(s, \omega)$; $d \in \{1, 2, \ldots, k; \infty\}$; and $t \in \mathbb{R}$.

The *π-history* of (t, d) for the drama (s, ω) and strategy σ is defined as $\mathcal{H}(t, d, s, \omega, \sigma) = (\mathcal{H}_s, \mathcal{H}_\omega)$ where:

$$\mathcal{H}_s = \{(p, s(p)) | P? \in \mathcal{OT},\ [\psi_r]_{P?} \leqslant t,\ \text{and}\ \pi_r(P?) < d\};$$
$$H_\omega = \{(A, C, [\psi_r]_C - [\psi_r]_A) \mid (A,\ [[x, x']\ [y', y]],\ C) \in \mathcal{G},\ \text{and}\ [\psi_r]_C \leqslant t\}.$$

$\mathcal{H}_s$ is the set of the truth values of all propositions p observed *before* time t in the schedule ψ_r, as well as those observed *at* time t if $P?$ is ordered *before* position d by the permutation π_r. $\mathcal{H}_\omega$ specifies the durations of all guarded links such that the associated contingent time-point occurred *at or before* time t in the schedule ψ_r.

As an example, considering the two above dramas $\mathbf{r_t}$ and $\mathbf{r_f}$ and the instant $t = 11$, the π-history of $(11, \infty)$ for the drama $\mathbf{r_t}, \mathcal{H}(11, \infty, s_1, \omega_1, \boldsymbol{\sigma}')$, is equal to the history $\mathcal{H}(11, \infty, s_2, \omega_1, \boldsymbol{\sigma}')$, relative to the drama $\mathbf{r_f}$ since

$$\mathcal{H}_{s_1} = \mathcal{H}_{s_2} = \{\};$$
$$\mathcal{H}_{\omega_1} = \{\omega_1[0] = 3, \omega_1[1] = 10, \omega_1[2] = 7\}.$$

The definition of dynamic execution is based on the fact that an execution strategy must consider only the history to schedule non-contingent time-points.

Definition 1 (*π-Dynamic Execution Strategy*). A π-execution strategy, σ, for an FTNU $\mathcal{S}$, is called *π-dynamic* if for every pair of dramas, (s_1, ω_1) and (s_2, ω_2), and every *non-contingent* time-point X:

let:	$(\psi_1, \pi_1) = \sigma(s_1, \omega_1)$ and $(\psi_2, \pi_2) = \sigma(s_2, \omega_2)$,
let:	$t = [\psi_1]_X$, and $d = \pi_1(X) \in \{1, 2, \ldots, \lvert\mathcal{OT}\rvert + \lvert\mathcal{G}\rvert; \infty\}$.
if:	$\mathcal{H}(t, d, s_1, \omega_1, \sigma) = \mathcal{H}(t, d, s_2, \omega_2, \sigma)$
then:	$[\psi_2]_X = t$ and $\pi_2(X) = d$.

Hence, if in the drama (s_1, ω_1) the schedule σ executes Y at t (and position d), and its history is the same to the one relative to the drama (s_2, ω_2) at time t, then σ must also execute Y at t (and position d).

As an example, the above strategy $\boldsymbol{\sigma}'$ is a viable strategy but not dynamic because at instant 11 the two considered dramas $\mathbf{r_t}$ and $\mathbf{r_f}$ have the same π-history but in drama $\mathbf{r_t}$ $[\psi_{r_t}]_X = 0$ while in drama $\mathbf{r_f}$ $[\psi_{r_f}]_X = 11$. To be dynamic, $\boldsymbol{\sigma}'$ should define a schedule that sets X at 11 in the drama $\mathbf{r_t}$, i.e., $[\psi_{r_t}]_X = 11$.

Now, the most important definitions for the following soundness proofs.

Definition 2 (*Satisfy an original constraint*). . Given an FTNU $\mathcal{S}$ and a labeled constraint $(Y - X \leqslant \delta, \ell)$ where $\ell \in \mathcal{P}^*$, a π-execution strategy σ *satisfies the labeled constraint* $(Y - X \leqslant \delta, \ell)$ if, for each drama (s_j, ω_j), it holds that

- either $s_j(\ell) = \bot$
- or $[\psi]_Y - [\psi]_X \leqslant \delta$,

where $(\psi, \pi) = \sigma(s_j, \omega_j)$.

As regards guarded links, σ *satisfies the guarded link* $(A_i,\ [[x_i, x_i']\ [y_i', y_i]],\ C_i)$ if, for each drama (s_j, ω_j), it sets $[\psi]_{A_i}$ such that $[\psi]_{C_i} - [\psi]_{A_i} = \omega_j[i]$. In such a case, we also say that σ satisfies the lower-case and upper-case edges associated with that guarded link.

From the previous definitions, we can characterize a *viable* π-dynamic execution strategy in a simpler way. Given an FTNU $\mathcal{S}$, a *viable* π-dynamic execution strategy σ must satisfy all the original labeled-constraints in $\mathcal{S}$ (before any constraint propagation) and all the original lower- and upper-case edges in $\mathcal{S}$.

In general, a constraint-propagation rule is defined *sound* if, whenever a viable and dynamic σ satisfies the pre-existing edge(s) in that rule, σ must also satisfy the edge generated by that rule. Now, the rules rG_5–rG_7 in Table 2 only generate edges leaving from Z, which represent upper-bound constraints, having possibly multiple lower-case (LC) letters. Therefore, to prove the soundness of such rules, it is necessary to define how a viable and dynamic σ satisfies an upper-bound constraint containing conjunct-lower-case labels.

Definition 3 (*Satisfy an Upper-Bound Constraint*). Given an FTNU $\mathcal{S}$ and a labeled upper-bound constraint $Z \xrightarrow{\langle \urcorner : v,\ \ell \rangle} A$, where $\ell \in \mathcal{P}^*$, and $\urcorner$ is a conjunction of lower-case contingent names, a π-execution strategy σ *satisfies the upper-bound constraint* $Z \xrightarrow{\langle \urcorner : v, \ell \rangle} A$ if for each drama (s, ω), *any* of the following conditions holds, where $(\psi, \pi) = \sigma(s, \omega)$:

1. $[\psi]_A \leqslant v$;
2. for some $C_i \in \urcorner$, where $(A_i,\ [[x_i, x_i']\ [y_i', y_i]],\ C_i) \in \mathcal{G}$, $\omega[i] > x_i'$ (i.e., the i^{th} guarded link does *not* take on its lower guard duration);
3. for some $p \in \ell$, $s(p) = \bot$;
4. for some $\neg p \in \ell$, $s(p) = \top$.

Let us consider a labeled upper-bound constraint defined at design time: $Z \xrightarrow{\langle v, \ell \rangle} Y$. Since such a constraint is present in the network before any propagation, $\ell \in \mathcal{P}^*$. For such a constraint, clause 2. in Definition 3 does not apply. Therefore, satisfaction reduces to: $[\psi]_Y \leqslant v$ or $s(\ell) = \bot$. Thus, Definition 3 reduces to Definition 2 for original labeled-edges that happen to be upper-bound edges.

More generally, it will be useful to note that for any upper-bound edge to Y labeled by $\langle \urcorner : v, \ell \rangle$, where $\ell \in \mathcal{P}^*$, satisfaction (i.e., Definition 3) reduces to:

(i) $[\psi]_Y \leqslant v$;
(ii) for some $C_i \in \urcorner$, where $(A_i,\ [[x_i, x_i']\ [y_i', y_i]],\ C_i) \in \mathcal{G}$, $\omega[i] > x_i'$; or
(iii) $s(\ell) = \bot$.

Now it is possible to formalize the concept of soundness.

Definition 4 (*Soundness*). A constraint-propagation rule is *sound* if whenever a *viable* and π-*dynamic* execution strategy σ satisfies the rule's pre-existing (parent) edges, it also satisfies the rule's generated (child) edge.

The soundness of rules in Table 1 was proved in previous articles.

Lemma 1 (*[7,29].*). *Rules rG_1–rM_2 in Table 1 are sound.*

Therefore, here we show the soundness of rules rG_5–rG_9 in Table 2.

Lemma 2. *Rules rG_5–rG_9 from Table 2 are sound.*

Proof. In the following, we present the proof for rules rG_6–rG_8; the proofs for rG_5 and rG_9 can be easily derived from the ones of rG_6–rG_8.

Let σ be a viable and π-dynamic strategy that satisfies the parent edges in the considered rule.

Rule rG_6.

Let (s, ω) be any drama in which all $C_i \in \neg$ occur exactly at the instant corresponding to their lower guard (i.e., $\omega[i] = x'_i$) and such that $s(\beta) = \top$; and let $(\psi, \pi) = \sigma(s, \omega)$. Since σ satisfies the parent constraint from Z to A, it follows that $[\psi]_A \leqslant v$. Next, since C takes on its minimum duration, then $[\psi]_C = [\psi]_A + x' \leqslant v + x'$. Thus, σ satisfies the generated edge from Z to C.

Rule rG_7.

Let us assume that σ does *not* satisfy the generated edge from Z to A in rule rG_7. By Definition 3, there is some drama (s, ω) such that *all* the following conditions hold:

($\neg$ i) $[\psi]_A > v - y'$;
($\neg$ ii) $\forall D_i \in \neg$, where $(E_i,\ [[x_i, x'_i]\ [y'_i, y_i]],\ D_i) \in \mathcal{G},\ \omega[i] = x'_i$; and
($\neg$ iii) $s(\beta) = \top$.

Condition ($\neg$ iii) holds because it holds in the parent constraint. Then, since σ satisfies the parent constraint from Z to C by assumption, ($\neg$ ii) holds and $[\psi]_C \leqslant v$. In ω the contingent link AC can have a value $\leqslant y' \leqslant y$. Let ω' be the same as ω except that the contingent link AC takes on its *upper guard* value y', i.e., $[\psi']_C - [\psi']_A = y'$; and let $(\psi', \pi') = \sigma(s, \omega')$. Since $C \notin \neg$, ($\neg$ ii) also holds for ω'; thus, $[\psi']_C \leqslant v$ must hold. And, since the only difference between (s, ω) and (s, ω') is the duration of the contingent link AC, the first difference between ψ and ψ' must occur when C executes, which happens *after* A executes. Thus, $[\psi]_A = [\psi']_A = [\psi']_C - y' \leqslant v - y'$, contradicting ($\neg$ i).

Rule rG_8.

Let (s, ω) be any drama for which:

- all contingent time-points $D_i \in \aleph$ occur exactly at the instant corresponding to their upper guard (i.e., $\omega[i] = y'_i$),
- all contingent time-points $E_j \in \neg$ occur exactly at the instant corresponding to their lower guard (i.e., $\omega[j] = x'_j$),
- $s(\alpha) = \top$,
- and $s(\beta) = \top$.

Such drama is possible because $\aleph$ and $\neg$ have no common contingent time-point. Let $(\psi, \pi) = \sigma(s, \omega)$. Since σ satisfies the parent constraints, it follows that $[\psi]_A \geqslant v'$ and $[\psi]_C \leqslant v$. Moreover, since C is a contingent time-point, the relation $[\psi]_C \leqslant v$ means that A is surely executed before $v - y'$ in the partial situation $\neg$ and partial scenario β. Otherwise, C cannot be executed in any instant in $[x', y']$ after A in the partial situation $\neg$, i.e., the parent constraints belong to a not-DC instance. Therefore, $[\psi]_C - [\psi]_A$ must be $\leqslant v - v'$. Note that if drama (s, ω) did not allow the execution of C at any instant $t \in [x', y']$ after A, then the application of rules rG_1–rG_3 would be sufficient to find a negative semi-reducible cycle in such a drama [29]. Thus, σ satisfies the generated constraint from A to C.

Theorem 1. *Rules rG_1–rM_2 from Table 1 and rules rG_5–rG_9 from Table 2 are sound.*

Proof. Lemma 1 shows the soundness of rules rG_1–rM_2 from Table 1.
Lemma 2 shows the soundness of rules rG_5–rG_9 from Table 2. □

5.3. Completeness

The completeness of the new algorithm we proposed here, Algorithm 1 may be given extending the completeness proof for CSTNU presented in [29]. We recall, as a lemma, a previous result useful for proving the theorem.

Let us consider an FTNU instance $\mathcal{S}$ and let $\mathcal{S}^*$ be the FTNU distance-graph obtained by successfully executing Algorithm 1 with input $\mathcal{S}$.

Let $\mathcal{S}^*_{CSTN}$ be the graph obtained from $\mathcal{S}^*$ by 1) deleting all conjunct-lower-case edges and lower-case one and 2) removing all conjunct-upper-case labels from the labeled values. In other words, all conjunct-upper-case edges become ordinary

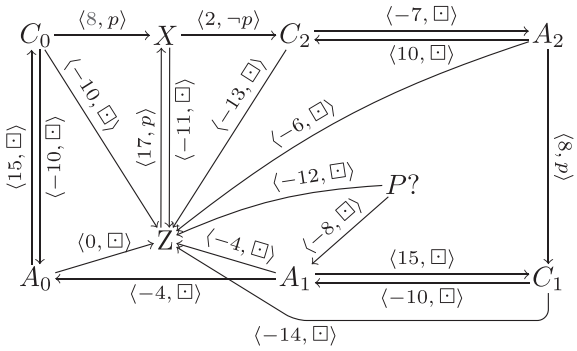


Fig. 7. An example of an AllMax Conditional Simple Temporal Network where only the most important edges are represented. It is relative to the DC distance graph depicted in Fig. 5b.

edges, replacing possible already present ordinary edges. $\mathcal{S}^*_{CSTN}$ is called *AllMax* Conditional Simple Temporal Network (CSTN) because it preserves observation nodes (conditions) while does not contain all uncertainties (guarded links). $\mathcal{S}^*_{CSTN}$ is useful for showing that the *early execution strategy* (detailed in proof of Theorem 2) can always determine which nodes to execute with respect to the current scenario [37, Spreading Lemma].

As an example, Fig. 7 depicts the AllMax Conditional Simple Temporal Network relative to the DC distance graph depicted in Fig. 5b.

In the completeness proof, we consider both $\mathcal{S}^*$ and $\mathcal{S}^*_{CSTN}$. Considering $\mathcal{S}^*_{CSTN}$ does not require any extra computation as shown by the following lemma.

Lemma 3. *Given an FTNU instance $\mathcal{S}$ that is successfully checked by Algorithm 1, the corresponding AllMax CSTN $\mathcal{S}^*_{CSTN}$ is fully propagated and DC.*

Proof. Ignoring conjunctive upper-/lower-case labels in rules rG_1–rG_9 from Tables 1 and 2 for FTNUs reduces them to the three fundamentals rules used for checking CSTNs [37]. Since $\mathcal{S}^*$ is fully propagated without negative cycles having consistent labels by hypothesis, then $\mathcal{S}^*_{CSTN}$ must be fully propagated without negative cycles and, hence, DC. □

The **execution** of a time-point Y at time t in an FTNU is represented by adding two edges $\mathrm{Z} \xrightarrow{\langle t,\mathrm{CPS}\rangle} Y$ $\mathrm{Z} \xleftarrow{\langle -t,\mathrm{CPS}\rangle} Y$ in $\mathcal{S}$ and in $\mathcal{S}^*$, where CPS stands for *current partial scenario*, i.e., the label containing all the observed literals associated to the already executed observation time-points.

Now, it is possible to introduce the completeness theorem.

Theorem 2. *The set of rules rG_1–rG_9 from Table 1 and Table 2 is complete for DC checking of FTNUs.*

Proof. Given an FTNU $\mathcal{S}$, let $\mathcal{S}^*$ be the FTNU distance-graph obtained by successfully executing Algorithm 1 with input $\mathcal{S}$; then, let $\mathcal{S}^*_{CSTN}$ be the corresponding *AllMax* CSTN that is fully propagated and DC (Lemma 3).

For sake of brevity, hereinafter, every time we add a new edge into $\mathcal{S}^*$, we implicitly consider that the same edge is also added to $\mathcal{S}^*_{CSTN}$.

Let CPS be the *current partial scenario*. At the start of execution CPS = ⊡.

Let $\mathcal{T}_u$ be the set of unexecuted time-points. At start $\mathcal{T}_u = \mathcal{T} \setminus \{\mathrm{Z}\}$.

Consider the following *earliest-first strategy* σ for $\mathcal{S}$.

For each $Y \in \mathcal{T}_u$, we determine its *effective lower bound* as $\mathrm{ELB}(Y) = \max\{v \mid \exists\, \mathrm{Z} \xleftarrow{\langle -v,\alpha\rangle} Y \text{ in } \mathcal{S}^*_{CSTN}, appl(\alpha, \mathrm{CPS})\}$, where $appl(\alpha, \mathrm{CPS})$ holds if α is *applicable* given the current partial scenario CPS. In particular, $appl(\alpha, \mathrm{CPS})$ holds if each proposition p that appears in *both* α and CPS appears as the same literal in both [24].

Let (Λ, χ) be the first execution decision: "*if nothing happens before time Λ, then execute the time-points in χ*", where $\Lambda = \min\{\mathrm{ELB}(Y) \mid Y \in \mathcal{T}_u\}$; and $\chi = \{Y \in \mathcal{T}_u \mid \mathrm{ELB}(Y) = \Lambda\}$. In [37], the authors showed that such a strategy can execute any DC CSTNU instance (i.e., it is sufficient to show the completeness of the DC checking algorithm for CSTNUs). Here, we show that such a strategy can execute any DC FTNU instance.

Given the first execution decision, there are two possibilities:

Case 1: No contingent time-point executes before time Λ

We call a guarded link *active* when its activation time-point is already executed but its contingent time-point has not yet occurred. For each active $(A_i,\ [[x^*_i, x^*_i]\ [y^*_i, y^*_i]],\ C_i)$, raise the contingent time-point lower bound to $\mathrm{Z} \xleftarrow{\langle \Lambda - a_i,\mathrm{CPS}\rangle} C_i$, where $a_i = [\sigma(s)]_{A_i}$ in $\mathcal{S}^*$. This cannot introduce any new constraints because $\Lambda - a_i \in [x^*_i, y^*_i]$; thus, $\mathcal{S}^*$ is still DC [29]. Since no ELB

values have changed, (Λ, χ) is the earliest-first decision also for the CSTN $\mathcal{S}^*_{CSTN}$. Then, time-points in χ are executed. The execution of time-points in χ requires particular attention only when they are activation time-points because the execution of each of such time-points must update the external bounds of the relative guarded link.

Therefore, the execution of time-points in χ consists of the following steps:

1. for each observation time-point $X_i \in \chi$ (any order among concurrent observation time-points is allowed [37]):
   (a) **execute** X_i at time Λ;
   (b) update the CPS to include the value of proposition $O^{-1}(X_i)$;
2. If CPS changed:
   (a) delete any labeled values that are inconsistent with the updated CPS in $\mathcal{S}^*$.
3. for each activation time-point $A_i \in \chi$ (let us denote the original guarded link associate to A_i as $(A_i,\ [[x_i^*, x_i'] [y_i', y_i^*]],\ C_i)$:
   (a) **execute** A_i at time Λ;
   (b) propagate such execution (i.e., the new two added edges) in $\mathcal{S}^*$ using rules in Table 1 and Table 2. Such propagation can modify the external bounds x_i^* and y_i^* to x_i^{**} and y_i^{**}, respectively.
   (c) update the bounds of edges $A_i \xrightarrow{\langle c_i : x_i',\ \boxdot\rangle;\langle y_i^*,\ \boxdot\rangle} C_i$ and $A_i \xleftarrow{\langle C_i : -y_i',\ \boxdot\rangle;\langle -x_i^*,\ \boxdot\rangle} C_i$ in $\mathcal{S}^*$ (they represent the guarded link $(A_i,\ [[x_i^*, x_i'] [y_i', y_i^*]],\ C_i)$ to $A_i \xrightarrow{\langle c_i : x_i^{**},\ \boxdot\rangle;\langle y_i^{**},\ \boxdot\rangle} C_i$ and $A_i \xleftarrow{\langle C_i : -y_i^{**},\ \boxdot\rangle;\langle -x_i^{**},\ \boxdot\rangle} C_i$; this update makes the guarded link a contingent one with the largest allowed range;
4. for each other time-point $Y_i \in \chi$, **execute** Y_i at time Λ;
5. propagate such executions using rules in Table 1 and Table 2, i.e., update labeled values in the original $\mathcal{S}^*$ updated with the new constraints representing executions and new ranges for activated guarded links.
6. remove any executed time-points from $\mathcal{T}_u$;

It is necessary to show that inserting the constraints related to the execution of time-points (made in steps 1.(a), 3.(a) and 4.) and the update of constraints in step 3.(c) cannot introduce any negative cycle into any relevant STN projection.

Since for non-activation time-points the proof is very similar to the one presented in [37], here we consider only the case relative to activation time-points. In particular, we prove that, after each execution of an activation time-point, the modification of the bounds of the edges corresponding to its guarded link cannot introduce any negative cycle into any relevant STN projection.

Suppose a negative cycle with consistent label is discovered in the modified $\mathcal{S}^*$ after a guarded link $(A_i, [[x^*, x'] [y', y^*]],\ C_i)$ has been updated to $(A_i,\ [[x^*, x^*] [y^*, y^*]],\ C_i)$ (A_i was set to Λ). There are three possibilities. The negative cycle is determined by 1) lowering x' to x^* or 2) by increasing y' to y^* or 3) making both the two updates.

Let us consider the case "lowering x' to x^*".

The edge $A_i \xrightarrow{\langle c_i : x^*,\ \boxdot\rangle} C_i$ must be present in the negative cycle, otherwise it means that a negative cycle was present before the update contradicting the hypothesis that the network is DC.

Fig. 8a depicts the negative cycle induced by lowering the lower guard x' to x^*. We removed all non-important details like edge $Z \xleftarrow{\langle -\Lambda,\ \mathrm{CPS}\rangle} A_i$, and propositional labels. It holds that $\Lambda + x^* - l - w < 0$, i.e., $x^* < l + w - \Lambda$.

Fig. 8b depicts the network just after the execution of A_i. The upper bound to A is Λ. The lower bound for C induced by node E is $-l - w$, determined by rule rG_1. By rule rG_9, $-x^* \leqslant -l - w + \Lambda$, i.e., $x^* \geqslant l + w - \Lambda$, contradicting the hypothesis.

It is easy to verify that a similar proof can be given if E is a contingent time-point and, therefore, the lower bound for C_i can be a conjunct-upper-case lower bound.

Let us consider the case "increasing y' to y^*".

Analogously to case 1), the edge $A_i \xleftarrow{\langle C_i : -y^*,\ \boxdot\rangle} C_i$ must be present in the negative cycle.

Fig. 9a depicts the negative cycle induced by lowering the upper guard y' to $y^* : -\Lambda - y^* + l + w < 0$, i.e., $y^* > l + w - \Lambda$. Fig. 9b depicts the network just after the execution of A_i. The lower bound for A is $-\Lambda$. The upper bound for C induced by node E is $l + w$, determined by rule rG_5. By rule rG_8, $y^* \leqslant l + w - \Lambda$, contradicting the hypothesis.

The case in which both the previous cases occur it is proved combining the two previous analysis since they are complementary.

Case 2: A contingent time-point C_i executes at some time $t \leqslant \Lambda$

Let $\delta = t - [\sigma(s)]_{A_i}$ is the observed duration for the guarded link $(A_i,\ [[x^*, x^*] [y^*, y^*]],\ C_i)$. The semantics associated with the execution of contingent time-points guarantees that $\delta \in [x^*, y^*]$.

Update $\mathcal{S}^*$ in the following way:

1. Update the edges $A_i \xrightarrow{\langle c_i : x_i^*,\ \boxdot\rangle;\langle y_i^*,\ \boxdot\rangle} C_i$ and $A_i \xleftarrow{\langle C_i : -y_i^*,\ \boxdot\rangle;\langle -x_i^*,\ \boxdot\rangle} C_i$ to $A_i \xrightarrow{\langle \delta,\ \boxdot\rangle} C_i$ and $A_i \xleftarrow{\langle -\delta,\ \boxdot\rangle} C_i$; This update makes explicit the occurrence of C_i in the network.

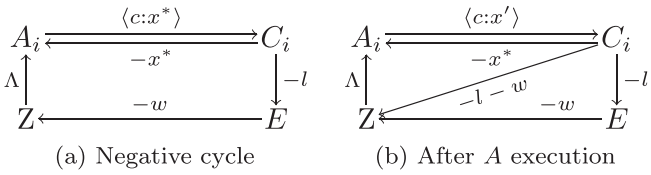


Fig. 8. A negative cycle in the modified (left) and DC original (right) FTNU $\mathcal{S}^*$ before the lower guard update.

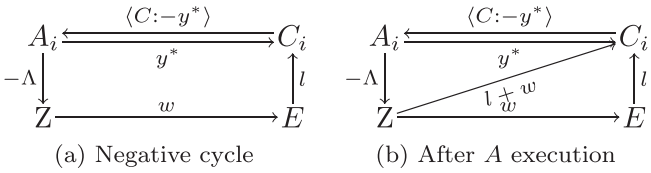


Fig. 9. A negative cycle in the modified (left) and DC original (right) FTNU $\mathcal{S}^*$ before the upper guard update.

2. Delete any conjunct-upper-case value $\langle \aleph{:}v, \beta \rangle$ from $\mathcal{S}^*$ for which $C \in \aleph$.[2]
3. For each $X \in \mathcal{T}_u$, insert a lower-bound constraint, $Z \xleftarrow{\langle t,\ \mathrm{CPS} \rangle} X$. This is useful only for the time-points for which their ELB value could have been determined by a conjunct-upper-case edge having C in $\aleph$. Setting such a lower bound makes explicit that X can be executed at or after t.
4. Propagate the new constraints in $\mathcal{S}^*$ using rules in Table 1 and Table 2.

After such first step, the *earliest-first strategy* σ re-executes the step considering the remaining time-points till there is any time-point unexecuted.

6. An example of FTNU DC-checking and execution

In this section, we present an example of how our FTNU DC checking algorithm can be used 1) to check the Adult Stroke FTNU (see Fig. 1) and 2) to update the guarded links during an execution of the instance.

Fig. 2 depicts the distance graph of the Adult Stroke FTNU presented in Fig. 1. The distance graph is the alternative representation of an FTNU instance that the DC checking algorithm accepts as input.

The Adult Stroke FTNU is DC and, after the execution of the DC checking algorithm, its distance graph is updated by the algorithm adding/updating, for each time-point, the two edges to/from Z. Each of such edges gives information about the possible execution times for the considered time-point with respect to the possible scenario and/or execution time of contingent time-points (wait constraints).

Fig. 10 shows a screenshot of `TNEditor`, an application that allows the management and checking of different constraint networks, as STNs, STNUs, CSTNs, CSTNUs and FTNUs [38, Version 4.3]. As regards FTNUs, the application is able to execute the DC checking by considering only the non-redundant labeled values. In the screenshot, on the left, there is the editor-window where it is possible to edit a network instance; on the right, there is the window where the program shows the instance after the execution of one of the different possible checking algorithms, in this case the FTNU DC-checking algorithm. The selection of which algorithm to apply is done by the user pushing a button on the command bar present on the lower part of the main window of `TNEditor`. In particular, in the right part of Fig. 10, there is the resulting distance-graph obtained by applying the FTNU DC-checking algorithm to the distance graph of the Adult Stroke FTNU instance, represented on the left part.

In the checked Adult Stroke FTNU network, all the external bounds of the guarded links are not modified with respect to all their original values. This means that there are one or more possible executions where all or part of guarded links can be executed with their maximal designed range.

Let us consider a fragment of a possible execution, focusing on two guarded links: $(V_s,\ [[2,4]\ [6,8]],V_e)$ and $(G_s,\ [[1,1]\ [2,4]],\ G_e)$. We adopt here a mixed strategy–not an early execution one–just to show that the checked network allows the possibility of other execution strategies. Indeed, the early execution strategy, i.e., "execute any time-point as soon as you are allowed to execute it", is often used in checking the controllability of temporal constraint network with uncertainty, as it avoids to compute the overall allowed ranges for all time-points [39,40] and simplifies the proof of soundness and completeness of checking algorithms, as in our case. However, other execution strategies could be relevant in several application domains. For example, the dual *late execution strategy*, i.e., "execute any time-point at the last allowed moment" has the effect of delaying as much as possible the execution of time-points, thus allowing the maximum amount of time for "preparing" such execution. In general, execution strategies different from the early-execution one require more computation (at run-time) to decide the right value for time-points.

[2] Since we are considering an early execution strategy, conjunct-lower-case edges (that determine upper bounds to the execution time for time-points) are not considered.

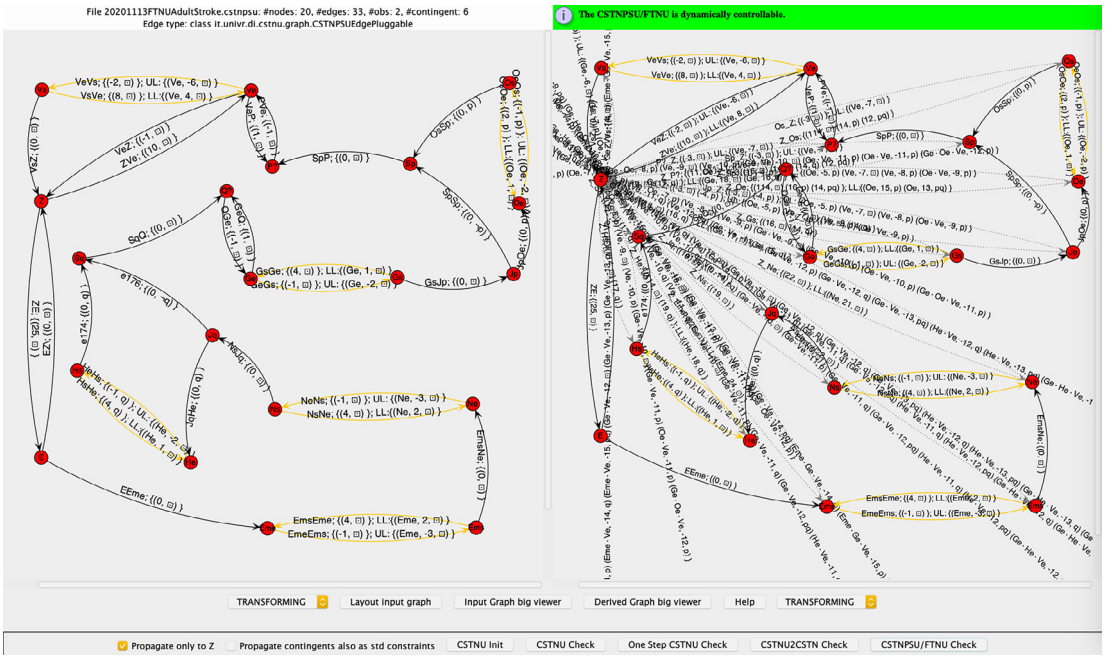


Fig. 10. The distance graph in Fig. 2 after the execution of the DC checking algorithm.

Here, we empirically show how the network evolves through an execution strategy we named "mixed", as it sometimes delays the execution of time-points within the allowed ranges. The execution starts setting $Z = 0$. Then, the only time-point that can be executed is V_s since it is the only one with no negative outgoing edge to other time-points different from Z. Reading the values on the two edges connecting V_s and Z, it results that the time range for V_s is $[0, 4]$. The early execution strategy dictates to execute V_s at 0, but it is simple to verify that any value in $[0, 4]$ is admissible. Let us assume that V_s is executed at 4. The execution is "done" updating the edge $Z \xleftarrow{\langle 0,\ \boxdot\rangle} V_s$ to $Z \xleftarrow{\langle -4,\ \boxdot\rangle} V_s$ and adding the edge $Z \xrightarrow{\langle 4,\ \boxdot\rangle} V_s$. Since V_s is the activation time-point for the guarded link $(V_s,\ [[2, 4]\ [6, 8]],\ V_e)$, it is necessary to update the guarded range before the activation of the guarded link; therefore, the DC checking algorithm is executed and, then, the guarded link (V_s, V_e) fixed to its external range. The resulting range for (V_s, V_e) is $(V_s,\ [[2, 2]\ [6, 6]],\ V_e)$. After such a set, the DC checking algorithm is executed again to propagate such a new range. The DC check updates the guarded link (G_s, G_e) as $(G_s, [[1, 1]\ [2, 3]],\ G_e)$ (see Fig. 11). The restriction of (V_s, V_e) range is due to the presence of constraint $Z \xrightarrow{\langle 10,\ \boxdot\rangle} V_e$ while the restriction of (G_s, G_e) range is due to the presence of constraint $Z \xrightarrow{\langle 25,\ \boxdot\rangle} E$.

Let us consider that the environment executes the contingent time-point V_e at time 10, its last possible allowed instant. Such occurrence is "stored" in the network, replacing the 4 values in the two edges between V_s and V_e (representing the contingent link) by the value 6, i.e., $V_s \xrightarrow{\langle 6,\ \boxdot\rangle} V_e$ and $V_s \xleftarrow{\langle -6,\ \boxdot\rangle} V_e$.

Then, another execution of the DC checking algorithm propagates such occurrence to the rest of the network; it results that the guarded link $(G_s,\ [[1, 1]\ [2, 3]],\ G_e)$ maintains its bounds.

At instant 10, the only time-point that can be considered for the execution is $P?$ since it is the only time-point having as preceding nodes V_e and Z: $P? \xrightarrow{\langle -1,\ \boxdot\rangle} V_e$, and $P? \xrightarrow{\langle -11,\ \boxdot\rangle} Z$. Since there exists also the edge $Z \xrightarrow{\langle 11,\ \boxdot\rangle} P?$ that fixes an upper bound for the execution of $P?, P?$ must be executed at 11.

Once P is executed, let us assume the environment fixes $p = \bot$. Then, all the constraints and time-points related to the scenarios where $p = \top$ must be deleted. Fig. 12 shows the resulting distance graph after p assumed value $\bot$: the constraints and time-points associated with scenarios with $p = \top$ and literal $\neg p$ in the remaining labels have been removed.

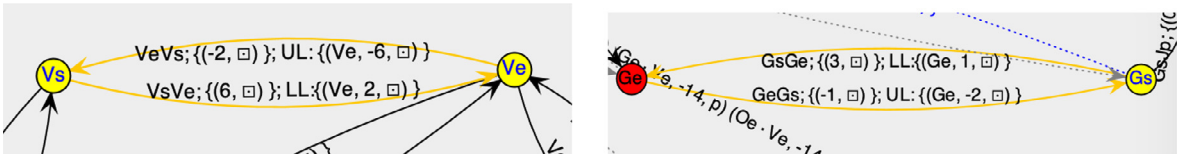


Fig. 11. The updated of the guarded links (V_s, V_e) and (G_s, G_e) in Fig. 2 after the activation of the guarded link (V_s, V_e).

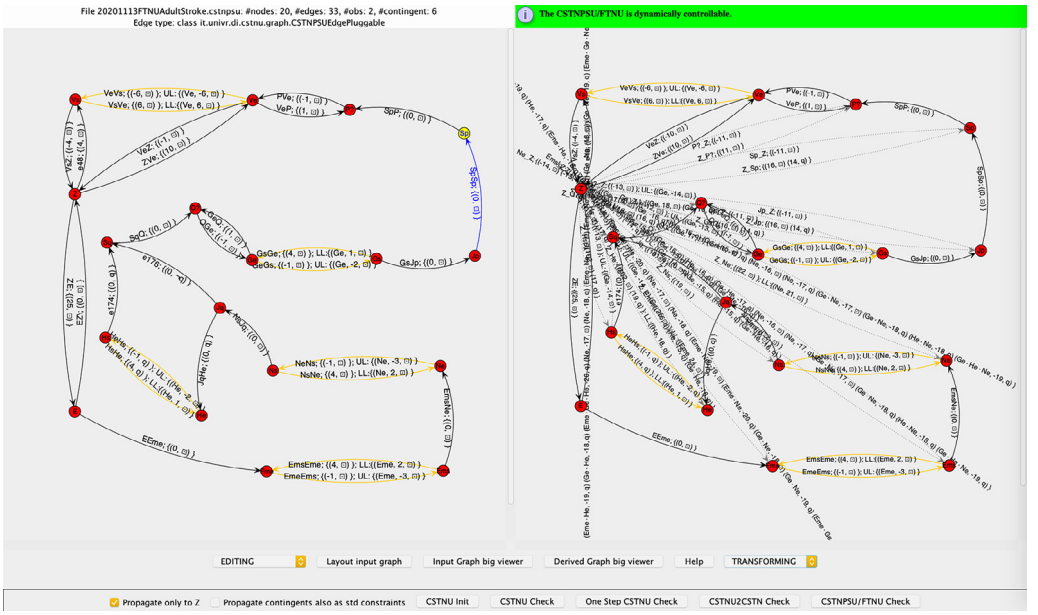


Fig. 12. The distance graph in Fig. 2 after the execution of observation time-point $P?$ where p is set to $\bot$.

Executing the DC-checking algorithm at the instant 11, the previously restricted guarded link $(G_s,\ [[1,1]\ [2,3]],\ G_e)$ is now left to its original range, i.e., $(G_s,\ [[1,1]\ [2,4]],\ G_e)$ because of the execution of $P?$ at 11 and the "removal" of constraints related to scenarios having $p = \top$ "relaxes" the remaining constraints.

Time-point S_p can be now executed because it is the only time-point having its preceding nodes already executed: $S_p \xrightarrow{\langle 0,\ \boxdot\rangle} P?$ and $S_p \xrightarrow{\langle -11,\ \boxdot\rangle} Z$. Since there is also the constraint $Z \xrightarrow{\langle 14,\ q\rangle,\langle 16,\ \boxdot\rangle} S_p$, S_p must be executed before 14 in order to guarantee that this last constraint is satisfied also in case that q will be $\top$. $Q?$ is not yet executed, but the DC checking algorithm is able to determine all constraints with respect to all possible scenarios that may occur.

Let us assume that S_p is executed at 12, i.e., the constraints between Z and S_p becomes $Z \xleftarrow{\langle -12,\ \boxdot\rangle} S_p$ and $Z \xrightarrow{\langle 12,\ \boxdot\rangle} S_p$, respectively.

Then, J_p can be considered. The edges $Z \xrightarrow{\langle 14,\ q\rangle,\langle 16,\ \boxdot\rangle} J_s$ limit the time range for executing J_p to $[12, 14]$.

Let us consider, as the last step of this example, what occurs executing J_p and G_s at different instants. The execution of both time-points J_p and G_s at 12 allows the preservation of guarded link $(G_s,\ [[1,1]\ [2,4]],\ G_e)$, while the execution of J_p at 13 forces the execution of G_s at 13 and the reformulation of the guarded link (G_s, G_e) as $(G_s,\ [[1,1]\ [2,3]],\ G_e)$. As the last possible execution time for J_p, if it is executed at 14, then it is necessary to execute G_s at 14 and to reformulate the guarded link to its core (no flexibility), i.e., $(G_s,\ [[1,1]\ [2,2]],\ G_e)$.

The rest of the execution is managed in a similar way. Updating the remaining constraints after each time-point execution guarantees that guarded link ranges are always correct and updated to their maximal extensions to guarantee the maximal allowed flexibility.

7. Conclusions

In this paper, we considered the problem of checking controllability for temporal constraint networks, where contingent links are allowed to be shrunk until a predefined contingent *core*. Such a problem was previously discussed for other temporal constraint models, i.e., CSTNPSUs and STNPSUs. Here, as a first contribution, we proved that such algorithms do not derive the right ranges for guarded links in such models. Thus, those algorithms cannot be considered for executing FTNU networks, exploiting the nature of guarded links.

Then, we proposed here a new kind of temporal constraint networks, i.e., FTNUs, that improve and refine the previously proposed CSTNPSUs, and a new set of propagation rules that, besides checking whether a network is controllable, are also able to derive the right bounds of guarded links. Such bounds allow us to directly execute the network resulting from DC checking. At any execution of an activation time-point, the new rules determine the right bounds of the corresponding guarded link, making it a fully-fledged contingent link. An implementation of the new DC-checking algorithm is available as a proof-of-concept [38].

Currently, the proposed execution algorithm runs the DC-checking algorithm at each step of an execution, which is quite expensive and not always necessary. Therefore, as for future work, we plan to work towards a more efficient execution algorithm, to avoid runtime redundant computations and to allow the application of different execution strategies. An optimized execution algorithm is important for making possible the application of FTNUs in real contexts, as temporal business process management.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by:

- MIUR, Project *Italian Outstanding Departments, 2018–2022*,
- INdAM, GNCS 2020, Project *Automated Reasoning about Time in Medical and Business Applications*,
- Centro Piattaforme Tecnologiche (CPT) at University of Verona, Italy.

References

[1] T. Vidal, H. Fargier, Contingent durations in temporal csps: from consistency to controllabilities, in: 4th Int. Workshop on Temporal Representation and Reasoning (TIME-1997), 1997, pp. 78–85. doi:10.1109/TIME.1997.600786.
[2] P.H. Morris, N. Muscettola, T. Vidal, Dynamic control of plans with temporal uncertainty, in: 17th International Joint Conference on Artificial Intelligence (IJCAI-2001), 2001, pp. 494–502.
[3] C. Combi, R. Posenato, Controllability in temporal conceptual workflow schemata, in: BPM, 2009, pp. 64–79, doi: 10.1007/978-3-642-03848-8_6.
[4] E. Beaudry, F. Kabanza, F. Michaud, Planning for concurrent action executions under action duration uncertainty using dynamically generated Bayesian networks, in: 20th International Conference on Automated Planning and Scheduling (ICAPS-2010), 2010, pp. 10–17.
[5] A. Micheli, M. Do, D.E. Smith, Compiling away uncertainty in strong temporal planning with uncontrollable durations, in: International Joint Conference on Artificial Intelligence (IJCAI-2015), vol. 2015-Janua, 2015, pp. 1631–1637.
[6] A. Lanz, R. Posenato, C. Combi, M. Reichert, Simple temporal networks with partially shrinkable uncertainty, in: 6th International Conference on Agents and Artificial Intelligence (ICAART 2015), vol. 2, 2015, pp. 370–381. doi:10.5220/0005200903700381.
[7] C. Combi, R. Posenato, Extending Conditional Simple Temporal Networks with Partially Shrinkable Uncertainty, in: N. Alechina, K. Nørvåg, W. Penczek (Eds.), 25th International Symposium on Temporal Representation and Reasoning (TIME 2018), vol. 120 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 9:1–9:15. doi:10.4230/LIPIcs.TIME.2018.9.
[8] R. Posenato, A. Lanz, C. Combi, M. Reichert, Managing time-awareness in modularized processes, Software & Systems Modeling 18 (2) (2019) 1135–1154, doi: 10.1007/s10270-017-0643-4.
[9] R. Dechter, I. Meiri, J. Pearl, Temporal Constraint Networks, Artificial Intelligence 49 (1–3) (1991) 61–95, doi: 10.1016/0004-3702(91)90006-6.
[10] M. Wilson, T. Klos, C. Witteveen, B. Huisman, Flexibility and decoupling in simple temporal networks, Artificial Intelligence 214 (2014) 26–44, doi: 10.1016/j.artint.2014.05.003.
[11] C. Witteveen, Optimising flexibility for simple temporal networks, in: ICAART (2), SciTePress, 2016, pp. 524–531.
[12] L. Hunsberger, Algorithms for a temporal decoupling problem in multi-agent planning, in: 8th Nat. Conf. on Artificial Intelligence (AAAI-2002), 2002, pp. 468–475.
[13] A. Huang, L. Lloyd, M. Omar, B. James, New Perspectives on Flexibility in Simple Temporal Planning, in: 28th Int. Conf. on Automated Planning and Scheduling (ICAPS-2018), 2018, pp. 123–131. URL: https://www.aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/ viewPaper/17775.
[14] L. Khatib, P. Morris, R. Morris, F. Rossi, Temporal Constraint Reasoning With Preferences, in: 17th Int. Joint Conf. on Artificial Intelligence (IJCAI-2001), 2001, pp. 322–327.
[15] S. Badaloni, M. Giacomin, The algebra ia^{fuz}: a framework for qualitative fuzzy temporal reasoning, Artificial Intelligence 170 (10) (2006) 872–908.
[16] C. Muise, J.C. Beck, S.A. McIlraith, Flexible Execution of Partial Order Plans With Temporal Constraints, in: IJCAI2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, 2013, pp. 2328–2335.
[17] P.R. Conrad, J.A. Shah, B.C. Williams, Flexible execution of plans with choice, in: 19th International Conference on Automated Planning and Scheduling, ICAPS 2009 AAAI, 2009, pp. 74–81, URL: http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/730.
[18] P.R. Conrad, B.C. Williams, Drake: An efficient executive for temporal plans with choice, Journal of Artificial Intelligence Research (JAIR) 42 (2011) 607–659, doi: 10.1613/jair.3478.
[19] K. Stergiou, M. Koubarakis, Backtracking algorithms for disjunctions of temporal constraints, Artificial Intelligence 120 (1) (2000) 81–117, doi: 10.1016/S0004-3702(00)00019-9.
[20] M. Cairo, C. Combi, C. Comin, L. Hunsberger, R. Posenato, R. Rizzi, M. Zavatteri, Incorporating Decision Nodes into Conditional Simple Temporal Networks, in: 24th Int. Symp. on Temporal Representation and Reasoning (TIME-2017), vol. 90 of LIPIcs, 2017, pp. 9:1–9:18. doi:10.4230/LIPIcs.TIME.2017.9.
[21] M. Zavatteri, C. Combi, R. Rizzi, L. Viganò, Hybrid SAT-Based Consistency Checking Algorithms for Simple Temporal Networks with Decisions, in: TIME, vol. 147 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 16:1–16:17.
[22] F. Rossi, K.B. Venable, N. Yorke-Smith, Uncertainty in Soft Temporal Constraint Problems: A General Framework and Controllability Algorithms for The Fuzzy Case, J. Artif. Intell. Res. 27 (2006) 617–674, doi: 10.1613/jair.2135.
[23] B.W. Wah, D. Xin, Optimization of bounds in temporal flexible plans with dynamic controllability, Int. J. Artif. Intell. Tools 16 (1) (2007) 17–44, doi: 10.1142/S0218213007003187, URL: https://doi.org/10.1142/S0218213007003187.
[24] L. Hunsberger, R. Posenato, C. Combi, A Sound-and-Complete Propagation-based Algorithm for Checking the Dynamic Consistency of Conditional Simple Temporal Networks, in: F. Grandi, M. Lange, A. Lomuscio (Eds.), 22nd Int. Symp. on Temporal Representation and Reasoning (TIME-2015), 2015, pp. 4–18, doi: 10.1109/TIME.2015.26.
[25] I. Tsamardinos, T. Vidal, M.E. Pollack, CTP: A New Constraint-Based Formalism for Conditional, Temporal Planning, Constraints 8 (4) (2003) 365–388, doi: 10.1023/A:1025894003623.
[26] M. Falda, F. Rossi, K.B. Venable, Dynamic consistency of fuzzy conditional temporal problems, J. Intell. Manuf. 21 (1) (2010) 75–88.

[27] L. Hunsberger, R. Posenato, C. Combi, The Dynamic Controllability of Conditional STNs with Uncertainty, in: Work on Planning and Plan Execution for Real-World Systems (PlanEx) at ICAPS-2012, 2012, pp. 1–8, URL: http://arxiv.org/abs/1212.2005.
[28] C. Combi, L. Hunsberger, R. Posenato, An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty - revisited, in: Agents and Artificial Intelligence, vol. 449 of Communications in Computer and Information Science (CCIS), Springer, 2014, pp. 314–331, doi: 10.1007/978-3-662-44440-5_19.
[29] L. Hunsberger, R. Posenato, Sound-and-Complete Algorithms for Checking the Dynamic Controllability of Conditional Simple Temporal Networks with Uncertainty, in: 25th Int. Symp. on Temporal Representation and Reasoning (TIME-2018), vol. 120 of LIPIcs, 2018, pp. 14:1–14:17. doi:10.4230/LIPIcs.TIME.2018.14.
[30] M. Zavatteri, Conditional Simple Temporal Networks with Uncertainty and Decisions, in: TIME '17, vol. 90 of LIPIcs, 2017, pp. 23:1–23:17. doi:10.4230/LIPIcs.TIME.2017.23.
[31] M. Zavatteri, L. Viganò, Conditional simple temporal networks with uncertainty and decisions, Theoretical Computer Science 797 (2019) 77–101, doi: 10.1016/j.tcs.2018.09.023.
[32] R.T. Effinger, B.C. Williams, G. Kelly, M. Sheehy, Dynamic controllability of temporally-flexible reactive programs, in: 19th International Conference on Automated Planning and Scheduling, ICAPS 2009 AAAI, 2009, pp. 122–129, URL: http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/739.
[33] E.C. Jauch, B. Cucchiara, O. Adeoye, W. Meurer, J. Brice, Y.Y. Chan, N. Gentile, M.F. Hazinski, Part 11: adult stroke: 2010 American Heart Association Guidelines for Cardiopulmonary Resuscitation and Emergency Cardiovascular Care, Circulation 122 (18 Suppl 3) (2010) S818–828, doi: 10.1161/CIRCULATIONAHA.110.971044.
[34] M. Cairo, R. Rizzi, Dynamic Controllability of Conditional Simple Temporal Networks is PSPACE-complete, in: 23rd Int. Symp. on Temporal Representation and Reasoning (TIME-2016), 2016, pp. 90–99. doi:10.1109/TIME.2016.17.
[35] P.H. Morris, N. Muscettola, Temporal dynamic controllability revisited, in: 20th National Conference on Artificial Intelligence (AAAI-2005), 2005, pp. 1193–1198.
[36] P. Morris, A Structural Characterization of Temporal Dynamic Controllability, in: Principles and Practice of Constraint Programming (CP-2006), vol. 4204, 2006, pp. 375–389. doi:10.1007/11889205_28.
[37] L. Hunsberger, R. Posenato, Simpler and Faster Algorithm for Checking the Dynamic Consistency of Conditional Simple Temporal Networks, in: 26th Int. Joint Conf. on Artificial Intelligence, (IJCAI-2018), 2018, pp. 1324–1330. doi:10.24963/ijcai.2018/184.
[38] R. Posenato, CSTNU Tool: A Java Library for Checking Temporal Networks, SoftwareX (2022), doi: 10.1016/j.softx.2021.100905.
[39] L. Hunsberger, Efficient execution of dynamically controllable simple temporal networks with uncertainty, Acta Informatica 53 (2) (2015) 89–147, doi: 10.1007/s00236-015-0227-0.
[40] P. Morris, Dynamic controllability and dispatchability relationships, in: Integration of AI and OR Techniques in Constraint Programming. CPAIOR 2014, vol. 8451 of LNCS, Springer, 2014, pp. 464–479. doi:10.1007/978-3-319-07046-9_33.