


Extending Conditional Simple Temporal Networks with Partially Shrinkable Uncertainty

Carlo Combi

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


carlo.combi@univr.it

 <https://orcid.org/0000-0002-4837-4701>

Roberto Posenato

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

roberto.posenato@univr.it

 <https://orcid.org/0000-0003-0944-0419>

Abstract

The proper handling of temporal constraints is crucial in many domains. As a particular challenge, temporal constraints must be also handled when different specific situations happen (conditional constraints) and when some event occurrences can be only observed at run time (contingent constraints). In this paper we introduce *Conditional Simple Temporal Networks with Partially Shrinkable Uncertainty (CSTNPSUs)*, in which contingent constraints are made more flexible (guarded constraints) and they are also specified as conditional constraints. It turns out that guarded constraints require the ability to reason on both kinds of constraints in a seamless way. In particular, we discuss CSTNPSU features through a motivating example and, then, we introduce the concept of controllability for such networks and the related sound checking algorithm.

2012 ACM Subject Classification Computing methodologies → Temporal reasoning, Computing methodologies → Planning under uncertainty, Computing methodologies → Planning for deterministic actions

Keywords and phrases Conditional Simple Temporal Networks with Uncertainty, Partial Shrinkable Temporal Constraint, Dynamic Controllability, Temporal Constraints

Digital Object Identifier 10.4230/LIPIcs.TIME.2018.9

1 Introduction

Temporal constraint networks have been adopted in different research areas for reasoning on temporal requirements. Among such areas, we mention here planning and scheduling [1, 19, 24], business process [7, 14, 18] and healthcare informatics [4][5].

In such areas the proposed temporal constraint models allow the representation of both conditional and contingent constraints. In general, *conditional constraints* hold only when some specific situations happen. Thus, it is possible to specify in a compact way constraints holding in different contexts. Sometimes such contexts are related to decisions made by the executing agent, while in other cases, they are related to external events [8][23]. *Contingent constraints* allow the representation of event occurrences that are not under the control of the executing agent but are tied to happen within some specified time interval. Therefore, it is necessary to guarantee that all the other constraints are consistent for all possible event occurrences [21].



© Carlo Combi and Roberto Posenato;

licensed under Creative Commons License CC-BY

25th International Symposium on Temporal Representation and Reasoning (TIME 2018).

Editors: Natasha Alechina, Kjetil Nørsvåg, and Wojciech Penczek; Article No. 9; pp. 9:1–9:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Recently, a new temporal constraint model, called *Simple Temporal Network with Partially Shrinkable Uncertainty (STNPSU)*, was proposed for the analysis of modular temporal business processes [17]. It introduces a new type of constraint, called *guarded* constraint, and allows the representation of temporal features of business sub-processes as guarded constraints. A guarded constraint is a contingent constraint that is partially under control of the executing agent (i.e., it may be shrunk to a *core*).

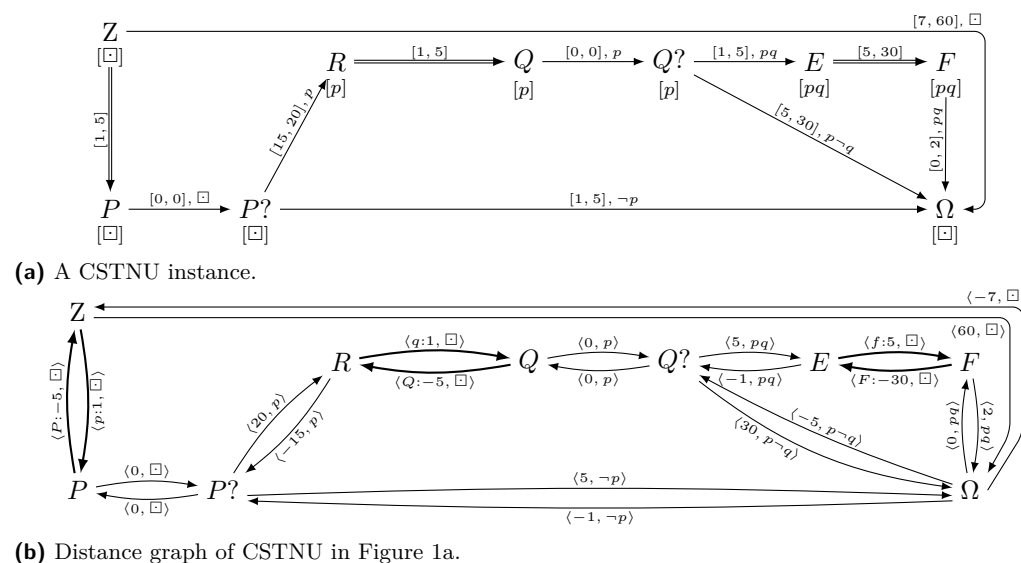
STNPSU lacks of an important feature, especially in the business process domain: the possibility of expressing conditional (possibly guarded) constraints. To the best of our knowledge, the issue of representing conditional constraints and guarded constraints in the same model has not been considered in the literature to date.

Contributions. The novelty of this paper, therefore, is that it focuses on the representation of and reasoning on both conditional constraints and guarded ones. In particular, we introduce *Conditional Simple Temporal Networks with Partially Shrinkable Uncertainty (CSTNPSUs)*, in which all constraints (guarded included) may be specified with respect to particular contexts. At first glance, conditional constraints and guarded ones seem to be orthogonal features that may be managed in an independent way. When taking a closer view on them, however, it turns out that conditional constraints in combination with the guarded ones require the ability to reason on both constraints in a seamless way, because, for example, reasoning techniques for guarded constraints are no longer applicable. Then, we focus on the *dynamic controllability* (DC) of the proposed model introducing a new DC checking algorithm. In general, DC corresponds to the capability of an executing agent to dynamically execute a network for *all* allowed occurrences of *all* guarded events, while still satisfying *all* temporal constraints in *all* possible situations (which are revealed at run time in incremental way); i.e., DC ensures that it is possible to assign a timestamp to each node of a network without any need to modify any guarded *core* to be able to satisfy anyone of the other temporal constraints in any occurring situation.

Structure of the paper. The rest of the paper is organized as follows. Section 2 provides the fundamentals of Conditional Simple Temporal Problem with Uncertainty (CSTNU) and checking related dynamic controllability. Moreover, we introduce here the motivating example we use throughout the paper. Section 3 provides the definition of the new model Conditional Simple Temporal Networks with Partially Shrinkable Uncertainty. Then, it presents the concept of dynamic controllability and how it can be checked reducing the problem to the CSTNU controllability check. Finally, we propose an improved controllability check algorithm by extending a recently proposed sound-and-complete algorithm for CSTNU to deal with the introduced guarded constraints. Section 5 gives some concluding remarks and outlines some future work.

2 Background and Related Work

Given a set \mathcal{P} of propositional letters, a *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 \square . 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 $\ell_1 \Rightarrow \ell_2$) if and only if all literals in ℓ_2 appear in ℓ_1 too.



(a) A CSTNU instance.

(b) Distance graph of CSTNU in Figure 1a.

Time-point	Meaning
Z	Begin First Blood Test
P	End First Blood Test
P?	Check Blood Test (generates truth value for p ; $p = \top$ indicates positive reaction)
R	Begin Second Blood Test
Q	End Second Blood Test
Q?	Check again Blood Test (only applicable if $p = \top$; generates truth value for q ; $q = \top$ indicates positive reaction)
E	Do Emergency Procedure (only applicable if $p = q = \top$)
F	End Emergency Procedure (only applicable if $p = q = \top$)
Ω	End

Constraints are shown as single arrows, whereas contingent links as double ones. Each single-arrow edge between X and Y is decorated by a range $[l, u]$ and label ℓ corresponding to constraint $(l \leq Y - X \leq u, \ell)$. Each double-arrow edge between A and C is decorated by a range $[x, y]$ corresponding to constraint (A, x, y, C) . Each time-point has a label $\ell \in \mathcal{P}^*$ representing the scenario in which it is present.

■ Figure 1 Example of CSTNU.

In this section, we recall the definition of Conditional Simple Temporal Network with Uncertainty [12] and some basic properties presented in [13] that must hold also in such networks.

► **Definition 1** (Conditional Simple Temporal Network with Uncertainty). A *Conditional Simple Temporal Network with Uncertainty* (CSTNU) is a tuple $(\mathcal{T}, \mathcal{OT}, \mathcal{P}, L, O, \mathcal{C}, \mathcal{L})$, where:

- $\mathcal{T} = \{X, Y, \dots\}$ is a finite set of *time-points* (i.e., variables with continuous domain).
- $\mathcal{OT} \subseteq \mathcal{T}$ is a set of *observation time-points*.
- $\mathcal{P} = \{p, q, \dots\}$ is a finite set of propositional letters.
- $L: \mathcal{T} \rightarrow \mathcal{P}^*$ is a function assigning a label to each time-point $X \in \mathcal{T}$.
- $O: \mathcal{P} \rightarrow \mathcal{OT}$ is a bijection associating a unique observation time-point to each propositional letter.
- \mathcal{C} is a set of (*labeled*) *constraints* each one having the form $(l \leq Y - X \leq u, \ell)$, where $X, Y \in \mathcal{T}$, $l, u \in \mathbb{R}$ with $l \leq u$ and $\ell \in \mathcal{P}^*$.
- \mathcal{L} is a set of *contingent links* each having the form (A, x, y, C) , where $A \in \mathcal{T}$ and $C \in \mathcal{T} \setminus \mathcal{OT}$ are different time-points (written $A \not\equiv C$), $0 < x < y < \infty$, $L(A) = L(C)$. For any pair $(A_1, x_1, y_1, C_1), (A_2, x_2, y_2, C_2) \in \mathcal{L}$ with $A_1 \not\equiv A_2$ it holds $C_1 \not\equiv C_2$.

- For each constraint $(l \leq Y - X \leq u, \ell) \in \mathcal{C}$, $\ell \Rightarrow L(Y) \wedge L(X)$ (*constraint label coherence*).
- For each literal p or $\neg p$ appearing in ℓ , $\ell \Rightarrow L(O(p))$ (*constraint label honesty*).
- For each $X \in \mathcal{T}$, if literal p or $\neg p$ appears in $L(X)$, then $L(X) \Rightarrow L(O(p))$, and $O(p)$ has to occur before X , i.e., $(\epsilon \leq X - O(p) \leq +\infty, L(X)) \in \mathcal{C}$ for some $\epsilon > 0$ (*time-point label honesty*).

For any contingent link (A, x, y, C) , A is called *activation time-point* whereas C is called *contingent time-point*. Once A is executed, C is only observed to occur. However, C is guaranteed to execute such that $C - A \in [x, y]$. Moreover, a contingent link has an implicit label given by the label $\ell = L(A) = L(C)$.

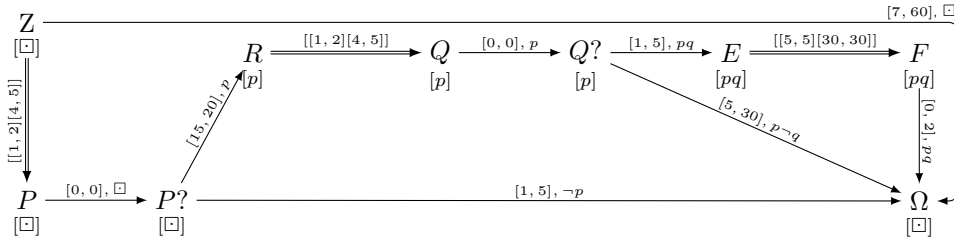
Figure 1a gives an example of CSTNU representing a temporal plan in a healthcare setting with minutes as a temporal granularity. The CSTNU is shown in its graphical form where nodes represent time-points, single-line directed edges represent constraints, and double-line directed edges represent contingent links. It starts with the execution of Z , which is *usually assumed to be the first time-point to be executed*, and ends with the execution of Ω , which is *usually assumed to be the last time-point to be executed*. For example, time-point $P?$ represents the time at which a particular blood test is checked. This test generates a truth value for the propositional letter p , where $p = \top$ indicates that the patient tested positive. In this example, if the test generates a positive result, then the test is repeated at time-point R and ends at time-point Q ; then, at time-point $Q?$ the truth value for the propositional letter q is set. Since the time-points R , Q , and $Q?$ apply only in scenarios where $p = \top$, they are labeled by p . Similarly, the edge from $P?$ to R is labeled by p . It represents the constraint, $R - P? \in [15, 20]$ (i.e., the second test must be performed between 15 and 20 minutes after the first one). Finally, note that the constraints from $Q?$ to E , and from E to Ω are labeled by pq , indicating that they apply only in scenarios where both p and q are \top . The three contingent links represent the execution of the first blood test, the second one, and the emergency procedure, respectively. Constraints and time-points labeled by \square are always taken into consideration. Constraint $(7 \leq \Omega - Z \leq 60, \square)$ represents a constraint for the overall execution time of the network. It is worth noting that the couple P and $P?$ represents two time-points executed at the same time but in the given order. The first one, P , refers to the end of blood test, while the second one refers to observing the blood test result (the same for couple Q and $Q?$). They must be different, according to CSTNU model.

In general, by *executing* a non-contingent time-point we mean to assign a real value to it, while a contingent time-point is *executed* when the “environment” sets a real value to it.

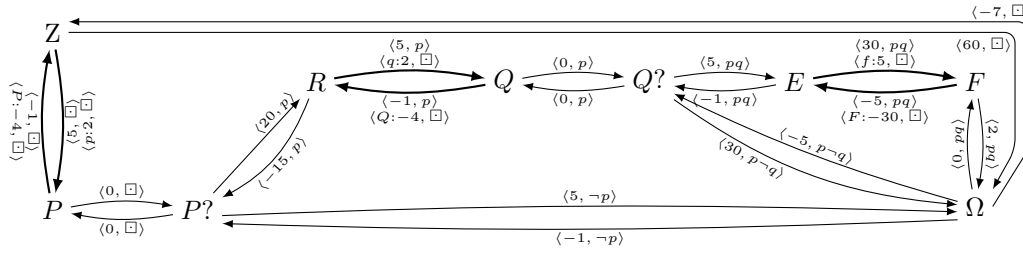
The truth values of propositions and the duration of contingent links in a CSTNU are not known in advance. Instead, they are incrementally revealed over time as the corresponding observation/contingent time-points are executed, respectively. A *dynamic execution strategy* for executing the time-points in a CSTNU is defined in a way that it can react to observations and contingent time-points as time passes (after an $\epsilon > 0$ reaction time). A *viable and dynamic execution strategy* for a CSTNU is a strategy that guarantees that all relevant constraints will be satisfied no matter which truth values for propositions and durations for contingent links are incrementally revealed over time. A CSTNU with such a strategy is called *dynamically controllable (DC)*. The problem of checking the dynamic controllability (*DC-checking problem*) consists of verifying, at design time, whether a CSTNU is DC.

In [2], authors showed that the problem of checking the dynamic consistency of conditional simple temporal networks (CSTNs) is PSPACE-complete. It is straightforward to show that a CSTN is a special case of a CSTNU: it is a network where there is no contingent links. Therefore, the CSTNU DC-checking problem is PSPACE-hard.

Nevertheless, in literature there are some proposals to solve the DC-checking problem using different algorithm techniques [2, 3, 6] and only one of them, presented in [6], results



(a) A CSTNPSU extension of the CSTNU in Figure 1a.



(b) Distance graph of CSTNPSU in Figure 2a.

■ **Figure 2** A CSTNPSU and its distance graph.

to be applicable to real world problems as shown in [5, 15, 19]. It determines all possible constraints from the initial ones using a suitable set of constraint propagation rules, while verifying their satisfiability. Recently, a new *sound-and-complete* algorithm for CSTNUs DC checking has been proposed in [10], where the authors extend and merge the algorithms proposed in [5] and [13] using a new technique and show its practical applicability.

3 Conditional Simple Temporal Networks with Partially Shrinkable Uncertainty

In some real-world contexts, contingent links seem to be not flexible enough to represent task durations. Indeed, it may be that in some cases task duration constraints can be slightly modified during the execution of a complex plan. For example, the duration range [1, 5] of blood tests in the CSTNU in Figure 1a could be made more flexible by saying that it can be reduced to [1, 4] if needed. In other contexts it could be that also the lower bound of a contingent link can be modified, e.g., range [2, 4] for blood test could be less expensive because it requires different devices.

In [16] and [17] the authors addressed this issue in the context of STNUs (and related business process modeling). A STNU is a simplified version of a CSTNU without conditions [21]. In this section we extend such approach by introducing a new proposal of temporal constraint networks, named *Conditional Simple Temporal Networks with Partially Shrinkable Uncertainty (CSTNPSU)*.

Conditional Simple Temporal Networks with Partially Shrinkable Uncertainty is an extension of CSTNU where contingent links are generalized to *guarded links*. Each guarded link has the form, $(A, [x, x'], [y', y], C)$, where A is the activation time-point, and C is the contingent one, and $0 < x < y < \infty$, $x \leq x'$, $x' < y'$, $y' \leq y$. Once A is executed, C is guaranteed to execute such that $C - A \in [x, y]$. However, bounds x and y can be modified before executing A with the limitations specified by the two *guards* x' and y' , i.e., x can be incremented up to x' , while y can be reduced down to y' . The range $[x', y'] \subseteq [x, y]$

of a guarded link represents its unshrinkable range, named guarded *core*. Moreover, if $x = x' \wedge y = y'$ hold, the guarded link is equivalent to a contingent link of CSTNU.

► **Definition 2 (CSTNPSU).** A *Conditional Simple Temporal Networks with Partially Shrinkable Uncertainty* (CSTNPSU) is a tuple $(\mathcal{T}, \mathcal{OT}, \mathcal{P}, L, O, \mathcal{C}, \mathcal{G})$, where $\mathcal{T}, \mathcal{OT}, \mathcal{P}, L, O, \mathcal{C}$ are the same as in CSTNU definition and have the properties specified in Theorem 1. \mathcal{G} is a set of *guarded links* each having the form $(A, [x, x'], [y', y], C)$, where A and C are time-points, $x, y \in \mathbb{R}$, $0 < x \leq x' < y' \leq y < \infty$, $L(A) = L(C)$.

Figure 2a shows an extension of the example introduced in Section 2. Previous contingent links have been replaced by guarded links. In particular, both contingent links $(R, 1, 5, Q)$ and $(Z, 1, 5, P)$ were replaced by the guarded links $(R, [1, 2], [4, 5], Q)$ and $(Z, [1, 2], [4, 5], P)$, respectively, representing the fact that blood tests do not need to have $[1, 5]$ as contingent range but the less demanding $[2, 4]$ if it is required during the execution. Contingent link $(E, 5, 30, F)$ was replaced by the guarded link $(E, [5, 5], [30, 30], F)$ meaning that, in this case, the guarded link is identical to the original contingent one: such constraint cannot be partially shrunk.

3.1 Dynamic Controllability of CSTNPSU

Informally, a CSTNPSU is dynamically controllable if there exists a strategy for executing the time-points in the network such that all constraints are guaranteed to be satisfied no matter how the durations of the guarded links turn out, and no matter how the observations of the various propositions turn out, in real time – paying attention to the fact that in any given scenario, only the time-points whose labels are true in that scenario need to be executed, and only the constraints whose labels are meaningful in that scenario need to be satisfied.

In order to verify whether a CSTNPSU is dynamically controllable it is possible to consider the CSTNU corresponding to the CSTNPSU where all guarded links have been restricted to their core.

► **Definition 3 (Core Situations).** Let $\mathcal{S} = (\mathcal{T}, \mathcal{OT}, \mathcal{P}, L, O, \mathcal{C}, \mathcal{G})$ be a CSTNPSU. If \mathcal{G} contains k guarded links, $(A_1, [x_1, x'_1], [y'_1, y_1], C_1), \dots, (A_k, [x_k, x'_k], [y'_k, y_k], C_k)$, then $\Omega_{\mathcal{S}} = [x'_1, y'_1] \times \dots \times [x'_k, y'_k]$ is called the *space of core situations* for \mathcal{S} . Any $\omega = (d_1, \dots, d_k) \in \Omega_{\mathcal{S}}$ is called a *core situation*.

Given the space of core situations $\Omega_{\mathcal{S}}$ of a CSTNPSU \mathcal{S} , a projection of \mathcal{S} onto a CSTNU can be obtained as follows: each guarded link in \mathcal{G} is replaced by a contingent link with the range specified in $\Omega_{\mathcal{S}}$.

► **Definition 4 (Core CSTNU of a CSTNPSU).** Let $\mathcal{S} = (\mathcal{T}, \mathcal{OT}, \mathcal{P}, L, O, \mathcal{C}, \mathcal{G})$ be a CSTNPSU. The *core CSTNU* of \mathcal{S} onto its space of core situations $\Omega_{\mathcal{S}}$ corresponds to a CSTNU $(\mathcal{T}, \mathcal{OT}, \mathcal{P}, L, O, \mathcal{C}, \mathcal{L})$ where:

$$\mathcal{L} = \{(A_i, x'_i, y'_i, C_i) \mid 1 \leq i \leq k, [x'_i, y'_i] \text{ is the } i\text{-th component of } \Omega_{\mathcal{S}}\}$$

Finally, this leads us to the dynamic controllability of an CSTNPSU. We provide a formalization of the dynamic controllability of an CSTNPSU based on the dynamic controllability of a CSTNU. We choose this approach since the formalization of dynamic controllability of CSTNU is robust and verified in literature [6][10][11][12].

► **Theorem 5 (Dynamic Controllability of CSTNPSU).** A *CSTNPSU* $\mathcal{S} = (\mathcal{T}, \mathcal{OT}, \mathcal{P}, L, O, \mathcal{C}, \mathcal{G})$ is dynamically controllable (DC) if its core CSTNU is dynamically controllable.

Proof. \Rightarrow It is a matter of definitions to show that, if the core CSTNU is DC, then \mathcal{S} is DC as well. Indeed, it is always possible to restrict \mathcal{S} to its core situations, and for each core situation of \mathcal{S} , a viable dynamic execution strategy (DES) for the core CSTNU, is also a viable DES for \mathcal{S} .

\Leftarrow If the core CSTNU is not DC, at least one core situation ω exists for which no DES exists. Hence, \mathcal{S} is not DC either. \blacktriangleleft

To check the controllability of a CSTNPSU, we proceed in two steps:

1. we consider the sound-and-complete CSTNU DC checking algorithm proposed in [10];
2. we extend such algorithm for directly checking the controllability of CSTNPSUs obtaining also minimized constraint networks.

3.2 An improved CSTNU DC checking algorithm

Before introducing the algorithm, it is necessary to consider the CSTNU representation by *distance graphs*. In [10], the authors showed that, for the DC checking task, it is possible to consider a *streamlined* representation of distance graphs where nodes are not labeled, i.e., labels are present only on constraints¹. A *distance graph* $\mathcal{D} = (\mathcal{T}, \mathcal{E})$ of a CSTNU $(\mathcal{T}, \mathcal{OT}, \mathcal{P}, L, O, \mathcal{C}, \mathcal{L})$ is a graph having the same set of nodes and edges derived from the upper and lower bound constraints.

Each constraint $(l \leq Y - X \leq u, \ell) \in \mathcal{C}$ between a pair of time-points X and Y is represented in the distance graph as two *ordinary edges* in \mathcal{E} : $X \xrightarrow{(u, \ell)} Y$, representing constraint $Y \leq X + u$, and $X \xleftarrow{(-l, \ell)} Y$, for constraint $Y \geq X + l$.

Moreover, for each contingent link (A, x, y, C) between a pair of time-points A and C , \mathcal{E} contains two other edges with special values, called *lower* and *upper case values* [20]. An edge with a lower case value, $A \xrightarrow{(cx, \square)} C$, represents the fact that C cannot be forced to be executed at a time greater than x after A , i.e., it is not possible to add a constraint $A \xleftarrow{(-x', \square)} C, x < x'$ to the network. In turn, an edge with an upper case value, $A \xleftarrow{(C:-y, \square)} C$, represents the fact that C cannot be forced to be executed at a time less than y after A , i.e., it is not possible to add a constraint $A \xrightarrow{(y', \square)} C, y' < y$ to the network.

These two kinds of edges containing special values are fundamental for determining the dynamic controllability of the network as explained in the following.

Figure 1b depicts the distance graph of CSTNU in Figure 1a. If multiple edges exist between two nodes (e.g., an ordinary and an upper-case edge), for the sake of readability, we draw only one arrow between the nodes and annotate it with the values of the respective edges.

The DC checking algorithm, CSTNU-DC-Check, consists of deriving new constraints (edges) through the application of propagation rules. Such technique narrows the search space of possible partial solutions by creating equivalent yet more explicit networks. In such equivalent networks, the search for a solution is more efficient. On the other hand, solving a complete problem by inference often requires the addition of an exponential number of new constraints [9].

Propagation rules for CSTNUs fall into two main groups. The first group extends the edge-generation rules proposed in [20] for STNU DC checking algorithm to accommodate labeled edges; the second group consists of label-modification rules that address interactions involving observation nodes.

¹ The notation of constraint values we use in this paper is slightly different from the one adopted in [10].

■ **Table 1** Edge-generation rules of CSTNU-DC-Check algorithm.

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

$Z = 0$; $A, C, X, Y \in \mathcal{T}$; 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.

The first group consists of four rules, rG_1 – rG_4 , depicted in Table 1. All rules only generate ordinary or conjuncted upper-case edges (introduced below).

Each rule generates an edge whose label, e.g., $\alpha\beta$, is the conjunction of the labels of its parent edges, e.g., α and β . If the resulting label is unsatisfiable (e.g., $p\neg p$), then the new edge is not generated (or kept). By \aleph we represent a, possibly empty, conjunction of one or more upper-case names of contingent nodes.

When the generated edge contains a label having a non-empty \aleph , then it represents an extension of *wait* constraints [21]. An edge labeled by a non-empty conjuncted upper-case value, i.e., $X \xleftarrow{\langle \aleph:w, \square \rangle} Y$, represents the following constraint²: *as long any of contingent time-points in \aleph remains unexecuted, Y must wait at least w units after the execution of X* . If any contingent time-point in \aleph is executed before w units after the execution of X , then the constraint is not significant and, therefore, ignored.

Effectively, rule rG_4 does not generate a new conjuncted upper-case value, but reduces one already present removing a contingent name from it. Moreover, rG_4 (as well as rM_1 and rM_2) can handle also a new kind of labels, named *q-labels*, that indicate that a constraint need only hold as long as the value of its special literals, named *q-literals*, are unknown [10].

More formally, q-literals and q-label can be introduced as the following definition.

► **Definition 6** (Q-literals, q-labels). If $p \in \mathcal{P}$, then $?p$ is a *q-literal*. A *q-label* is a (possibly empty) conjunction of literals and/or q-literals. \mathcal{Q}^* denotes the set of all q-labels. (For example, $p(?q)\neg r$ and $(?q)(?r)t\neg u$ are both q-labels.)

The \star operator extends ordinary label conjunction to q-labels. Intuitively, if constraint C_1 is labeled by p , and constraint C_2 is labeled by $\neg p$, then *both* C_1 and C_2 must hold as long as p is unknown, which is represented by $p\star\neg p = ?p$.

► **Definition 7** (\star). The operator, $\star: \mathcal{Q}^* \times \mathcal{Q}^* \rightarrow \mathcal{Q}^*$, is defined as follows. First, for any $p \in \mathcal{P}$, $p\star p = p$ and $\neg p\star\neg p = \neg p$; otherwise, for any $p_1, p_2 \in \{p, \neg p, ?p\}$, $p_1\star p_2 = ?p$. Next, for any $\ell_1, \ell_2 \in \mathcal{Q}^*$, $\ell_1\star\ell_2 \in \mathcal{Q}^*$ denotes the conjunction obtained by applying \star in pairwise fashion to matching literals from ℓ_1 and ℓ_2 , and conjoining any unmatched literals.

(For example, $(p\neg q(?r)t)\star(qr\neg s) = p(?q)(?r)\neg st$.)

² Morris et al. [21] named it also as *conditional*. Hereinafter, we call it *wait* to avoid confusion with respect to the introduced conditional constraints.

Moreover, if in $X \xleftarrow{\langle \aleph:w, \square \rangle} Y$ \aleph is empty, then the wait constraint degenerates to an ordinary one $X \xleftarrow{\langle w, \square \rangle} Y$.

■ **Table 2** Updated Upper-Case Removal Rule of CSTNPSU-DC-Check algorithm.

Rule	Conditions	Pre-existing and Generated Edges
rG_4^*	$m = \max\{v, w - x\}$, x is the lower bound of the guarded link $(A, [x, x'], [y', y], C)$, $C \notin \mathbb{N}\mathbb{N}_1$, $\beta, \gamma \in \mathcal{Q}^*$, and γ entails γ' .	$Y \xrightarrow[\langle \mathbb{N}\mathbb{N}_1 : m, \beta \star \gamma \rangle]{\langle C\mathbb{N} : v, \beta \rangle} Z \xleftarrow{\langle \mathbb{N}_1 : w, \gamma \rangle} A \xrightleftharpoons[\langle -x, \gamma' \rangle; \langle C : -y', \square \rangle]{\langle y, \gamma' \rangle; \langle c : x', \square \rangle} C$
$Z = 0$; $A, C, Y \in \mathcal{T}$; C is contingent; each of \mathbb{N} and \mathbb{N}_1 is a conjunction of one or more upper-case names of contingent nodes, possibly empty.		

The second group of propagation rules, composed by rM_1 and rM_2 , consists of a variety of label-modification rules that share some resemblance to rule rG_4 . The label-modification rules rM_1 and rM_2 have the same general flavor, except that they deal with the uncertainty associated with observation nodes, rather than contingent links.

For example, consider the edge, $Z \xleftarrow{\langle w, \alpha p \rangle} P?$, where neither p nor $\neg p$ appears in α , and $w < 0$. This edge represents the requirement that “in scenarios where αp is true, since $Z = 0$, $P? \geq -w$ must hold.” But that, in turn, implies that the truth value of p cannot be known at the time 0, i.e., when Z is executed. And, of course, the truth value of p cannot be known when the decision to execute $P?$ is made either. As a result, decisions about when to execute $P?$ cannot depend on the truth value of p . Thus, the label on the edge from $P?$ to Z should be modified to remove the occurrence of p , yielding the new edge, $Z \xleftarrow{\langle w, \alpha \rangle} P?$, which represents the constraint that in scenarios where α holds, $Z - P? \leq w$ must hold. This is the idea behind the label-modification rule, rM_1 , shown in Table 1. While rM_1 simplifies labels on edges outgoing from an observation time-point to Z , rule rM_2 simplifies labels on edges going to Z with respect to the labels present on edges from observation time-points to Z .

3.3 DC-Checking for CSTNPSU considering Guarded Links

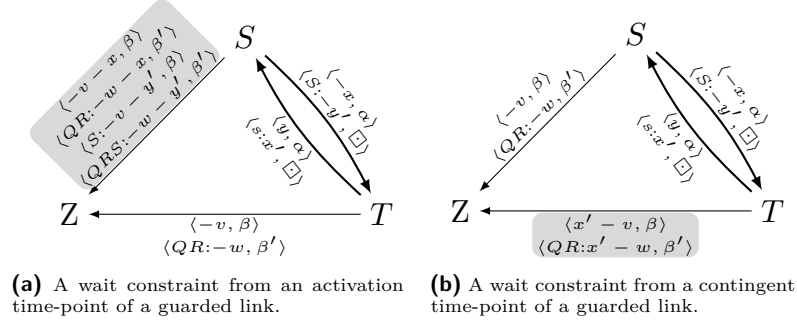
The dynamic controllability of an CSTNPSU may be checked without the need of restricting it to its core CSTNU. First, we extend the concept of distance graph. Then, we extend propagation rules in order to adapt the DC-checking algorithm for CSTNU to CSTNPSU.

► **Definition 8** (Distance Graph of a CSTNPSU). The distance graph for a CSTNPSU \mathcal{S} has the form $(\mathcal{T}, \mathcal{E})$, where each (unlabeled) time-point in \mathcal{T} corresponds to a node in \mathcal{S} and \mathcal{E} is a set of ordinary and special value edges, defined as follow:

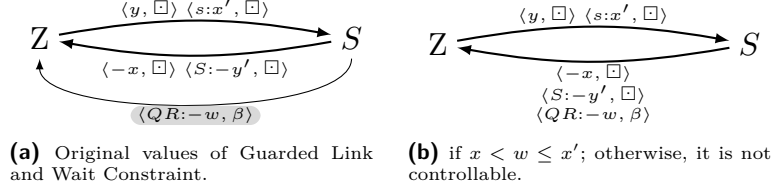
- Each constraint $(l \leq Y - X \leq u, \ell)$ of \mathcal{S} is represented by two ordinary edges $X \xrightarrow{\langle u, \ell \rangle} Y$ and $X \xleftarrow{\langle -l, \ell \rangle} Y$.
- Each guarded link $(A, [x, x'], [y', y], C)$ having $\ell = L(A) = L(C)$ of \mathcal{S} is represented by
 - two ordinary edges $A \xrightarrow{\langle y, \ell \rangle} C$ and $A \xleftarrow{\langle -x, \ell \rangle} C$, considering lower and upper bounds x, y , respectively,
 - one lower-case edge $A \xrightarrow{\langle c : x', \square \rangle} C$, considering guard x' , and
 - one upper-case edge $A \xleftarrow{\langle C : -y', \square \rangle} C$, considering guard y' .

Figure 2b shows the distance graph of CSTPSU of Figure 2a.

In the following we will show that rules of Table 1 may be reused in order to check dynamic controllability of a CSTNPSU, provided that rule rG_4 has to be refined as rG_4^* (cf. Table 2) where the value to use for comparison is the lower bound of a guarded link instead of the lower case value used in the original rule.



■ **Figure 3** Combining a Guarded Link with a Wait Constraint.



■ **Figure 4** Combining a Guarded Link with a derived Wait Constraint.

Particularly, we analyze all possible combinations of edges between three nodes of a CSTNPSU graph and show that the resulting distance graph has no negative loops if and only if the distance graph of the core CSTNU has no negative loops as well. Without loss of generality, we focus only on the more interesting cases when Z may be also an activation time point of a guarded link.

Figure 3a shows how a wait constraint on T (i.e., from T to Z having a conjuncted upper-case value) may be combined with a guarded link from T to S. Labels β and β' entail α and v and w are non negative (otherwise, rules don't apply). The shadow values denotes the ones generated by applying rG_1 and rG_3 rules for the new edge between S and Z. It can be shown that the generated edge is similar to the one that would be generated for the core CSTNU and, most important, the resulting temporal constraint is equivalent. Indeed, the generated edge for the core CSTNU would have only two labeled values that differ from the ones shown in Figure 3a: $\langle -v - x', \beta \rangle$, and $\langle QR: -w - x', \beta' \rangle$ instead of $\langle -v - x, \beta \rangle$, and $\langle QR: -w - x, \beta' \rangle$, respectively. Such two different values are not relevant for checking the DC property because the two labeled values $\langle S: -v - y', \beta \rangle$ and $\langle QRS: -w - y', \beta' \rangle$, present also in the core CSTNU, are stronger and, therefore, make the former redundant.

Figure 3b depicts a similar case where a wait constraint is present on a contingent time-point (i.e., from S to Z). In this case, applying rG_2 , one wait constraint and one ordinary constraint must be added between T and Z. Note that also rG_1 can be applied, but the resulting labeled values have the same form but with weaker values than the ones generated by rG_2 rule and, therefore, they can be ignored. It is easy to verify that two new labeled values are identical to the ones that would be generated for the core CSTNU.

One of the most interesting cases is depicted in Figure 4a, where Z is the activation time-point of a guarded link. Due to possible previous constraint propagations, a wait constraint $S \xrightarrow{\langle QR: -w, \beta \rangle} Z$ has been added next to the guarded link. For the case $x < w \leq x'$ the result is depicted in Figure 4b. Note that for a CSTNU, such case is not possible. Indeed, in a CSTNU $x' = x$ and, therefore, there are two cases: either $w > x' = x$ (i.e., the CSTNU is not controllable) or $w \leq x$ (i.e., the wait constraint is redundant) holds. For a

Algorithm 1: CSTNPSU-DC-Check(G).

Input: $G = (\mathcal{T}, \mathcal{OT}, \mathcal{P}, L, O, \mathcal{C}, \mathcal{G})$: a CSTNPSU 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' the lower bound $X \xrightarrow{(0, \square)} Z$
 Add to G' the global upper bound $Z \xrightarrow{(h, \square)} X$
do
 $G' = rM_1(G')$ // Label Modification
 $G' = rM_2(G')$
 $G' = rG_1(G')$ // Edge Generation
 $G' = rG_2(G')$
 $G' = rG_3(G')$
 $G' = rG_4^*(G')$
 if (any negative cycle has been found) **then return not DC**
while (rules are applied)
return DC

CSTNPSU, Figure 4b can be interpreted as follows: wait $\langle QR: -w, \beta \rangle$ is not relevant for the controllability check because $\langle S: -y', \alpha \rangle$ is more restrictive. On the other hand, when executing the CSTNPSU the wait constraint must be observed in the β scenario. Particularly, if the guarded link is activated prior to the execution of time-point R and time-point Q , w must be used as lower bound when executing the guarded link. Otherwise, x is used as lower bound.

Using the above technique, we can prove that the propagation rules can be applied to all possible combinations of distance graph edges (constraints) to derive new constraints retaining the property that there are (no) negative loops if and only if the distance graph of the core CSTNU has (no) negative loops as well.

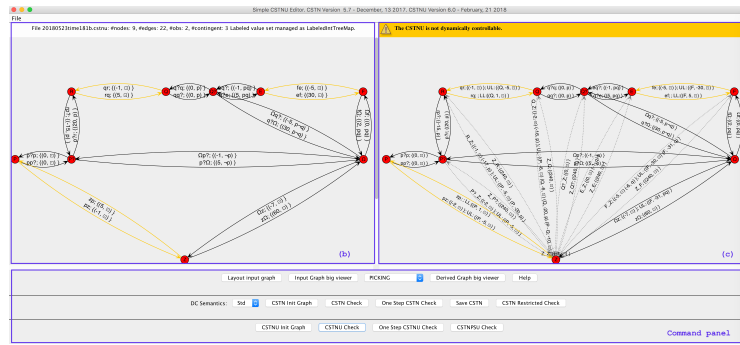
Algorithm 1 depicts the pseudo-code for dynamic controllability check, based on the propagation rules presented in Table 1–2. As shown in [10], the termination of the DC checking algorithm is guaranteed by adding a global upper upper bound and the algorithm is sound-and-complete. As regards the computational complexity of Algorithm 1, it has been shown that an upper bound is $O(M|\mathcal{T}|^4 3^{|\mathcal{P}|} 2^{|\mathcal{L}|})$, where M is the maximum absolute value of any weight in the graph [10].

4 Proof Of Concept

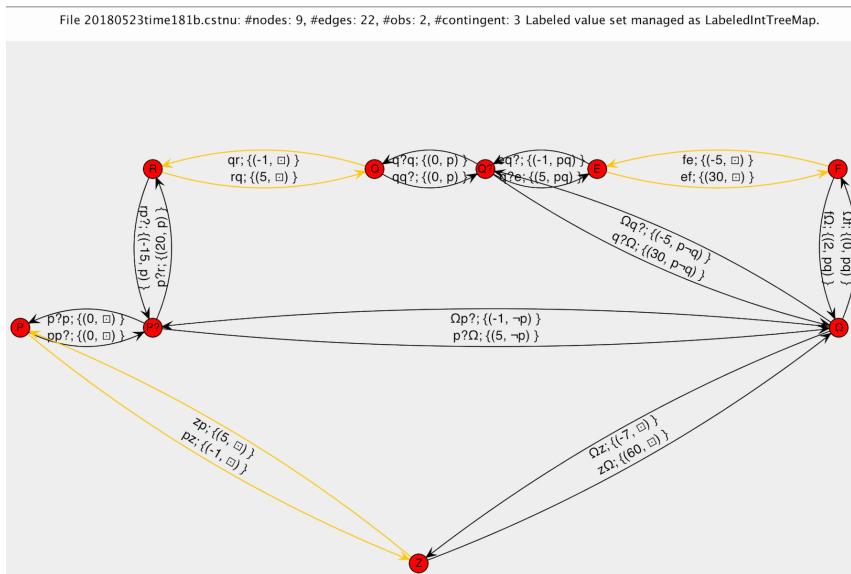
The presented model and its dynamic controllability check algorithm were implemented as a proof-of-concept prototype in the CSTNU Toolset [22]. This prototype enables users to create, edit, and load/save CSTNPSU instances using a graphical editor. A loaded CSTNPSU can then be checked for dynamic controllability. Moreover, the minimal temporal constraints between any time-point and time-point Z are shown after a successful check.

The screenshot in Figure 5 shows the CSTNU Toolset after the checking of the CSTNU depicted in Figure 1b: there are three panels, the *editor*, on the right, the *checker*, on the left, and the *command* below. Figure 5 highlights the three panels. The editor panel shows the loaded CSTNU instance while the checker panel shows the same instance after the dynamic

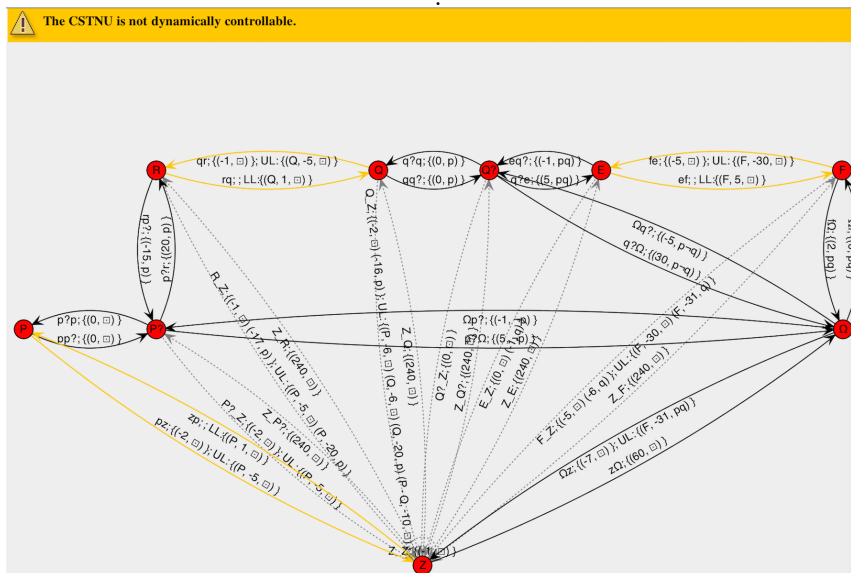
9:12 Extending CSTN with Partially Shrinkable Uncertainty



(a) Toolset screen after the check of CSTNU depicted in Figure 1b.

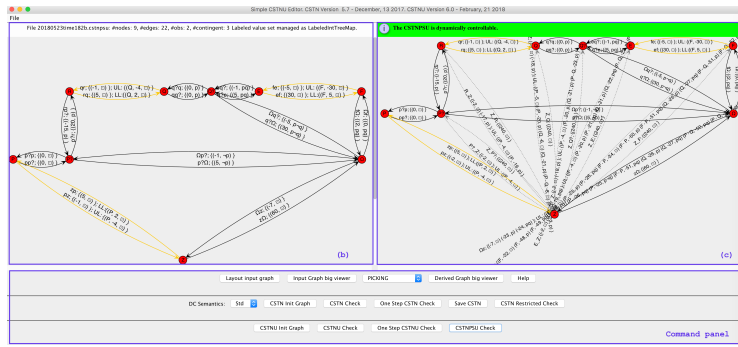


(b) Editor Detail.

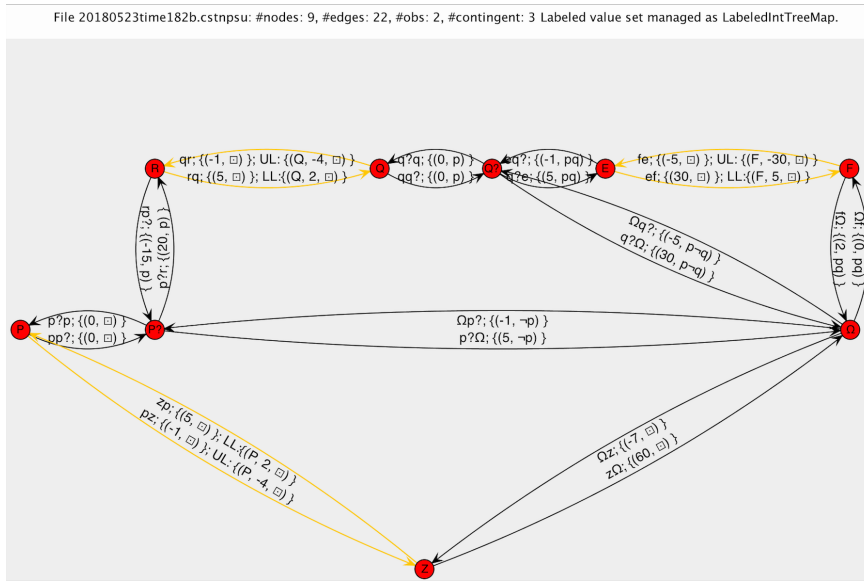


(c) Checker Detail.

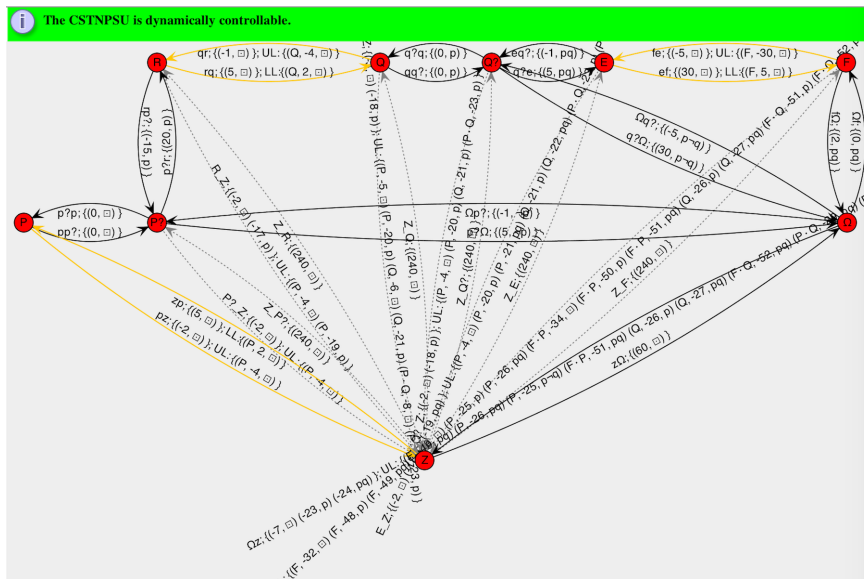
■ **Figure 5** Determining the Dynamic Controllability of CSTNU depicted in Figure 1b.



(a) Toolset screen after the check of CSTNPSU depicted in Figure 2b.



(b) Editor Detail.



(c) Checker Detail.

■ **Figure 6** Determining the Dynamic Controllability of CSTNPSU depicted in Figure 2b.

controllability check button has been clicked. For CSTNU depicted in Figure 1b, the check determines that it is not dynamically controllable. Therefore, the new constraints shown in the checker panel are the minimal number of constraints that determine a negative cycle in the instance.

The screenshot in Figure 6 shows the CSTNU Toolset after the checking of the CSTNPSU depicted in Figure 2b. In this case, the check determines that the instance is dynamically controllable and the new constraints shown in the checker panel represent the minimal distance between time-point Z and all the other time-points.

5 Conclusions

In this paper we presented a new temporal constraint model, CSTNPSU, that allows one to represent guarded links and conditions completing previous proposals dealing with either guarded links or conditions, separately. In particular, we discussed CSTNPSU features through a motivating example and, then, we introduced the concept of controllability for such networks and a related checking algorithm. Besides checking DC controllability of CSTNPSUs, such algorithm performs the minimization of constraints, in case the network is controllable. We showed that the algorithm is sound-and-complete. As for future work, we will focus on the proposal of an efficient run-time execution algorithm that will extend in an appropriate way the run-time execution algorithm for STNUPS presented in [16].

References

- 1 Arthur Bit-Monnot, Malik Ghallab, and Félix Ingrand. Which contingent events to observe for the dynamic controllability of a plan. In *Proceedings of the Twenty-Fifth Intl Joint Conf on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 3038–3044, 2016. URL: <http://www.ijcai.org/Abstract/16/431>.
- 2 Massimo Cairo and Romeo Rizzi. Dynamic controllability of conditional simple temporal networks is PSPACE-complete. In *23rd Intl Symposium on Temporal Representation and Reasoning, TIME 2016*, pages 90–99, 2016. doi:10.1109/TIME.2016.17.
- 3 Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Dynamic controllability via Timed Game Automata. *Acta Informatica*, 53(6–8):681–722, October 2016. doi:10.1007/s00236-016-0257-2.
- 4 Carlo Combi, Mauro Gambini, Sara Migliorini, and Roberto Posenato. Modelling temporal, data-centric medical processes. In *ACM Intl Health Informatics Symposium, IHI '12, Miami, FL, USA, January 28-30, 2012*, pages 141–150, 2012. doi:10.1145/2110363.2110382.
- 5 Carlo Combi, Mauro Gambini, Sara Migliorini, and Roberto Posenato. Representing business processes through a temporal data-centric workflow modeling language: An application to the management of clinical pathways. *IEEE T. Systems, Man, and Cybernetics: Systems*, 44(9):1182–1203, September 2014. doi:10.1109/TSMC.2014.2300055.
- 6 Carlo Combi, Luke Hunsberger, and Roberto Posenato. An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty - revisited. In *Agents and Artificial Intelligence - 5th Intl Conf, ICAART 2013, Revised Selected Papers*, volume 449 of *Communications in Computer and Information Science*, pages 314–331. Springer, 2014. doi:10.1007/978-3-662-44440-5_19.
- 7 Carlo Combi and Roberto Posenato. Towards temporal controllabilities for workflow schemata. In Nicolas Markey and Jef Wijsen, editors, *17th Intern. Symp. on Temporal Representation and Reasoning (TIME 2010)*, pages 129–136. IEEE Computer Society, September 2010. doi:10.1109/TIME.2010.17.

- 8 Patrick R. Conrad and Brian C. Williams. Drake: An efficient executive for temporal plans with choice. *Journal of Artificial Intelligence Research (JAIR)*, 42:607–659, 2011. doi:10.1613/jair.3478.
- 9 Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers Inc., 2003.
- 10 Luke Hunsberger and Roberto Posenato. Dynamic Controllability Checking for Conditional Simple Temporal Networks with Uncertainty: New Sound-and-Complete Algorithms based on Constraint Propagation. In Natasha Alechina, Kjetil Nørsvåg, , and Wojciech Penczek, editors, *25th International Symposium on Temporal Representation and Reasoning (TIME 2018)*, volume 120 of *LIPICs*, pages 14:1–14:17, October 2018. doi:10.4230/LIPICs.TIME.2018.14.
- 11 Luke Hunsberger and Roberto Posenato. Simpler and faster algorithm for checking the dynamic consistency of conditional simple temporal networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 1324–1330. International Joint Conferences on Artificial Intelligence Organization, July 2018. doi:10.24963/ijcai.2018/184.
- 12 Luke Hunsberger, Roberto Posenato, and Carlo Combi. The Dynamic Controllability of Conditional STNs with Uncertainty. In *Workshop on Planning and Plan Execution for Real-World Systems: Principles and Practices (PlanEx) @ ICAPS-2012*, pages 1–8, Atibaia, June 2012.
- 13 Luke Hunsberger, Roberto Posenato, and Carlo Combi. A Sound-and-Complete Propagation-Based Algorithm for Checking the Dynamic Consistency of Conditional Simple Temporal Networks. In *TIME 2015: 22nd Intl Symposium on Temporal Representation and Reasoning (TIME 2015)*, pages 4–18. IEEE, sep 2015. doi:10.1109/TIME.2015.26.
- 14 Akhil Kumar and Russell R. Barton. Controlled violation of temporal process constraints - models, algorithms and results. *Inf. Syst.*, 64:410–424, 2017. doi:10.1016/j.is.2016.06.003.
- 15 Andreas Lanz, Roberto Posenato, Carlo Combi, and Manfred Reichert. Controllability of time-aware processes at run time. In *On the Move to Meaningful Internet Systems: OTM 2013 Confs - Confederated Intl Confs: CoopIS, DOA-Trusted Cloud, and ODBASE*, volume 8185 of *LNCS*, pages 39–56. Springer, September 2013. doi:10.1007/978-3-642-41030-7_4.
- 16 Andreas Lanz, Roberto Posenato, Carlo Combi, and Manfred Reichert. Simple temporal networks with partially shrinkable uncertainty. In *Proceedings of the 6th International Conference on Agents and Artificial Intelligence (ICAART 2015)*, volume 2, pages 370–381. SciTePress, 2015. doi:10.5220/0005200903700381.
- 17 Andreas Lanz, Roberto Posenato, Carlo Combi, and Manfred Reichert. Controlling Time-Awareness in Modularized Processes. In *Enterprise, Business-Process and Information Systems Modeling, 17th Intl Conf, BPMDS 2016, 21st Intl Conf, EMMSAD 2016*, volume 248 of *Lecture Notes in Business Information Processing*, pages 157–172. Springer, 2016. doi:10.1007/978-3-319-39429-9_11.
- 18 Andreas Lanz, Manfred Reichert, and Barbara Weber. Process time patterns: A formal foundation. *Inf. Syst.*, 57:38–68, 2016. doi:10.1016/j.is.2015.10.002.
- 19 Dian Liu, Hongwei Wang, Chao Qi, Peng Zhao, and Jian Wang. Hierarchical task network-based emergency task planning with incomplete information, concurrency and uncertain duration. *Knowledge-Based Systems*, 112:67–79, 2016. doi:10.1016/j.knsys.2016.08.029.
- 20 Paul H. Morris and Nicola Muscettola. Temporal dynamic controllability revisited. In *National Conf on Artificial Intelligence (AAAI'05)*, pages 1193–1198, 2005.

- 21 Paul H. Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal uncertainty. In *Intl Joint Conf on Artificial Intelligence (IJCAI'01)*, pages 494–502. Morgan Kaufmann, 2001.
- 22 Roberto Posenato. The CSTNU toolset. version 1.23. <http://profs.scienze.univr.it/~posenato/software/cstnu>, 2018.
- 23 Ioannis Tsamardinos, Thierry Vidal, and Martha E. Pollack. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints*, 8:365–388, 2003.
- 24 Peng Yu, Brian Charles Williams, Cheng Fang, Jing Cui, and Patrik Haslum. Resolving over-constrained temporal problems with uncertainty through conflict-directed relaxation. *J. Artif. Intell. Res.*, 60:425–490, 2017. doi:10.1613/jair.5431.