Dynamic Controllability of Parameterized CSTNUs

Marco Franceschetti*
Institute of Computer Science, University of St. Gallen
St. Gallen, Switzerland
marco.franceschetti@unisg.ch

Roberto Posenato
Department of Computer Science, University of Verona
Verona, Italy
roberto.posenato@univr.it

Carlo Combi
Department of Computer Science, University of Verona
Verona, Italy
carlo.combi@univr.it

Johann Eder
Department Informatics-Systems, University of Klagenfurt
Klagenfurt, Austria
johann.eder@aau.at

ABSTRACT

A Conditional Simple Temporal Network with Uncertainty (CSTNU) models temporal constraint satisfaction problems in which the environment sets uncontrollable timepoints and conditions. The executor observes and reacts to such uncontrollable assignments as time advances with the CSTNU execution. However, there exist scenarios in which the occurrence of some future timepoints must be fixed as soon as the execution starts. We call these timepoints *parameters*. For a correct execution, parameters must assume values that guarantee the possibility of satisfying all temporal constraints, whatever the environment decides the execution time for uncontrollable timepoints and the truth value of conditions, i.e., dynamic controllability (DC). Here, we formalize the extension of the CSTNU with parameters. Furthermore, we define a set of rules to check the DC of such extended CSTNU. These rules additionally solve the problem inverse to checking DC: computing restrictions on parameter values that yield DC guarantees. The proposed rules can be composed into a sound and complete procedure.

CCS CONCEPTS

• **Information systems → Spatial-temporal systems**; • **Computing methodologies → Temporal reasoning**.

KEYWORDS

Temporal Constraint Network, Conditional, Uncertainty, Constraint-propagation, Dynamic controllability

ACM Reference Format:
Marco Franceschetti, Roberto Posenato, Carlo Combi, and Johann Eder. 2023. Dynamic Controllability of Parameterized CSTNUs. In *The 37th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27-April 2, 2023, Tallinn, Estonia.* ACM, New York, NY, USA, Article 12927.35, 9 pages. https://doi.org/10.1145/3555776.3577618

*At the time of submission for review , this author was affiliated with the Department of Informatics-Systems, University of Klagenfurt, Klagenfurt, Austria.

1 INTRODUCTION AND MOTIVATION

Temporal reasoning is an important topic in many contexts where temporal aspects of processes/data need to be suitably highlighted, represented, verified, and/or derived. While some approaches focus on the proposal and identification of computational properties of temporal reasoning approaches (e.g., temporal logic, time game automata, …), others aim to offer effective operational models for temporal reasoning (e.g., temporal networks, temporal constraint networks, …). Both approaches often propose graph-based representations of the faced aspects [2, 12, 13, 16]. For example, in [13], authors studied the satisfiability, implication, and validation problems for data dependencies through Temporal Graph Functional Dependencies, while in [16], authors proposed a graph-based algorithm for acquiring temporal constraints among relations.

Temporal constraint networks (TCN) are a formalism for representing and reasoning about temporal knowledge in time-constrained applications. TCNs encode binary temporal constraints between timepoints as linear inequalities. The first type of TCN, the Simple Temporal Network (STN), was introduced in [4] to solve the problem of verifying the *satisfiability* of a set of temporal constraints. Extensions of the STN tackle more complex problems in the presence of, e.g., uncontrollable durations (STNU, [19]), observed conditions (CSTN, [17]), or both (CSTNU, [10]). These extensions led to the definition of properties such as strong, weak, and *dynamic controllability* (DC) [8, 9, 18, 19]. In particular, DC ensures that a controller can dynamically respond to run-time observations of uncontrollable assignments of timepoints and observed conditions by assigning the controllable timepoints during the TCN execution. DC significantly differs from the satisfiability property. In parallel with game theory, verifying satisfiability can be seen as a single-player game to find an assignment of timepoints that satisfies a given set of temporal constraints. On the other hand, DC can be seen as a two-player game in which the player has a winning strategy for assigning controllable timepoints based on the observed assignments of uncontrollable timepoints and conditions chosen by the opponent so far.

Motivated by workflow time management [6] as an application of TCNs, the authors in [5] introduced the concept of parameter nodes in STNUs. Parameter nodes are STNU timepoints whose (future) value is fixed at the beginning of the STNU execution before any other timepoint. The semantics of parameter nodes is in contrast to the usual interpretation of a TCN timepoint, according to which a timepoint value is fixed when the timepoint is executed, and the fixed value is the time of execution. Parameter nodes enable the modeling of temporal problems in which the values of some future timepoints are known in advance. When an STNU with parameter nodes is executed, the assignments of the controllable (non-parameter) timepoints also depend on the values that the parameter timepoints have been assigned at the beginning. Thus, the introduction of parameter nodes redefined the DC property: the winning strategy assigning controllable timepoints is not only based on the values of the previously executed timepoints but also on the future values of parameter nodes fixed at the beginning.

Besides checking the DC of a network, parameter nodes pose an additional challenge: determining which admissible values they may be assigned before an STNU execution starts and preserving the DC of the STNU. To this end, the work in [7] introduced the P-Rules, i.e., rules for deriving the minimal bounds for parameter node values that guarantee the DC of the STNU. Each assignment of the parameter nodes satisfying these bounds represents a possible fixing of their values before the start of the STNU execution, which guarantees the DC of the network.

Although STNU is adequate to represent several problems, e.g., robotic plans, on the other hand, it is limited by the lack of semantics for conditional executions, i.e., executions that depend on observed conditions. Indeed, a broad number of applications, e.g., business processes, exhibit conditional executions and require more expressive temporal constraint networks for a formal representation and reasoning. As an example of a real temporal constraint network, we consider here an adapted excerpt from the guideline for the diagnosis and treatment of *ST-segment Elevation Myocardial Infarction (STEMI)*, published by the American College of Cardiology/American Heart Association in 2004 and updated in 2013 [14].

Example 1.1. The case starts when the patient is admitted to the hospital, and a first evaluation is done. After admission, the physician has to define the therapy according to the possible presence of hypotension. Then, the main decision has to be taken, i.e., whether to go for a fibrinolytic therapy (which is drug-based) or opt for catheter-based coronary intervention. The coronary intervention requires that the operating room is suitably prepared. At the end, a patient evaluation follows, independently of the procedure.

Figure 1 represents such an excerpt of the guidelines through a TCN. In such a network, timepoints are represented through textual labels. For example, Z represents the starting timepoint (Zero timepoint), DS represents the starting point of diuretics therapy, and so on.

Temporal constraints are represented as directed edges between timepoints, with the allowed range of distances between connected timepoints. Contingent links are represented through double-line directed edges. Such links represent that some timepoints (*contingent* timepoints) are not under the control of the system, which can only observe when they occur within the given range. For example, DE, the timepoint pointed by a contingent link, represents the end of the definition of the diuretic therapy, which may happen from 3 to 10 minutes after the start of this activity.

Not all the timepoints are always executed (i.e., happen). For example, in the case of hypotension, *condition* represented by the true literal h, the diuretics and nitrates therapy specification is not executed. Different execution paths in the network are expressed through labeled edges. As another example, the true literal f represents the *condition* that a Fibrinolytic Therapy must be executed.

A last kind of timepoint is represented through underlined textual labels. It represents some timepoint that has to be executed at a specific time distance from the starting timepoint Z, within the given range. Such timepoints are named *parameters*. This time distance has to be fixed at the beginning of the execution of the network. In the example, the parameter corresponding to *Operating Room Prepared* (*ORP*) must be specified immediately at the beginning of the execution. In this case, the range $[20, +\infty]$ requires the room not to be prepared too early with respect to the possible use of the operating room for the coronary intervention. The parameter *PMR* represents the time at which past medical records for the case must be available. Fixing it before the execution means making explicit the required time to retrieve such documents. Timepoints corresponding to *ORP* and *PMR* represent a kind of deadline that has to be shared with other external processes, which require the execution of some coordinated tasks. Thus, they must be set at the beginning of any execution of the considered network to allow the right execution of external processes that need some kind of synchronization with the execution of the considered network. For example, parameter *Operating Room Prepared* (*ORP*) must be shared with the stakeholders involved in the (complex) disinfection and sterilization of the operating room.

After defining the STEMI-related network, it is important to verify if it is possible to carry out a successful execution. We have to guarantee that as soon as we know the value of a contingent timepoint, we can move forward with the execution of the timepoints under the control of the system (controllable timepoints), satisfying all the given constraints without forbidding any possible (future) execution path, i.e., we have to guarantee the DC of the network. On the other hand, the parameter values have to be fixed before starting the network execution and must allow the following DC. In the example, we need to determine the values for parameters *ORP* and *PMR* that allow successful executions of the STEMI-related network. Thus, we need to determine adequate restrictions as done in STNUs by using the P-Rules. However, in CSTNUs, P-Rules are insufficient for this purpose, since they do not consider conditions.

Here, we propose to extend the CSTNUs with parameters and develop a new set of rules for checking the DC, overcoming the current limitations of (i) CSTNUs lacking support for parameters, and (ii) P-Rules lacking support for conditions. In general, given a network of constraints, a DC-checking algorithm determines whether the network is DC or not. We propose to solve the inverse problem with respect to parameters: given a CSTNU with parameters, determine a set of restrictions on parameters, which yield the DC of the network. The contributions of this paper are the following:

- The *parameterized CSTNU*, a new kind of temporal constraint network that is strictly more expressive than the CSTNU;
- An algorithm for parameterized CSTNUs solving the inverse problem of computing parameter restrictions yielding DC.

2 PRELIMINARIES

Since the concept of labeled constraint is important in the following definitions, let us introduce it. Given a set $\mathcal{P}$ of propositional letters, a *propositional label (p-label)* ℓ is any conjunction of literals, where

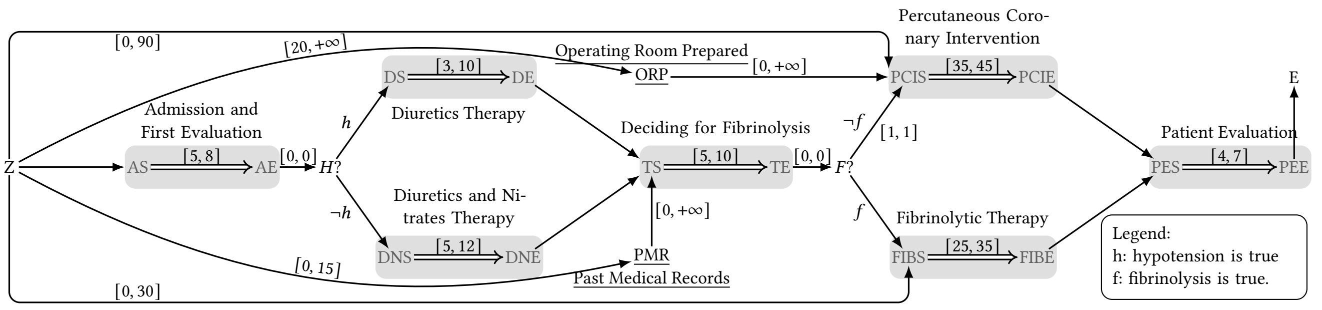

Figure 1: Temporal constraint network for a guideline excerpt for managing patients having a myocardial infarction. The contingent links are in a gray box having the associated clinical activity as a label. Each parameter also has the associated clinical name as a label. The implicit range is $[0, +\infty]$.

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. Following is the definition of CSTNU from [9].

Definition 2.1 (CSTNU). A *CSTNU* is a tuple $\langle \mathcal{T}, \mathcal{P}, L, C, \mathcal{OT}, O, \mathcal{L} \rangle$, where:

- $\mathcal{T}$ is a finite set of real-valued variables, called *timepoints*; $Z \in \mathcal{T}$ is the timepoint that occurs before any other one. For convenience, we assume that $Z = 0$.
- $\mathcal{P}$ is a finite set of propositional letters;
- $L\colon \mathcal{T} \to \mathcal{P}^*$ assigns p-labels to timepoints;
- C is a set of *labeled* constraints (*requirement links*), each with form $(Y - X \le \delta, \ell)$, where $X, Y \in \mathcal{T}$, $\delta \in \mathbb{R}$, and $\ell \in \mathcal{P}^*$; *note that if* $\delta \le 0$, *then* X *has to occur at least* $|\delta|$ *after* Y;
- $\mathcal{OT} \subseteq \mathcal{T}$ is a set of *observation* timepoints;
- $O\colon \mathcal{P} \to \mathcal{OT}$ is a bijection from propositional letters to observation timepoints;
- $\mathcal{L}$ is a set of *contingent links* each of the form (A, x, y, C), where: $A \in \mathcal{T}$, $C \in \mathcal{T} \setminus \mathcal{OT}$, $A \ne C$ are called *activation* and *contingent* timepoints, respectively; $L(A) = L(C)$; $0 < x < y < \infty$; and distinct contingent links have distinct contingent timepoints.
- For each labeled constraint $(Y - X \le v, \alpha)$, $\alpha \supset L(Y) \wedge L(X)$. This property is called *constraint label coherence* [11].
- For each literal q or $\neg q$ appearing in α, $\alpha \supset L(O(q))$. Such a property is called *constraint label honesty* [11].
- 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., $(\epsilon \le Y - O(q) \le +\infty, L(Y)) \in C$ for some $\epsilon > 0$. Such a property is called *timepoint label honesty* [11].

Contingent links have no propositional label since between an activation timepoint and the relative contingent one it is required that there is one contingent link at most. Hence, the label would be redundant. A full specification of the values of the propositions in $\mathcal{P}$ is called *scenario*, and is usually denoted by s. A full specification of contingent link durations is called *situation*, and it is usually denoted by ω. A pair of scenario and situation (s, ω) is called *drama*.

The properties *constraint label coherence, constraint label honesty*, and *timepoint label honesty* are necessary to guarantee well-defined labels and networks without label inconsistencies.

An *execution strategy* is a function that determines a schedule, i.e., a real-value assignment for the timepoints (but contingent ones) of a network. We say that a network is *executed* when a schedule for timepoints is applied. For historical reasons, given a schedule ψ and a timepoint X, we denote with $[\psi]_X$ the value assigned by ψ to X. Considering a CSTNU, a *dynamic execution strategy* is a function that determines a schedule considering the values that the environment sets to the propositional letters (*scenario*) and to the contingent timepoints (*situation*) as time passes [9].

We say that a dynamic execution strategy is *viable* when it guarantees that all relevant constraints will not be violated, no matter which scenario and situation are acquired incrementally over time. If the network admits a viable dynamic execution strategy, then the CSTNU is said *dynamically controllable (DC)*. Given a CSTNU, the *DC-checking problem* consists of verifying whether it is DC.

There are two important facts about the DC-checking problem:

(1) In [1], the authors showed that the DC-checking problem in Conditional Simple Temporal Networks (CSTNs) is PSPACE-complete; CSTNs are CSTNUs without contingent links. Thus, the DC-checking problem in CSTNUs is PSPACE-hard.
(2) In [9], 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 labels are present only on constraints because the three properties about labels in the definition allow one to ignore node labels without consequences. Ignoring node labels makes the analysis of the DC-checking problem simpler.

As proposed in [9], to solve the DC-checking problem, it is interesting to represent the network with an equivalent form called *distance graph*. Given a network instance, 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 labeled constraints/contingent ranges of the original instance [3, 9, 12]. In particular, each labeled constraint $(Y - X \le \delta, \ell)$ is represented as an *ordinary edge* $X \xrightarrow{\langle \delta, \ell \rangle} Y$, while each contingent link (A, x, y, C) between timepoints A and C is represented as two *ordinary edges*, $A \xrightarrow{\langle y, \boxdot \rangle} C$ and $C \xrightarrow{\langle -x, \boxdot \rangle} A$, representing the external bounds and two other edges, called *lower* and *upper-case edges*. Such edges are useful for representing the contingent property of contingent timepoints as temporal constraints. In particular, a *lower-case* edge, $A \xrightarrow{\langle c{:}x, \boxdot \rangle} C$, represents the property that the contingent timepoint 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 in [12].

Figure 2 represents the distance graph for the STEMI-related network in Figure 1. Hereinafter, we indifferently refer to timepoints as nodes, and to constraints as edges.

3 DYNAMIC CONTROLLABILITY OF PARAMETERIZED CSTNUS

To reason about temporal problems in which some timepoints are fixed at the beginning of the execution requires extending the CSTNUs with *parameters*. Thus, we first define parameterized CSTNUs; then, we tackle two fundamental problems: (1) how to check the DC property of a parameterized CSTNU; (2) how to determine all admissible parameter values that do not invalidate the DC property.

3.1 Parameterized CSTNUs

The difference between parameterized CSTNUs and traditional CSTNUs [9] lies in the partitioning of the set of timepoints into *controllable*, *uncontrollable*, *observation*, and (the newly introduced) *parameters*. Parameters are non-observation timepoints whose value is set at the beginning of the execution.

Definition 3.1 (Parameterized CSTNU). A *parameterized CSTNU* is a tuple $\langle \mathcal{T}, \mathcal{P}, L, C, \mathcal{OT}, \mathcal{O}, \mathcal{L}, \mathcal{PT}, \mathcal{PC}\rangle$, where:

- $\langle \mathcal{T}, \ldots, \mathcal{L}\rangle$ is a CSTNU exhibiting label coherence, label honesty, and timepoint label honesty, as in Definition 2.1;
- $\mathcal{PT} \subseteq \mathcal{T}$ is a set of timepoints, called *parameters*, whose values are set by the environment in advance. A parameter $X \in \mathcal{PT}$ is denoted as $\underline{X}$.
- $\mathcal{PT} \cap \mathcal{OT} = \emptyset$,
- $\mathcal{PC} \subseteq C$ is a set of *parameter constraints*, i.e., constraints of the form $(Y - X \le \delta, \ell)$, where $\{X,\ Y\} \cap \mathcal{PT} \neq \emptyset$.

In general, setting a value for a timepoint corresponds to fixing its time distance from Z, i.e., introducing two constraints with no propositional label between the timepoint and Z. Setting values to *all* parameters of a parameterized CSTNU is an *instantiation*:

Definition 3.2 (Instantiation). Let $N = \langle \mathcal{T}, \mathcal{P}, L, C, \mathcal{OT}, \mathcal{O}, \mathcal{L}, \mathcal{PT}, \mathcal{PC}\rangle$ be a parameterized CSTNU, with $\mathcal{PT} = \{\underline{X_1}, \ldots, \underline{X_m}\}$. An *instantiation* of N is obtained by adding to N additional parameter constraints $(\underline{X_i} - \mathrm{Z} \le x_i, \boxdot)$ and $(\mathrm{Z} - \underline{X_i} \le -x_i, \boxdot)$ for each $\underline{X_i} \in \mathcal{PT}$, where $x_i \in \mathbb{R}_{\ge 0}$. We call $I = (x_1, \ldots, x_m)$ a *parameter instantiation* for N, and denote by N_I the corresponding instantiation of N.

An instantiation of a parameterized CSTNU is a traditional CSTNU, where each parameter is a timepoint connected to Z with a pair of constraints that fix its value. For example, an instantiation of the network in Figure 2 fixing the values of *ORP* and *PMR* to 25 and 10, respectively, is the same network with the addition of edges $\mathrm{Z} \xrightarrow{25} ORP$, $ORP \xrightarrow{-25} \mathrm{Z}$, $\mathrm{Z} \xrightarrow{10} PMR$, $PMR \xrightarrow{-10} \mathrm{Z}$.

In the following, we discuss the DC of parameterized CSTNUs and how to determine all the admissible parameter instantiations yielding the DC property by computing parameter constraints.

3.2 Dynamic Controllability

The notions of execution strategy, dynamic execution strategy, and viable execution strategy for a CSTNU need to be extended in the case of parameterized CSTNUs to cope with parameters (we put in italics the extension with respect to the corresponding definition given in Section 2).

An **execution strategy** is a function that determines a schedule for the network timepoints *but parameter timepoints and contingent ones.* A **dynamic execution strategy** is a function that determines a schedule *considering the values that the environment sets to the parameter timepoints at the start of the execution*, the propositional letters, and the contingent timepoints as time passes. Given *a set of values for parameter timepoints that does not violate any existing constraint*, we say that a dynamic execution strategy is **viable** when it guarantees that all relevant constraints will not be violated no matter which truth values for propositions (*scenario*) and durations for contingent links (*situation*) are incrementally acquired over time.

A parameterized CSTNU can have different viable execution strategies according to the given set of parameter values. Therefore, when there exists at least one set of parameter values such that the network admits a viable dynamic execution strategy, then the parameterized CSTNU is said **dynamically controllable (DC).**

Like for CSTNUs, the **DC-checking problem** for a parameterized CSTNU consists of verifying whether it is DC. Extending the classification proposed by Vidal and Fargier in [19], the *DC-checking problem* can be viewed as the problem to verify if the network is 1) *strongly* controllable with respect to possible scenarios and situations when parameters are considered, and 2) *dynamically* controllable with respect to possible scenarios and situations when only controllable timepoints are considered. The DC-checking problem for parameterized CSTNUs is still a PSPACE-hard problem since a CSTNU is a parameterized CSTNU with no parameters. For the parameterized CSTNU, the following problems are interesting:

(1) the DC-checking problem for an instantiation of a parameterized CSTNU,
(2) the DC-checking problem for a parameterized CSTNU with given parameter constraints,
(3) the problem of finding all parameter values yielding DC for a parameterized CSTNU.

Example 3.3. Here is an example for each proposed problem:

(1) *Is the instantiation of the network in Figure 2 resulting from the setting of $ORP = 25$ and $PMR = 10$ DC?*
Checking DC of an instantiation of a parameterized CSTNU is straightforward since the instantiation is a traditional CSTNU composed of all the timepoints and the respective constraints, and the parameter nodes and their respective constraints (cf. Definition 3.2). Thus, applying a CSTNU DC-checking algorithm such as the one proposed in [9] is sufficient for verifying whether the instantiation is DC.
(2) *Is the network in Figure 2 DC when there are the parameter constraints $ORP \xrightarrow{\langle -20, \neg f\rangle} \mathrm{Z}$, $PCIS \xrightarrow{\langle 0, \neg f\rangle} ORP$, $\mathrm{Z} \xrightarrow{\langle 15, \boxdot\rangle} PMR$,*

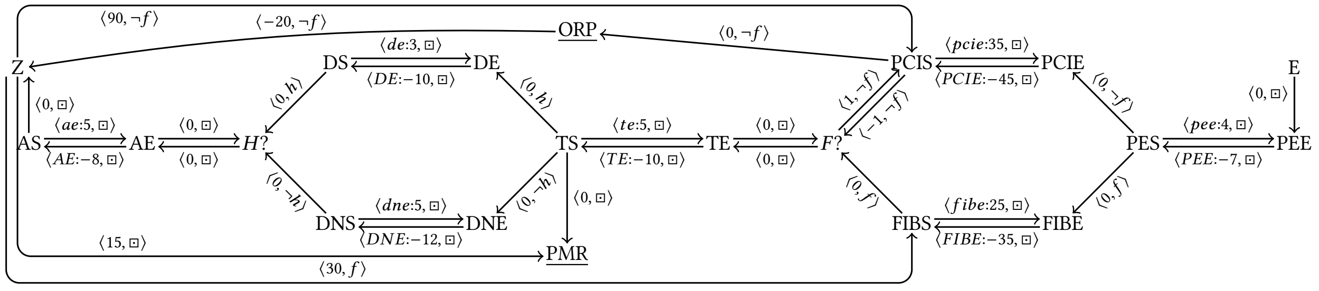

Figure 2: Distance graph for the network in Figure 1.

$TS \xrightarrow{\langle 0, \boxdot\rangle} PMR$? Checking the DC of a parameterized CSTNU with given parameter constraints is not trivial due to the different nature of parameter nodes and timepoints. In this case, checking DC requires considering all possible instantiations, and verifying that each instantiation is DC. A possible solution to this problem is to proceed in analogy to the approach shown in [5] for the case of STNUs with parameter nodes, applying an existing CSTNU DC-checking algorithm.

(3) *Given the parameterized CSTNU in Figure 2, which are all the admissible values for ORP and PMR that yield DC?*
Given a parameterized CSTNU, it is interesting to determine *all* the possible values that can be assigned to its parameters, while ensuring that the DC property is preserved. Indeed, this is an inverse problem to DC-checking, and it consists of determining the most general set of parameter constraints such that including this set into the set $\mathcal{PC}$ of the parameterized CSTNU (cf. Definition 3.1) keeps the network DC.

The third problem is the most challenging. In the next sections, we propose a technique to solve it.

3.3 Admissibility Set

To determine all the possible values that can be assigned to a parameter, let us start defining the concept of *admissibility set.*

Definition 3.4 (Admissibility Set). Let $N = \langle \mathcal{T}, \mathcal{P}, L, C, \mathcal{OT}, \mathcal{O}, \mathcal{L}, \mathcal{PT}, \mathcal{PC}\rangle$ be a parameterized CSTNU, with $\mathcal{PT} = \{\underline{X_1}, \dots, \underline{X_m}\}$. Let $\mathcal{A}$ be a set of *minimal* constraints over $\{Z, \underline{X_1}, \dots, \underline{X_m}\}$ [4]. We call $\mathcal{A}$ an *admissibility set* for $\mathcal{PT}$ if each solution of $\mathcal{A}$ is a parameter instantiation I for N such that N_I is dynamically controllable.

Given two constraints $(Y - X \le \delta')$ and $(Y - X \le \delta)$, we say that $(Y - X \le \delta')$ is more relaxed than $(Y - X \le \delta)$ if $\delta < \delta'$. We call an admissibility set $\mathcal{A}$ *maximal* if, for any other admissibility set $\mathcal{A}'$ and any two parameters $\underline{X}, \underline{Y}$, there is no constraint $(\underline{Y} - \underline{X} \le \delta')$ in $\mathcal{A}'$ that is more relaxed than the constraint $(\underline{Y} - \underline{X} \le \delta)$ in $\mathcal{A}$.

The admissibility set contains only constraints that must be satisfied by a parameter instantiation independent of any contingent duration and any condition—propositional letter—which are only revealed when executing the network.

It is easy to see that an STN is sufficient to represent such a set of constraints since an STN is a CSTNU without contingent links and observation timepoints. The STN for the maximal admissibility set is the unique STN that yields the largest set of solutions, i.e., of possible parameter instantiations. Thus, any solution for the STN encoding the admissibility set represents a fix of the parameter values that preserves the DC property of the parameterized CSTNU.

By computing the maximal admissibility set, one can compute the parameter instantiations that guarantee executions without constraint violations. In the following, we propose a procedure that, given a parameterized CSTNU, checks whether it is DC and derives the STN encoding the maximal admissibility set for its parameters.

3.4 Constraint Propagation Rules

A possible technique for solving the DC-checking problem is to determine an equivalent network propagating the constraints. If such a process finds a negative cycle, the network is not DC; otherwise, there is at least one solution in the resulting network [9]. Then, such a solution can be calculated incrementally, reacting to the outcomes of observations and execution time of contingent timepoints.

Propagating a constraint consists in combining two constraints by a proper rule in order to obtain an equivalent explicit constraint. In [9], the authors proposed a set of propagation rules that combine into a sound-and-complete algorithm to solve the DC-checking problem for CSTNUs. Briefly, the algorithm consists of a loop that applies rules until no new constraints are discovered; the computational complexity of the algorithm is $O(M|\mathcal{T}|^2 3^{|\mathcal{P}|} 2^{|\mathcal{L}|})$, with M the maximum absolute value of any negative weight in the graph.

Here, we propose an extension of such a set of rules for managing two new aspects that are not considered in [9]: the presence of parameters, and the determination also of upper bounds for the execution of timepoints. Determining upper bounds is necessary to find admissible ranges for the parameters. The new rule set is depicted in Table 1. Rules can be categorized into two main types.

The first type consists of *lower-bound rules*, i.e. rules for finding lower bounds to the execution times of non-contingent and parameter timepoints. Rules A, B, C, D, zqR$_0$, and zqR$_3^*$, are the ones in [9] adjusted for the possible presence of parameter nodes; rules A-Parameter and B-Parameter are new and manage the propagation of constraints involving parameters. These two last rules are stronger than the others of the same type when two timepoints are parameter ones or Z because they determine lower bounds that must be valid in any possible drama (strong validity).

The second type consists of *upper-bound rules* that are necessary to find upper bounds to the execution times of controllable and parameter timepoints. All such rules are new and have never been proposed in previous work because they are not necessary to determine the DC of CSTNUs. At the same time, they are necessary to determine a proper range of possible values for each parameter.

We do not describe each rule in detail, but we specify two aspects: the generated edges may have labels with $\aleph$ (multiple *upper-case* letters) or may have q-labels (i.e., labels with a third kind of literal); and many of the rules can propagate such labeled values.

As regards $\aleph$, it is composed by the conjunction of zero or more *upper-case* letters, each representing a contingent timepoint. An $\aleph$ represents the possibility that all represented contingent time points occur at their maximum allowed time. $\mathbb{A}^*$ is the set of all possible conjunct-upper-case letters. An edge $Y \xrightarrow{\langle\aleph:-\delta,\ell\rangle} X$ having a non-empty $\aleph = \{C_1, C_2, \dots\}$ represents a *wait* constraint that is meaningful only in a DC context: after the execution of X, if none of the contingent timepoints $C_i \in \aleph$ occurs before $X+\delta$, the system must wait such a time before executing Y; conversely, if any C_i occurs before $X+\delta$, the constraint is satisfied, hence Y can be executed (assuming that Y was bound only by this constraint).

As regards q-labels, if $q \in \mathcal{P}$, then $?q$ is a *q-literal*. $?q$ is $\top$ only when the value of q is not set, i.e., *unknown*. Unknown literals are necessary for propagating constraints that must be satisfied for any possible value of propositions present in their labels. The propositional label concept is extended to q-literals as follows: *"a q-label is a conjunction of literals and/or q-literals"*. $\mathcal{Q}^*$ is the set of all q-labels; it holds $\mathcal{P}^* \subset \mathcal{Q}^*$. Representing the conjunction of literals and q-literals requires the $\star$ operator. Informally, if a constraint has label q, and another one has label $\neg q$, then *both* constraints must hold as long as q is *unknown*, which is represented by $q \star \neg q = ?q$ [9]. A constraint propagation rule is sound if whenever a viable and dynamic execution strategy σ satisfies the preexisting edge(s) in that rule, σ must also satisfy the edge generated by that rule. The rules in Table 1 generate edges pointing at Z or at a parameter—which represent lower-bound constraints with respect to Z or the parameter—and edges leaving from Z or a parameter—which represent upper-bound constraints with respect to Z or the parameter.

In [9], the authors formalized the concept of *satisfying a lower-bound constraint* with respect to Z and showed that rules A, B, C, D, zqR$_0$, and zqR$_3^*$ in Table 1 are sound. Since the value of a parameter must be set before the start of a network execution, the derived constraints involving such a parameter with other parameter time-point or Z must be satisfied before knowing any possible scenario or any contingent timepoint occurrence. Thus, any propagation rule either involving two parameter nodes as the starting and the ending one, respectively, or involving a parameter node and Z (always as starting/ending nodes) has to derive a value that must hold in any scenario (i.e., the scenario label must be $\boxdot$) and any situation (i.e., if there is a conjunct-upper-case label, it must be removed).

The formalization of the concept *satisfying a lower/upper-bound constraint* with respect to a parameter can be adapted as follows. The following definition refers to a π-execution strategy: since its formalization is out of the scope of this paper, we refer the reader to [9] for further details. In brief, a π-execution strategy is an execution strategy specifying both a schedule of timepoints and an order of dependency between observation timepoints.

Definition 3.5 (Satisfy a Lower-Bound Constraint with respect to a parameter). A *π-execution strategy σ satisfies the lower-bound*

Rule	Conditions Pre-existing and Generated Edges
(A)	Apply if $u+v<0$ and $\gamma := \alpha\beta \in \mathcal{P}^*$. If $X \in \mathcal{PT}$, set $\aleph := \emptyset$ and $\gamma := \boxdot$ $\begin{tikzcd} Z & Y \arrow[l, "{\langle\aleph:v,\beta\rangle}"'] & X \arrow[l, "{\langle u,\alpha\rangle}"'] \arrow[ll, bend left, "{\langle\aleph:u+v,\gamma\rangle}"] \end{tikzcd}$
(A-Parameter)	Apply if $\gamma := \alpha\beta \in \mathcal{P}^*$. If $X \equiv Z \vee X \in \mathcal{PT}$, set $\gamma := \boxdot$. $\begin{tikzcd} \underline{P} & Y \arrow[l, "{\langle v,\beta\rangle}"'] & X \arrow[l, "{\langle\aleph:u,\alpha\rangle}"'] \arrow[ll, bend left, "{\langle\aleph:u+v,\gamma\rangle}"] \end{tikzcd}$
(B)	Apply if $x+v<0$, $C \notin \aleph$, and $\alpha \in \mathcal{P}^*$ $\begin{tikzcd} Z & C \arrow[l, "{\langle\aleph:v,\alpha\rangle}"'] & A \arrow[l, "{\langle c:x,\boxdot\rangle}"'] \arrow[ll, bend left, "{\langle\aleph:x+v,\alpha\rangle}"] \end{tikzcd}$
(B-Parameter)	Apply if $\alpha \in \mathcal{P}^*$ $\begin{tikzcd} \underline{P} & C \arrow[l, "{\langle v,\alpha\rangle}"'] & A \arrow[l, "{\langle c:x,\boxdot\rangle}"'] \arrow[ll, bend left, "{\langle x+v,\alpha\rangle}"] \end{tikzcd}$
(C)	Apply if $-y+v<0$ and $\alpha \in \mathcal{P}^*$ $\begin{tikzcd} Z & A \arrow[l, "{\langle\aleph:v,\alpha\rangle}"'] & C \arrow[l, "{\langle C:-y,\boxdot\rangle}"'] \arrow[ll, bend left, "{\langle C\aleph:-y+v,\alpha\rangle}"] \end{tikzcd}$
(D)	$\beta,\gamma \in \mathcal{Q}^*$. Apply if $C \notin \aleph\aleph_1$. Set $m := \max\{v, w-x\}$ $\begin{tikzcd} Y \arrow[r, "{\langle C\aleph:v,\beta\rangle}", "{\langle\aleph\aleph_1:m,\beta\star\gamma\rangle}"'] & Z & A \arrow[l, "{\langle\aleph_1:w,\gamma\rangle}"'] \arrow[r, "{\langle c:x,\boxdot\rangle}"] & C \end{tikzcd}$
(zqR$_0$)	$\beta \in \mathcal{Q}^*$ and $\tilde{r} \in \{r, \neg r, ?r\}$. Apply if $w<0$. $\begin{tikzcd} Z & R? \arrow[l, "{\langle\aleph:w,\beta\tilde{r}\rangle}"', "{\langle\aleph:w,\beta\rangle}"] \end{tikzcd}$
(zqR$_3^*$)	$\beta,\gamma \in \mathcal{Q}^*$ and $\tilde{r} \in \{r, \neg r, ?r\}$. Apply if $w<0$. $\begin{tikzcd} Y \arrow[r, "{\langle\aleph:v,\beta\tilde{r}\rangle}", "{\langle\aleph\aleph_1:\max\{v,w\},\beta\star\gamma\rangle}"'] & Z & R? \arrow[l, "{\langle\aleph_1:w,\gamma\rangle}"'] \end{tikzcd}$
(A$^+$)	Apply if $\gamma := \alpha\beta \in \mathcal{P}^*$. If $X \in \mathcal{PT}$, set $\gamma := \boxdot$ $\begin{tikzcd} Z \arrow[r, "{\langle v,\beta\rangle}"] \arrow[rr, bend right, "{\langle u+v,\gamma\rangle}"'] & Y \arrow[r, "{\langle u,\alpha\rangle}"] & X \end{tikzcd}$
(A$^+$-Parameter)	Apply if $\gamma := \alpha\beta \in \mathcal{P}^*$. If $X \equiv Z \vee X \in \mathcal{PT}$, set $\gamma := \boxdot$. $\begin{tikzcd} \underline{P} \arrow[r, "{\langle v,\beta\rangle}"] \arrow[rr, bend right, "{\langle u+v,\gamma\rangle}"'] & Y \arrow[r, "{\langle\aleph:u,\alpha\rangle}"] & X \end{tikzcd}$
(B$^+$)	Apply if $\alpha \in \mathcal{P}^*$ $\begin{tikzcd} Z \arrow[r, "{\langle v,\alpha\rangle}"] \arrow[rr, bend right, "{\langle v-y,\alpha\rangle}"'] & C \arrow[r, "{\langle C:-y,\boxdot\rangle}"] & A \end{tikzcd}$
(B$^+$-Parameter)	Apply if $\alpha \in \mathcal{P}^*$ $\begin{tikzcd} \underline{P} \arrow[r, "{\langle v,\alpha\rangle}"] \arrow[rr, bend right, "{\langle v-y,\alpha\rangle}"'] & C \arrow[r, "{\langle C:-y,\boxdot\rangle}"] & A \end{tikzcd}$
(zR$_1$)	$\beta \in \mathcal{P}^*$, $\tilde{r} \in \{r, \neg r\}$ Apply if $w \geq 0$. $\begin{tikzcd} Z \arrow[r, "{\langle w,\beta\tilde{r}\rangle}", "{\langle w,\beta\rangle}"'] & R? \end{tikzcd}$
(zR$_2$)	$\beta,\gamma \in \mathcal{P}^*$, $\tilde{r} \in \{r, \neg r\}$. Apply if $w<0$, $v<-w$. $\begin{tikzcd} Y & Z \arrow[l, "{\langle v,\beta\tilde{r}\rangle}"', "{\langle v,\beta\gamma\rangle}"] & R? \arrow[l, "{\langle w,\gamma\rangle}"'] \end{tikzcd}$

Z; $A, C, X, Y \in \mathcal{T}$; C is a cont. node; $\underline{P} \in \mathcal{PT}$; $R? \in \mathcal{OT}$; $\aleph, \aleph_1 \in \mathbb{A}^*$.

Table 1: Constraint-propagation rules for parameterized CSTNU. Rules and conditions in blue are the new ones.

constraint $(\underline{P} - Y \leq -\delta, \boxdot)$, represented by $\underline{P} \xleftarrow{\langle -\delta, \boxdot \rangle} Y$, if, for each drama (s, ω), it holds $[\psi]_Y \geq [\psi]_{\underline{P}} + \delta$, where $(\psi, \pi) = \sigma(s, \omega)$.

Definition 3.6 (Satisfy a Upper-Bound Constraint with respect to a parameter). A π-execution strategy σ *satisfies the upper-bound constraint* $(Y - \underline{P} \leq \delta, \boxdot)$ represented by the edge from $\underline{P}$ to Y labeled by $\langle \delta, \boxdot \rangle$, where $\underline{P}$ is a parameter if, for each drama (s, ω), it holds $[\psi]_Y \leq [\psi]_{\underline{P}} + \delta$, where $(\psi, \pi) = \sigma(s, \omega)$.

We now prove the soundness of the remaining rules in Table 1.

Lemma 3.7 (A-Parameter is sound). *Rule* A*-Parameter in Table 1 is sound.*

Proof Sketch. When X is neither a parameter timepoint nor Z, the proof is equal to the soundness-proof of rule A [9]. Otherwise, the rule determines the same value but preserves neither the propositional label $\alpha\beta$ nor the possible $\aleph$. A soundness proof requires showing that if both the constraint between Y and $\underline{P}$ and the constraint between Z and Y hold (antecedents), also the new constraint between Z and $\underline{P}$ holds (consequence). The removal of the propositional label $\alpha\beta$ and (possible) $\aleph$ makes the new constraint present in more scenarios, and therefore the soundness is guaranteed. □

Lemma 3.8 (B-Parameter and B$^+$-Parameter are sound). *Rule* B*-Parameter and* B$^+$*-Parameter in Table 1 are sound.*

Proof Sketch. B-Parameter is the extension of the P-Lower rule [7] for considering also the scenarios. Since it does not alter the considered scenario, the soundness proof in [7] is still valid. The same holds for B$^+$-Parameter because it is an extension of the P-upper rule presented and proven to be sound in STNU [7]. □

The proofs of soundness of rules A$^+$ and B$^+$ are analogous to the corresponding rules A and B while the soundness of zR$_1$ and zR$_2$ are straightforward and, therefore, omitted.

Lemma 3.9 (A$^+$-Parameter is sound). *Rule* A$^+$*-Parameter in Table 1 is sound.*

Proof Sketch. When X is neither a parameter timepoint nor Z, then $\aleph$ is empty, and the form of the rule is equal to the form of rule A$^+$, but the starting timepoint is a parameter timepoint instead of Z. Since the soundness proof of A$^+$ does not use the fact that the starting point is Z, the soundness proof for such a case is the same.

When X is either a parameter timepoint or Z, $\aleph$ cannot be empty. In both cases, $\aleph$ is not added to the resulting edge because the new edge has to represent the constraint in the worst case (strong controllability). Note that this stronger requirement implies the soundness of the rule. □

The proposed rules can be combined into a constraint-propagation procedure for checking the DC property and deriving the maximal admissibility set:

Theorem 3.10. *Let* $N = \langle \mathcal{T}, \mathcal{P}, L, C, \mathcal{OT}, \mathcal{O}, \mathcal{L}, \mathcal{PT}, \mathcal{PC} \rangle$ *be a parameterized CSTNU. Let* N' *be the CSTNU resulting from a successful constraint propagation in* N *with the rules in Table 1, that is, no new edges can be derived, and no negative cycle exists in* N'*. Let* $\mathcal{S}$ *be the STN formed by* $\{Z\} \cup \mathcal{PT}$ *and the respective constraints in* N'*. Then,* N' *is DC, and* $\mathcal{S}$ *encodes the maximal admissibility set for* $\mathcal{PT}$*.*

Proof. We prove as follows: since the rules of [9] are also in the proposed rule set, a full propagation with the rule set checks whether the CSTNU is DC at all. Then, we show that the rule set is sufficient to derive any constraint involving a parameter that explicitly or implicitly holds. We argue that the rule set cares for deriving constraints making parameters independent of any *situation* and any *scenario*. The projection of the graph determined by the DC-checking algorithm on Z, the parameters, and the edges between them forms an STN. Such an STN has the property that any valid instantiation of it yields a set of values for which the parameterized CSTNU is DC. As the edges produced by the rules are both necessary and sufficient for making the parameters independent, the procedure derives the most general specification of admissible parameter values, i.e., the maximal admissibility set.

(1) The DC property of the CSTNU obtained from N by making parameters controllable timepoints is checked by the procedure as all rules of the procedure in [9] are also in the proposed rule set. If such a CSTNU is not DC, it is not possible to dynamically assign parameters in response to observed contingent durations and conditions so that all constraints can be met. If a dynamic assignment is not possible, then also an assignment at the beginning of the execution is not possible, and the parameterized CSTNU is not DC.
(2) The application of the rules to derive lower bounds for non-parameter timepoints (rules A, B, C, D, zqR$_0$, and zqR$_3^*$) ends when no new edge like $X \xrightarrow{\langle \aleph:v, \alpha \rangle} Z$ can be introduced. In the end, each edge $X \xrightarrow{\langle \aleph:v, \alpha \rangle} Z$ is the strictest lower bound for X. For a given $\aleph$, an edge $X \xrightarrow{\langle \aleph:v, \alpha \rangle} Z$ is the strictest if for each other $\langle \aleph:v', \alpha' \rangle$ with $\alpha \rightarrow \alpha'$, it holds that $v < v'$. In [9], the authors show the completeness of rules A, B, C, D, zqR$_0$, and zqR$_3^*$ for computing a dynamic *early execution strategy*. Thus, they derive the strictest lower bounds for timepoints for each drama.
(3) Rules A$^+$, B$^+$, zR$_1$, zR$_2$ are sound rules to derive upper-bounds for non-parameter timepoints. It is easy to see that these rules are symmetric to rules A, B, zqR$_0$, and zqR$_3^*$. The completeness for determining a dynamic *late execution strategy* can be shown in a similar way to the proof of completeness for the rules deriving lower bounds. So, the application of rules A$^+$, B$^+$, zR$_1$, and zR$_2$ is repeated until they derive the strictest upper bounds for timepoints for each drama.
(4) Rules A-Parameter, B-Parameter, A$^+$-Parameter, B$^+$-Parameter are an extension of rules A, B, A$^+$, and B$^+$, respectively, where Z is replaced by a parameter. Thus, they are also sound and complete rules, which determine, with an iterated application, the strictest constraints between pairs of parameters and between any parameter and Z.
(5) A parameter has to be independent of any observed duration (any situation) and any observed condition (any scenario). The two constraints between a parameter $\underline{P}$ and Z (or any other parameter) must be satisfiable under all possible drama. This can be guaranteed by requiring that any constraint between $\underline{P}$ and any timepoint X when propagated to Z (or from Z) becomes a constraint without conditions and wait, i.e., become the strongest version of the constraint.

(6) An assignment of a timepoint X restricts the admissible instantiations of $\underline{P}$ only if a constraint between X and $\underline{P}$ can be derived. Also, an observed condition can only restrict the possible instantiations of $\underline{P}$ if $\underline{P}$ participates in a conditional constraint. Thus, it is sufficient to check (a) for each edge between $\underline{P}$ and a contingent timepoint, and (b) for each conditional edge involving $\underline{P}$, whether the values for $\underline{P}$ depend on situations or scenarios.

(7) The case Item 6(a) is covered by rules B-Parameter and B$^+$-Parameter, which are proved to be sound in Lemma 3.8. For each edge from contingent C to parameter $\underline{P}$ with value $\langle v, \alpha\rangle$, resp. from $\underline{P}$ to C with value $\langle v, \alpha\rangle$, rule B-Parameter, resp. B$^+$-Parameter, derives an edge from A to $\underline{P}$ with value $\langle x + v, \alpha\rangle$, resp. from $\underline{P}$ to A with value $\langle v - y, \alpha\rangle$, where A is the activation timepoint for C. This derived edge represents a constraint such that any instantiation of $\underline{P}$ satisfying this constraint also satisfies the original constraint between C and $\underline{P}$, no matter how long the contingent link lasts.

(8) The above Item 6(b) is covered by rules A-Parameter and A$^+$-Parameter, which are proved to be sound in Lemma 3.7 and Lemma 3.9. Any possible edge between Z and a parameter represents a lower/upper bound for the parameter. Generally, such an edge can have a labeled value in which there may be a label (representing the scenarios in which the constraints must hold) and an upper-case label, representing which contingent timepoints must be considered. Since the value for the parameter must be set before the execution of the network, all possible lower/upper bounds must be guaranteed to be satisfied in any case. Hence, it has to hold unconditionally in any scenario. The same applies to constraints between any pairs of parameters $\underline{P}$ and $\underline{P'}$. Therefore, any edge between a parameter and Z or another parameter must be unconditional. With rule A-Parameter, resp. A$^+$-Parameter, an edge with value $\langle u + v, \alpha\rangle$ is introduced from parameter $\underline{P}$ to timepoint X, resp. from timepoint X to parameter $\underline{P}$. With full propagation, such an edge is eventually introduced to, resp. from, a parameter $\underline{P'}$, and timepoint Z when X = Z.

(9) A full constraint propagation determines a set of edges between Z and parameters and between pairs of parameters. These edges represent non-contingent constraints that need to be fulfilled by the parameter instantiations for the CSTNU to be DC. Thus, the projection of Z and all parameters, along with these derived edges between them, form an STN $\mathcal{S}$. Hence, any solution to $\mathcal{S}$ represents an admissible instantiation of parameters, that is, an admissibility set for $\mathcal{PT}$. Since the rule set derives the strictest such edges, it is not possible to relax any derived edge without violating the DC property of the CSTNU. Thus, we conclude that the admissibility set encoded by $\mathcal{S}$ is also the maximal.

□

A full constraint propagation using the rules in Table 1 does not only solve the DC-checking problem for a parameterized CSTNU but also derives the maximal admissibility set for its parameters.

Figure 3 shows the STN composed of parameter nodes *ORP* and *PMR* and Z found performing the DC-check on the network in Figure 2. The STN encodes the maximal admissibility set $\mathcal{A} = \{20 \leq ORP \leq 31, 0 \leq PMR \leq 15, PMR \leq ORP - 5, ORP \leq PMR + 31\}$. Any instantiation of *ORP* and *PMR* that satisfies all the constraints in $\mathcal{A}$ is an admissible assignment of the parameters—before executing the parameterized CSTNU—which allows dynamically controllable executions. On the contrary, any instantiation that does not satisfy $\mathcal{A}$ might lead to executions in which some temporal constraint is not satisfied.

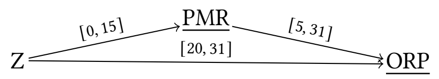

Figure 3: The STN representing the maximal admissibility set for the parameterized CSTNU depicted in Figure 2

Regarding computational complexity, the original CSTNU DC-checking complexity is determined assuming a propagation of all possible labeled values present in each edge to all other edges. If a rule determines a value already present, the value is ignored; hence no significant computational cost occurs. Such an analysis is still valid for the new algorithm even with eight new rules (the blue ones in Table 1); thus, the complexity is still $O(M|\mathcal{T}|^2 3^{|\mathcal{P}|} 2^{|\mathcal{L}|})$.

As a proof of concept, we extended the open-source CSTNU tool [15] to check the DC of the parameterized CSTNU and find the STN representing the maximal admissibility set. Our implementation is available as open-source code [15]. Figure 4 depicts a screenshot of the application after a successful check of the parameterized CSTNU in Figure 2. On the left side, there is the input parameterized CSTNU. On the right side, the instance after a successful DC checking. The edges forming the STN of the parameter nodes are magenta.

4 DISCUSSION AND CONCLUSION

CSTNUs are an established formalism representing temporal properties and requirements of dynamic models (e.g., minimum time spans between events). Frequently, such models feature parameters to communicate temporal properties or requirements for the execution. These properties and requirements are set before executing the model and may decide the values for specific timepoints in advance. With parameterized CSTNU, we can represent such models with parameter nodes and compute their maximal admissibility set.

Revisiting Example 1.1, parameters represent an agreement with respect to external actors, managing the room preparation and the provision of medical records. Establishing the specific execution time of the corresponding parameters has to be done before dealing with the specific patient, i.e., the values of parameter nodes *ORP* and *PMR* have to be fixed before the execution of the parameterized CSTNU starts. With the proposed algorithm, we can verify that the STEMI-related network is dynamically controllable and derive the STN depicted in Figure 3. It represents the admissible ranges for setting the parameters' values before executing the STEMI-related network. In the example, it means that the operating room has to be prepared between 20 and 31 minutes after the start Z of the network. Once we fix the time for the operating room to be prepared, we can set the time for having the past medical records, which has to be at least 5 minutes before the room is prepared. Parameter values can be fixed in any order before executing the network.

Moreover, the allowed ranges of the derived STN may be stricter than the corresponding ranges when considering parameters as

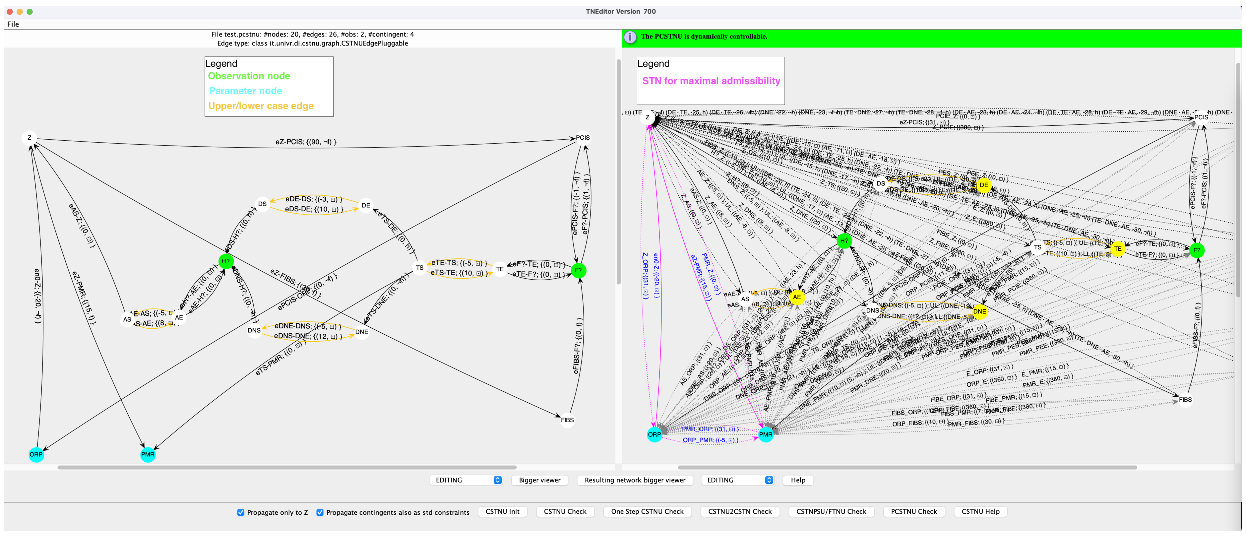

Figure 4: Screenshot of the Parameterized CSTNU checker when the input is the Parameterized CSTNU of Figure 2.

simple timepoints. Indeed, while for parameters, we have to set *before starting* the right execution time within the range, for timepoints, we can dynamically set the suitable execution time *during the execution* of the network. In the example, the derived range $[20, 31]$ for parameter *ORP* would remain $[20, +\infty]$ if we were allowed to consider it as an ordinary timepoint and apply the classical CSTNU checking algorithm. Indeed, the proposed algorithm for parameterized CSTNUs derives the maximal admissible ranges for parameters, which do not depend on the execution of the network. In contrast, the classical CSTNU algorithm for checking dynamic controllability does not derive the maximum allowed ranges.

Parameterized CSTNUs and the algorithm defined here can be considered to represent and solve also temporal problems from domains other than clinical. For Web services, for example, they can serve as the basis for defining possible temporal Service Level Agreements, e.g., a provider may advertise the admissible values for the service calls that yield the guarantee of timely execution.

In summary, this paper showed that there are temporal problems in which some timepoint assignments are decoupled from execution. In particular, it considered the case in which a future timepoint is fixed before starting the execution. To enable representing such problems, this paper introduced the notions of CSTNU parameter node and parameterized CSTNU. Then, it introduced a new set of constraint propagation rules for checking DC and computing schedules of parameterized CSTNUs. Finally, it defined a procedure for deriving a set of necessary and sufficient restrictions on parameter values that yield the DC of a parameterized CSTNU.

REFERENCES

[1] Massimo Cairo and Romeo Rizzi. 2016. Dynamic Controllability of Conditional Simple Temporal Networks is PSPACE-complete. In *23rd Int. Symp. on Temporal Repres. and Reasoning (TIME 2016)*. 90–99. https://doi.org/10.1109/TIME.2016.17

[2] Carlo Combi and Roberto Posenato. 2009. Controllability in temporal conceptual workflow schemata. In *Business Process Management, 7th Int. Conf., BPM 2009 (LNCS, Vol. 5701)*. 64–79. https://doi.org/10.1007/978-3-642-03848-8_6

[3] Carlo Combi and Roberto Posenato. 2018. Extending Conditional Simple Temporal Networks with Partially Shrinkable Uncertainty. In *25th Int. Symp. on Temporal Representation and Reasoning (TIME 2018) (LIPIcs, Vol. 120)*. 9:1–9:15. https://doi.org/10.4230/LIPIcs.TIME.2018.9

[4] Rina Dechter, Itay Meiri, and Judea Pearl. 1991. Temporal constraint networks. *Artif. intell.* 49, 1-3 (1991), 61–95. https://doi.org/10.1016/0004-3702(91)90006-6

[5] Johann Eder, Marco Franceschetti, and Julius Köpke. 2019. Controllability of Business Processes with Temporal Variables. In *Proc. of the 34th ACM/SIGAPP Symp. on Applied Computing*. 40–47. https://doi.org/10.1145/3297280.3297286

[6] Johann Eder, Euthimios Panagos, and Michael Rabinovich. 2013. Workflow Time Management Revisited. In *Seminal Contr. to Information Systems Eng.* 207–213. https://doi.org/10.1007/978-3-642-36926-1_16

[7] Marco Franceschetti and Johann Eder. 2021. Determining temporal agreements in cross-organizational business processes. *Information and Computation* 281 (2021), 104792. https://doi.org/10.1016/j.ic.2021.104792

[8] Luke Hunsberger. 2009. Fixing the semantics for dynamic controllability and providing a more practical characterization of dynamic execution strategies. In *16th Int. Symp. on Temporal Representation and Reasoning (TIME 2009)*. IEEE, 155–162. https://doi.org/10.1109/TIME.2009.25

[9] Luke Hunsberger and Roberto Posenato. 2018. Sound-and-Complete Algorithms for Checking the Dynamic Controllability of Conditional Simple Temporal Networks with Uncertainty. In *25th Int. Symp. on Temporal Repres. and Reasoning (TIME 2018)*, Vol. 120. 14:1–14:17. https://doi.org/10.4230/LIPICS.TIME.2018.14

[10] Luke Hunsberger, Roberto Posenato, and Carlo Combi. 2012. The Dynamic Controllability of Conditional STNs with Uncertainty. In *PlanEx @ ICAPS-2012*. 21–29. http://arxiv.org/abs/1212.2005

[11] Luke Hunsberger, Roberto Posenato, and Carlo Combi. 2015. A Sound-and-Complete Propagation-based Algorithm for Checking the Dynamic Consistency of Conditional Simple Temporal Networks. In *22nd Int. Symp. on Temporal Representation and Reasoning (TIME 2015)*. 4–18. https://doi.org/10.1109/TIME.2015.26

[12] Paul H. Morris and Nicola Muscettola. 2005. Temporal Dynamic Controllability Revisited. In *20th National Conf. on Artificial Intelligence (AAAI-2005)*. 1193–1198. http://www.aaai.org/Papers/AAAI/2005/AAAI05-189.pdf

[13] Levin Noronha and Fei Chiang. 2021. *Discovery of Temporal Graph Functional Dependencies.* 3348–3352. https://doi.org/10.1145/3459637.3482087

[14] Patrick T O'gara, Frederick G Kushner, Deborah D Ascheim, et al. 2013. 2013 ACCF/AHA guideline for the management of ST-elevation myocardial infarction. *Circulation* 127, 4 (2013), 529–555. https://doi.org/10.1161/CIR.0b013e3182742c84

[15] Roberto Posenato. 2022. CSTNU Tool: A Java library for checking temporal networks. *SoftwareX* 17 (2022), 100905. https://doi.org/10.1016/j.softx.2021.100905

[16] Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. 2012. Acquiring Temporal Constraints between Relations. In *Proc. of the 21st ACM Int. Conf. on Information and Knowledge Management (CIKM '12)*. 992–1001. https://doi.org/10.1145/2396761.2396886

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

[18] Thierry Vidal. 2000. Controllability characterization and checking in Contingent Temporal Constraint Networks. In *KR 2000*. 559–570.

[19] Thierry Vidal and Hélène Fargier. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *J. Exp. Theor. Artif. Intell.* 11, 1 (1999), 23–45. https://doi.org/10.1080/095281399146607