Mining Significant Temporal Networks Is Polynomial

Guido Sciavicco
Department of Mathematics and Computer Science, University of Ferrara, Italy
guido.sciavicco@unife.it

Matteo Zavatteri
Department of Computer Science, University of Verona, Italy
matteo.zavatteri@univr.it

Tiziano Villa
Department of Computer Science, University of Verona, Italy
tiziano.villa@univr.it

—— **Abstract** ——

A Conditional Simple Temporal Network with Uncertainty and Decisions (CSTNUD) is a formalism that tackles controllable and uncontrollable durations as well as controllable and uncontrollable choices simultaneously. In the classic top-down model-based engineering approach, a designer builds a CSTNUD to model, validate and execute some temporal plan of interest. Instead, in this paper, we investigate the bottom-up approach by providing a deterministic *polynomial time* algorithm to *mine* a CSTNUD from a set of execution traces (i.e., a log). This paper paves the way for the design of controllable temporal networks mined from traces that also contain information on uncontrollable events.

1 Introduction

Temporal networks are a possible framework to model temporal plans and check the coherence of their temporal constraints imposing delays and deadlines between the occurrences of pairs of events in the plan [7]. The main components of a temporal network are *time points* and *constraints*. Time points are real variables modeling temporal events. Executing time points means to assign them real values to fix "when" the corresponding temporal events occurred. Constraints are linear inequalities imposing minimal and maximal temporal distances between pairs of time points.

Over the years the core formalism of *Simple Temporal Networks* [7] has been extended in several ways to cope with uncontrollable durations [17], uncontrollable and controllable choices [5, 13] and, more recently, with combinations of them (see, e.g., [3, 11, 12, 18, 19]). The most expressive formalisms of temporal networks are those that simultaneously handle all such features. Moreover, such formalisms give rise to several, different, taxonomies in which sub-formalisms belonging to them can be ordered by expressive power. As a result, solving any problem for a top-level formalism (e.g., checking consistency or controllability)

results in solving the same problem for every sub-formalism in the corresponding hierarchy. *Conditional Simple Temporal Networks with Uncertainty and Decisions (CSTNUDs, [19, 22])* is a recent formalism tackling controllable and uncontrollable durations as well as controllable and uncontrollable choices simultaneously. CSTNUDs define a hierarchy of temporal networks in which any combination of features can be considered by focusing on the corresponding sub-formalism.

Like any model-based engineering approach, creating a temporal network is a complex, time-consuming, and error-prone task, where typically discrepancies between the actual process and the obtained network might eventually emerge asking the designer for refinement or abstraction of the model being created. This is a top-down, trial-and-error approach. Instead, the opposite, bottom-up, approach is known in the literature under the name of *process mining* and it aims to *mine* (i.e., synthesize) process descriptions (or, more reasonably, model approximations) from execution traces (i.e., process logs).

A *trace* formalizes a run of a process, and the set of all available traces can be thought of as a *log* of a process carried out many times by humans that base their actions on their experience only. As a result, since such actions may not follow any particular rule, we have no guarantee of consistency or controllability of the underlying process overall. One of the first contributions in process mining is that of Agrawal, Gunopulos, and Leymann [1], but, after this seminal work, many others have come by focusing on different process description languages [6, 8, 14, 15, 16]. However, to the best of our knowledge, the problem of mining temporal networks subject to uncontrollable parts has not received particular attention. Despite the current trend in process mining calls for machine learning techniques, we shall see that the well-founded mathematical structure of temporal networks allows us to solve this problem correctly and efficiently, because of a strong underlying monotonicity.

Contribution. We provide a deterministic polynomial time algorithm to mine a CSTNUD from a finite set of execution traces. By construction, every trace in the set will satisfy the constraints of the mined CSTNUD, therefore the CSTNUD is correct, complete and significant, meaning that every temporal event, (un)controllable duration and (un)controllable choice belonging to some processed trace occurs in it.

Organization. Section 2 discusses background and related work. Section 3 adapts CSTNUDs for the purpose of this paper. Section 4 defines the problem of mining significant CSTNUDs and provides a correct algorithm for it. In Section 5 we conclude by summing up and discussing future work.

2 Background and Related Work

Simple Temporal Networks (STNs) [7] model fully controllable and non-disjunctive temporal plans but they cannot deal with (un)controllable choices nor with uncontrollable durations. To bridge such gaps some extensions were put forth over the years. *Simple Temporal Networks with Uncertainty (STNUs)* [17] extend STNs with uncontrollable (but bounded) durations by means of contingent links. A contingent link consists of an activation (time) point, whose execution is under control, a contingent (time) point, whose execution is not, and a closed interval specifying the minimal and maximal duration of the link. *Conditional Simple Temporal Networks (CSTNs)* [10] (formerly, *Conditional Temporal Problem (CTP)* [13]) extend STNs with uncontrollable choices. Constraints are *labeled* by sets of consistent literals over a finite set of uncontrollable Boolean variables, or *booleans*, which describe when the

(a) A hierarchy of STNs.
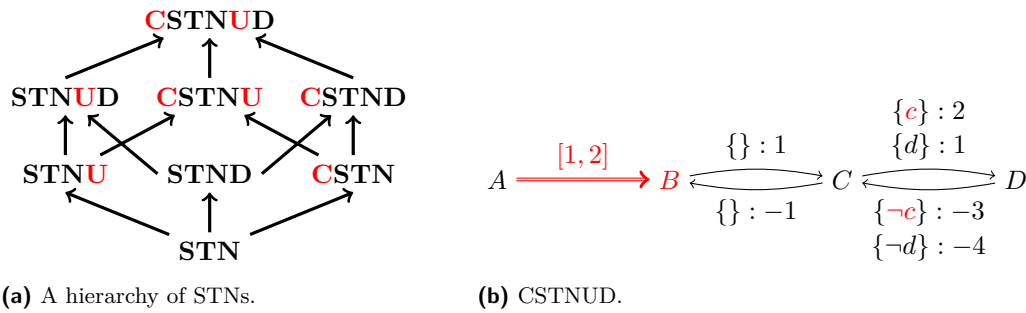
(b) CSTNUD.

■ **Figure 1** Hierarchy (left) and an example of CSTNUD (right). STNDs, STNUDs and CSTNDs are further frameworks implicitly arising from CSTNUDs. Any acronym speaks for what the corresponding framework supports: "C" (uncontrollable choices), "D" (controllable choices), "U" (contingent links).

components labeled by them are relevant for an execution. The truth value assignments to these booleans are out of control and take place upon the execution of specific time points. Initially, labels were on both time points and constraints but later it was proved that having labels on constraints only does not limit the expressiveness of the network [2]. Temporal networks with labels on constraints only are called *streamlined. Conditional Simple Temporal Networks with Uncertainty (CSTNUs)* [9] merge STNUs and CTPs/CSTNs, whereas *Conditional Simple Temporal Networks with Uncertainty and Decisions (CSTNUDs)* [19, 22] extend CSTNUs with controllable choices (i.e., controllable booleans). Figure 1a shows the hierarchy of CSTNUDs. Figure 1b gives an example of CSTNUD that we discuss in Section 3. Other formalisms were built on top of STNs (e.g., [3, 5, 11, 12, 18]) but are not employed in this work.

Process mining has been approached by several authors. Agrawal, Gunopulos and Leymann first introduced the problem of producing a process description from unstructured executions in a log [1]. Cook and Wolf investigated similar issues in the context of software engineering processes [6]. They described three methods for process discovery: one based on neural networks, one based on a purely algorithmic approach, and one based on a Markovian approach. In particular, in the second approach, they built a finite state machine where states are fused whenever their futures (in terms of possible behavior in the next k steps) are identical. In [8], Herbst addresses the issue of process mining in the context of workflow management using an inductive (machine learning based) approach. Finally, in [14], Van Der Aalst proposes an algorithm to extract a workflow network from logs of a hospital. The results of the experiments highlight that the proposed method can discover processes whose underlying models are acyclic and sound workflow nets, involving parallel, conditional and sequential workflow blocks.

3 Conditional STNUs with Decisions

When uncontrollable parts are supported, the corresponding planning and scheduling problem modeled by the underlying network can be seen as a two-player game between Controller (representing the executor) and Nature (representing the environment). Controller executes controllable time points and assigns truth values to controllable booleans. Nature does the same for uncontrollable time points and uncontrollable booleans. Controller aims to satisfy all constraints. Nature aims to have Controller violate at least one of them. In other words, we are the Controller and everything else is Nature.

▶ **Definition 1.** *A* Conditional Simple Temporal Network with Uncertainty and Decisions (CSTNUD) *is a tuple* $\langle \mathcal{T}, \mathcal{B}, \mathcal{T}_B, \beta, \mathcal{L}, \mathcal{C} \rangle$*, where:*

- $\mathcal{T} = \mathcal{T}_C \dot{\cup} \mathcal{T}_U = \{A, \ldots, Z\}$ *is a* finite *set of* time points *disjointly partitioned in* controllable time points *(those executed by Controller) and* uncontrollable time points *(those executed by Nature), respectively.*

- $\mathcal{B} = \mathcal{B}_C \dot{\cup} \mathcal{B}_U = \{a, \ldots, z\}$ *is a* finite *set of* booleans *disjointly partitioned in* controllable booleans *(those assigned by Controller) and* uncontrollable booleans *(those assigned by Nature), respectively.*

- $\mathcal{T}_B \subseteq \mathcal{T}_C$ *is the set of controllable time points having booleans associated according to* β.

- $\beta : \mathcal{T}_B \rightarrow \mathcal{B}$ *is a bijection assigning to any* $A \in \mathcal{T}_B$ *the boolean* $\beta(A)$*. Once a time point* $A \in \mathcal{T}_B$ *is executed, the truth value of* $\beta(A)$ *is set by Controller (if* $\beta(A) \in \mathcal{B}_C$*) or by Nature (if* $\beta(A) \in \mathcal{B}_U$*).*

- $\mathcal{L}$ *is a set of* contingent links *each having the form* (A, ℓ, u, B) *where* $A \in \mathcal{T}_C$*,* $B \in \mathcal{T}_U$*,* $\ell, u \in \mathbb{R}$ *with* $0 < \ell \leq u$*. Once* A *is executed by Controller,* B *is executed by Nature guaranteeing that the temporal distance between* A *and* B *falls in* $[\ell, u]$*. Contingent links do not share uncontrollable time points.*

- $\mathcal{C}$ *is a set of* temporal constraints *having the form* $\mathcal{S} : B - A \leq k$*, where* $\mathcal{S}$*, the* label *of the constraint, is a consistent set of literals over* $\mathcal{B}$*,* $B, A \in \mathcal{T}$ *and* $k \in \mathbb{R}$*. Many temporal constraints* $\mathcal{S}_i : B - A \leq k_i$ *may be defined for the same pair of time points (w.r.t. the same direction[1]) provided* $\mathcal{S}_i$ *is different. Any pair* $\mathcal{S}_1 : B - A \leq k_1$ *and* $\mathcal{S}_2 : B - A \leq k_2$ *with* $\mathcal{S}_1 = \mathcal{S}_2$*, implies* $\mathcal{S}_1 : B - A \leq \min\{k_1, k_2\}$ *(tightening). Temporal constraints labeled by* $\mathcal{S} = \emptyset$ *are unconditional (i.e., they must always hold). Those labeled by* $\mathcal{S} \neq \emptyset$ *are conditional: they hold only if all literals in* $\mathcal{S}$ *are satisfied by the truth value assignment to the booleans.*

Definition 1 differs from that given in [19] as follows. First, our CSTNUDs are streamlined. Second, we allow the intervals of contingent links to be a single point; despite this resembles no uncertainty, it is an extension that does not break the current semantics (we just know what Nature will do in that case). Third, we no longer differentiate between observation and decision time points[2] but we just focus on controllable and uncontrollable booleans.

We graphically represent a CSTNUD as a directed graph whose set of nodes coincides with the set of time points and whose set of edges divides in double and single edges. A double edge $A \Rightarrow B$ labeled by $[\ell, u]$ models a contingent link (A, ℓ, u, B). A single edge $A \rightarrow B$ labeled by $\mathcal{S} : k$ models a temporal constraint $\mathcal{S} : B - A \leq k$. Figure 1b shows an example of CSTNUD where we highlight uncontrollable parts in red. This network contains three controllable time points A, C, D, one uncontrollable time point B, one contingent link $(A, 1, 2, B)$, a controllable boolean d associated to D, an uncontrollable boolean c associated to C, two unconditional temporal constraint $\emptyset : C - B \leq 1$ and $\emptyset : B - C \leq -1$ and four conditional ones $\{\neg c\} : C - D \leq -3$, $\{\neg d\} : C - D \leq -4$, $\{c\} : D - C \leq 2$ and $\{d\} : D - C \leq 1$. A few problems are associated to CSTNUDs. For example, when $\mathcal{B}_U$ and $\mathcal{T}_U$ are both empty, the network does not have uncontrollable parts; in this case, we may ask whether the network is *consistent*. On the other hand, when at least one among $\mathcal{B}_U$ and $\mathcal{T}_U$ is nonempty, then the network has at least one uncontrollable part, and consistency is no longer a well-defined problem. In this case, we worry about *controllability*, that is,

---

[1] Regardless of $\mathcal{S}$ and k, a constraint on a pair of time points A, B has two possible "directions": $\mathcal{S} : B - A \leq k$ and $\mathcal{S} : A - B \leq k$.

[2] Historically, time points associated to controllable (resp., uncontrollable) booleans were called decisions (resp., observations).

if we can find an execution strategy for the CSTNUD, according to different assumptions on Nature's behavior. Given a CSTNUD $\mathcal{Z} = \langle \mathcal{T}, \mathcal{B}, \mathcal{T}_B, \beta, \mathcal{L}, \mathcal{C} \rangle$, we say that a (partial) mapping $t \colon \mathcal{T} \to \mathbb{R}$ assigning real values to the time points is a *schedule* if it enforces that for each $(A, \ell, u, B) \in \mathcal{L}$, if $B \in \mathtt{dom}(t)$, then $A \in \mathtt{dom}(t)$ and $t(B) \in [t(A) + \ell, t(A) + u]$. Furthermore, let $\mathcal{P}$ be a consistent set of literals over $\mathcal{B}$, that is, a (partial) instantiation of truth values to the booleans of a network. We call $\langle \mathcal{P}, t \rangle$ a *model*. $\langle \mathcal{P}, t \rangle$ is total iff $\mathtt{dom}(t) = \mathcal{T}$ and for each $a \in \mathcal{B}$ either a or $\neg a$ is in $\mathcal{P}$.

▶ **Definition 2.** *Let $\mathcal{Z} = \langle \mathcal{T}, \mathcal{B}, \mathcal{T}_B, \beta, \mathcal{L}, \mathcal{C} \rangle$ be a CSTNUD. We say that the model $\langle \mathcal{P}, t \rangle$ satisfies a network $\mathcal{Z}$ (in symbols, $\langle \mathcal{P}, t \rangle \models \mathcal{Z}$) if and only if for each $\mathcal{S} \colon B - A \leq k \in \mathcal{C}$, whenever $\mathcal{S} \subseteq \mathcal{P}$ and $A, B \in \mathtt{dom}(t)$, then $t(B) - t(A) \leq k$.*

▶ **Definition 3.** *Let $\mathcal{Z} = \langle \mathcal{T}, \mathcal{B}, \mathcal{T}_B, \beta, \mathcal{L}, \mathcal{C} \rangle$ be a CSTNUD. We say that:*
- *$\mathcal{Z}$ is* weakly controllable *if whenever Nature tells Controller (before starting the execution) what durations and truth values she is going to assign to contingent links and uncontrollable booleans, Controller can generate a schedule and a consistent set of literals that contain the information given by Nature and satisfy all constraints.*
- *$\mathcal{Z}$ is* strongly controllable *if Controller can find a unique schedule for controllable time points and a unique consistent set of literals over controllable booleans that will satisfy all constraints regardless of any possible extension that Nature can provide for uncontrollable time points and uncontrollable booleans.*
- *$\mathcal{Z}$ is* dynamically controllable *if Controller can dynamically generate a schedule over controllable time points and a consistent set of literals over controllable booleans depending on which extension Nature is providing for uncontrollable time points and uncontrollable booleans.*

The CSTNUD shown in Figure 1b is weakly, dynamically but not strongly controllable. We provide an example of dynamic execution strategy. Controller executes A at 0. Then, Nature executes B at a time falling in $[1, 2]$. After that, Controller executes C exactly 1 after B and Nature chooses a truth value for c. If c is true, then Controller executes D within 1 after C and assigns true to d. If c is false, then Controller executes D after 4 since C and assigns false to d.

4 Mining Significant CSTNUDs

As we explained in Section 2, a CSTNUD models a temporal plan in a compact way. The problems that are associated with a network allow one to study intrinsic properties of the network itself. In real-world cases, often the network is not given, but, on the contrary, it must be hand-craftily designed. In this section, we approach this problem: given a finite set of execution traces (e.g., a *log*) of the underlying temporal plan, we solve the problem of automatically mine a "good" CSTNUD that describes it.

▶ **Definition 4.** *A trace τ is a sequence of these statements:*
- *$A = t_A$, where $t_A \in \mathbb{R}_{\geq 0}$. This statement models the execution of a controllable time point A at time t_A.*
- *$B(A) = t_B$, where $t_B \in \mathbb{R}_{\geq 0}$. This statement models the execution of an uncontrollable time point B at time t_B whose corresponding activation is A (contingent link).*
- *$a!$ (resp., $\neg a!$). This statement models that the controllable boolean a was assigned true (resp., false).*
- *$a?$ (resp., $\neg a?$). This statement models that the uncontrollable boolean a was assigned true (resp., false).*
Controllable and uncontrollable booleans are identified by the suffixes ! and ?, respectively.

The following are example of traces:

τ_1: $Z = 0$, $A = 0$, $\neg a?$

τ_2: $Z = 0$, $A = 0$, $\neg a?$, $E(A) = 5$

τ_3: $Z = 0$, $A = 0$, $a?$, $B = 2$, $b!$

τ_4: $Z = 0$, $A = 0$, $a?$, $C = 1$, $B = 2$, $\neg b!$, $D = 4$

τ_5: $Z = 0$, $B = 0$, $b!$, $C = 2$, $E(A) = 2$, $A = 3$, $a?$, $D = 4$

τ_6: $Z = 0$, $A = 0$, $a?$, $B = 1$, $b!$, $C = 4$, $D = 6$, $E(A) = 7$

τ_7: $Z = 0$, $A = 0$, $a?$, $D = 5$, $B = 5$, $\neg b!$

▶ **Definition 5.** *A trace τ is* well-defined *if it is finite, starts with $Z = 0$, and:*

- *Any time point (resp., any boolean) appearing in τ is assigned a real value (resp., truth value) exactly once.*
- *If $B(A) = t_B$ appears in τ, then $A = t_A$ appears in τ before $B(A) = t_B$.*
- *If any of $a!, \neg a!, a?, \neg a?$ appears in τ, then the statement appearing immediately before it is $A = t_A$, meaning that $\beta(A) = a$, whereas that appearing immediately after it (if any) is either $A' = t_{A'}$ or $A'(A'') = t_{A'}$.*
- *If a statement $B = t_B$ (or $B(B') = t_B$) appears after a statement $A = t_A$ (or $A(A') = t_A$), then $t_B \geq t_A$.*

▶ **Definition 6.** *A pair of traces is* coherent *if and only if:*

- *Any time point appearing in both traces is of the same type, and, if such a time point is uncontrollable, then it also refers to the same activation.*
- *Any boolean appearing in both traces is of the same type, and it is associated to the same time point.*

A set of traces is coherent if every pair of traces in it is coherent.

▶ **Definition 7.** *A CSTNUD $\mathcal{Z}$ is* significant *for a well-defined trace τ if the following conditions hold.*

- *If $A = t_A \in \tau$, then $A \in \mathcal{T}_C$*
- *If $B(A) = t_B \in \tau$, then $B \in \mathcal{T}_U$ and $(A, \ell, u, B) \in \mathcal{L}$ for some $\ell, u \in \mathbb{R}$, $0 \leq \ell \leq u$ such that $t_B - t_A \in [\ell, u]$.*
- *If $a! = \top \in \tau$ or $a! = \bot \in \tau$ (resp., $a? = \top \in \tau$ or $a? = \bot \in \tau$), then $a \in \mathcal{B}_C$ (resp., $a \in \mathcal{B}_U$) and $\beta(A) = a$; moreover, if $A = t_A$ is the statement before it, then $\beta(A) = a$.*
- *$\langle \mathcal{P}, t \rangle \models \mathcal{Z}$, where $\mathcal{P}$ and t are the consistent set of literals and schedule arising from τ, respectively.*

The set $\mathcal{I} = \{\tau_1\tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7\}$ in the example above is coherent and contains well-defined traces.

> **Problem.** Given a finite set of well-defined and coherent traces, mine a significant CSTNUD.

CstnudMiner (Algorithm 1) starts by creating a CSTNUD containing the zero-time point Z only. After processing a trace, $\mathcal{Z}$ contains all time points, booleans, contingent links and temporal constraints specified by that trace. After processing a set of traces, $\mathcal{Z}$ is such that all traces in that set satisfy it, and once a trace is processed, it can be forgotten. **CstnudMiner** internally uses the rules **WeakenTC** and **WeakenCL** on temporal constraints and contingent links, respectively. Table 1 shows these weakening rules. The aim of these two sub-procedures is to add temporal constraints and contingent links if they do not exists in $\mathcal{Z}$ or to *weaken* them otherwise (to allow new traces to satisfy $\mathcal{Z}$ as well). Before proceeding we introduce some useful notation. Given a pair of time point A, B and a set of literals $\mathcal{S}$, let $L(B, A) = \{\mathcal{S} \mid \mathcal{S} : B - A \leq k \in \mathcal{C}\}$ be the set of labels of all temporal constraints going from A to B and let:

Table 1 Weakening rules for Algorithm 1.

WeakenTC($\mathcal{S} : B - A \leq k$)	
Case T1: $L_1(\mathcal{S}, B, A) = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\} \neq \emptyset$.	
$\cdots$ $\mathcal{S}_1 : k_1 \ldots \mathcal{S}_n : k_n$ $A \xrightarrow{\cdots} B$	$\cdots$ $\mathcal{S}_1 : \max\{k_1, k\} \ldots \mathcal{S}_n : \max\{k_n, k\}$ $A \xrightarrow{\cdots} B$
Example: WeakenTC($\{a, b\} : C - Z \leq 7$)	
$\{\neg c, \neg a\} : 6 \quad \{b\} : 8$ $Z \xrightarrow{\{a\} : 3} C$	$\{\neg c, \neg a\} : 6 \quad \{b\} : 8$ $Z \xrightarrow{\{a\} : 7} C$
Case T2: $L_2(\mathcal{S}, B, A) = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\} \neq \emptyset$.	
$\cdots$ $\mathcal{S}_1 : k_1 \ldots \mathcal{S}_n : k_n$ $A \xrightarrow{\cdots} B$	$\cdots$ $\mathcal{S} : \max\{K_2(\mathcal{S}, B, A), k\}$ $A \xrightarrow{\cdots} B$
Example: WeakenTC($\{b\} : Z - C \leq -7$)	
$\{\neg c, b\} : -8 \quad \{\neg b, \neg a\} : -6$ $Z \xleftarrow{\{a, b\} : -3} C$	$\{\neg b, \neg a\} : -6$ $Z \xleftarrow{\{b\} : -3} C$
Case T3: $L_1(\mathcal{S}, B, A) = L_2(\mathcal{S}, B, A) = \emptyset$.	
$A \xrightarrow{\cdots} B$	$A \xrightarrow{\cdots \mathcal{S} : k \cdots} B$
Example: WeakenTC($\{b\} : C - Z \leq 5$)	
$\{a, \neg b\} : 8$ $Z \xrightarrow{\{\neg b\} : 3} C$	$\{a, \neg b\} : 8 \quad \{b\} : 5$ $Z \xrightarrow{\{\neg b\} : 3} C$
WeakenCL(A, k, k, B)	
Case L1: contingent link between A and B exists	
$A \xRightarrow{[\ell, u]} B$	$A \xRightarrow{[\min\{l, k\}, \max\{u, k\}]} B$
Case L2: no contingent link between A and B	
$A \qquad\qquad B$	$A \xRightarrow{[k, k]} B$

- $L_1(\mathcal{S}, B, A) = \{\mathcal{S}_i \mid \mathcal{S}_i \in L(B, A), \mathcal{S}_i \subseteq \mathcal{S}\}$ be the set of labels in $L(B, A)$ contained in $\mathcal{S}$.
- $L_2(\mathcal{S}, B, A) = \{\mathcal{S}_i \mid \mathcal{S}_i \in L(B, A), \mathcal{S} \subset \mathcal{S}_i\}$ be the set of labels in $L(B, A)$ strictly containing $\mathcal{S}$.
- $L_3(\mathcal{S}, B, A) = L(B, A) \setminus (L_1 \cup L_2)$ be the set of all other labels in $L(B, A)$ neither in L_1 nor in L_2.
- $K_i(\mathcal{S}, B, A) = \{k_j \mid \mathcal{S}_j : B - A \leq k_j \in \mathcal{C}, \mathcal{S}_j \in L_i(\mathcal{S}, B, A)\}$ be the set of weights of all constraints from A to B labeled by $\mathcal{S}_j \in L_i(\mathcal{S}, B, A)$ for $i = 1, 2, 3$.

■ **Algorithm 1 CstnudMiner.**

Input: A set $\mathcal{I}$ of well-defined and coherent traces.
Output: A significant CSTNUD.

1 $\mathcal{Z} = \langle \{Z\}, \emptyset, \emptyset, \beta, \emptyset, \emptyset \rangle$ ▷ "Initial CSTNUD"
2 **foreach** $\tau \in \mathcal{I}$ **do**
3 $\mathcal{S} \leftarrow \emptyset$
4   **foreach** *statement in* τ **do**
5   **if** *the statement is* $A = t_A$ **then**
6 $\mathcal{T}_C \leftarrow \mathcal{T}_C \cup \{A\}$
7   **WeakenTC**$(\mathcal{S} : A - Z \leq t_A)$
8   **WeakenTC**$(\mathcal{S} : Z - A \leq -t_A)$
9   **if** *the statement is* $B(A) = t_B$ **then**
10 $\mathcal{T}_U \leftarrow \mathcal{T}_U \cup \{B\}$
11   Let t_A be the execution time of A.
12   **WeakenCL**$(A, t_B - t_A, t_B - t_A, B)$
13   **if** *the statement is* $a!$ *or* $\neg a!$ *or* $a?$ *or* $\neg a?$ **then**
14   Let $A = t_A$ be the previous statement.
15   **if** *the suffix is* ! **then** $\mathcal{B}_C \leftarrow \mathcal{B}_C \cup \{a\}$;
16   **else** $\mathcal{B}_U \leftarrow \mathcal{B}_U \cup \{a\}$;
17 $\beta(A) = a$
18   **if** *the prefix is* $\neg$ **then** $\mathcal{S} \leftarrow \mathcal{S} \cup \{\neg a\}$;
19   **else** $\mathcal{S} \leftarrow \mathcal{S} \cup \{a\}$;

20 **return** $\mathcal{Z}$

WeakenTC works on a temporal constraint $\mathcal{S} : B - A \leq k$. When called, **WeakenTC** first computes $L_1(\mathcal{S}, B, A)$, $L_2(\mathcal{S}, B, A)$ and $L_3(\mathcal{S}, B, A)$. Then, it handles three cases as follows:

Case T1. This case applies whenever $L_1(\mathcal{S}, B, A) \neq \emptyset$. In such a case, each weight k_i associated to a constraint going from A to B labeled by any $\mathcal{S}_i \in L_1(\mathcal{S}, B, A)$ is possibly weakened (meaning raised) to k if $k > k_i$.

Case T2. This case applies whenever $L_2(\mathcal{S}, B, A) \neq \emptyset$. In such a case, we add a single constraint labeled by $\mathcal{S}$ whose numeric weight is the maximum value in the set $K_2(\mathcal{S}, B, A) \cup \{k\}$. Finally, we remove each constraint labeled by any $\mathcal{S}_i \in L_2(\mathcal{S}, B, A)$.
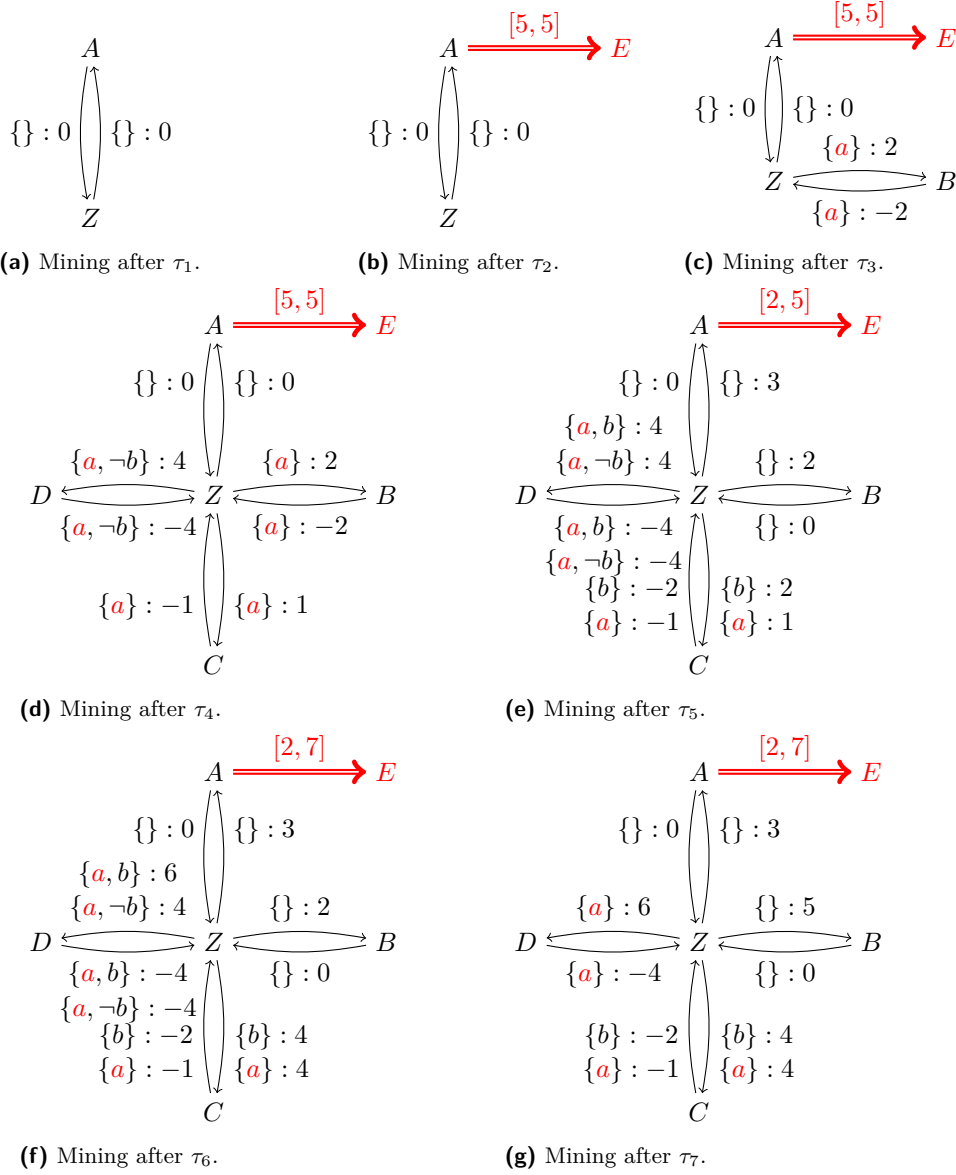
Case T3. This case applies whenever $L_1(\mathcal{S}, B, A) = L_2(\mathcal{S}, B, A) = \emptyset$ (i.e., whenever neither Case T1 nor Case T2 does). In such a case we just add $\mathcal{S} : A - B \leq k$.

WeakenCL, on the other hand, works on a contingent link (A, k, k, B) by means of two mutually-exclusive cases:

Case L1. This case applies whenever (A, ℓ, u, B) already exists. In such a case, we weaken (meaning lower) ℓ to k if $k < \ell$ or weaken (meaning raise) u to k if $u < k$ (note that at most one among ℓ and u is weakened).

Case L2. This case applies whenever (A, ℓ, u, B) does not exist. In such a case, we add (A, k, k, B).

We provide application examples for **WeakenTC** in Table 1. We omit those for **WeakenCL** due to their triviality. Figure 2 shows the result of applying **CstnudMiner** on the previous discussed set of traces $\mathcal{I} = \{\tau_1, \ldots, \tau_7\}$.

(a) Mining after τ_1.

(b) Mining after τ_2.

(c) Mining after τ_3.

(d) Mining after τ_4.

(e) Mining after τ_5.

(f) Mining after τ_6.

(g) Mining after τ_7.

Figure 2 Mining a significant CSTNUD from execution traces.

We are left to discuss invariants, correctness and complexity of **CstnudMiner**. Let $\mathcal{Z}$ be the CSTNUD being mined.

▶ **Invariant 1.** *Cases T1, T2 and T3 of* **WeakenTC** *are mutually-exclusive. So are cases L1 and L2 of* **WeakenCL**.

Proof. We only need to focus on **WeakenTC**, as cases L1 and L2 of **WeakenCL** are mutually-exclusive by definition (either a contingent link exists or it doesn't). When **CstnudMiner** starts, the invariant is trivially true as no constraints exist (so only case T3 is enable). Now assume that Invariant 1 holds and let $\mathcal{S}$ be the set of literals collected in the current trace τ being processed. Moreover, assume the current processed statement is $A = t_A$. If Case T1 applies, only the weights of the constraints labeled by some $\mathcal{S}_i \in L_1(\mathcal{S}, B, A)$

are (possibly) modified. After processing the statement, $L(B, A)$ remains the same, thus Invariant 1 still holds. If Case T2 applies, all constraints labeled by some $\mathcal{S}_i \in L_2(\mathcal{S}, B, A)$ are thrown away and replaced by a single constraint labeled by $\mathcal{S}$. After processing the statement, (the new) $L(B, A)$ contains $\mathcal{S}$ but does not contain any $\mathcal{S}_i$ in the previous $L_2(\mathcal{S}, B, A)$. Note that $\mathcal{S}$ is not a superset of any other $\mathcal{S}_i \in L(B, A)$, $\mathcal{S}_i \neq \mathcal{S}$ as if it were, so would each set in $L_2(\mathcal{S}, B, A)$ before processing the statement (invariant contradiction). If Case T3 applies, a constraint labeled by $\mathcal{S}$ is merely added. After that, the new $L(B, A)$ contains $\mathcal{S}$, and Invariant 1 still holds. ◄

▶ **Theorem 8.** *CstnudMiner mines a significant CSTNUD.*

Proof. We need to prove the following: first, that **WeakenTC** and **WeakenCL** are *sound* meaning that any constraint and uncontrollable duration for contingent links that held before applying the rules keeps holding after applying them, and, second, that **CstnudMiner** mines the same CSTNUD regardless of the ordering in which traces are processed.

Soundness of the rules. From Invariant 1 we know that all cases in Table 1 are mutually exclusive. Therefore, we analyze each case of each rule in isolation. Consider a call to **WeakenTC**$(\mathcal{S} : B - A \leq k)$. In Case T1 only the numeric weights of all constraints $\mathcal{S}_i : B - A \leq k_i$ where $\mathcal{S}_i \in L_1(\mathcal{S}, B, A)$ are (possibly) modified. For each $\mathcal{S}_i \in L_1(\mathcal{S}, B, A)$, let k_i be the weight of the constraint $\mathcal{S}_i : B - A \leq k_i$. After the rule applies we have that the new weight is $\max\{k_i, k\}$. It is clear that the initial constraint still holds as $\mathcal{S}_i \subseteq \mathcal{S} : B - A \leq k_i \leq \max\{k_i, k\}$. All remaining constraints in the CSTNUD hold as well as they are left untouched. In Case T2 all constraints labeled by some some set in $L_2(\mathcal{S}, B, A)$ are thrown away and replaced by a single constraint labeled by $\mathcal{S}$. After the rule applies we have that $\mathcal{S} : B - A \leq \max(K_2(\mathcal{S}, B, A) \cup \{k\})$ is added. Each removed constraint $\mathcal{S}_i : B - A \leq k_i$ still holds as $\mathcal{S} \subset \mathcal{S}_i : B - A \leq k_i \leq \max(K_2(\mathcal{S}, B, A) \cup \{k\})$. Once again, all remaining constraints in the CSTNUD hold as well as they are left untouched. In Case T3 no existing constraint is modified. Now, consider a call to **WeakenCL**(A, k, k, B). We only need to focus on the part of the durations related to this link as all other parts of such durations are left untouched. In Case L1 a contingent link (A, ℓ, u, B) already exists.

After applying the rule, either ℓ is lowered to k (if $k < \ell$) or u is raised to k (if $u < k$) or both are left untouched (if $\ell \leq k \leq u$). Let ℓ' and u' be the new minimal and maximal durations after the rule is applied. It is clear that $\ell' \leq \ell \leq u \leq u'$, therefore all previous durations are still possible. In Case L2 the contingent link does not exist yet, therefore we have nothing to verify.

Processing order of traces. CstnudMiner processes each trace once. Processing a trace means to process each statement in it. Whenever a time point does not exist, the algorithm adds it to the CSTNUD associating it to the right activation if the time point is uncontrollable. Likewise, whenever a boolean does not exist, the algorithm adds it to the CSTNUD and sets β accordingly. Also, the significance of the resulting CSTNUD (constraints and contingent links) follows from the soundness of the rules. ◄

We are left to discuss the complexity of **CstnudMiner**. Let $\mathcal{I}$ be the set of well-defined and coherent traces in input.

▶ **Theorem 9.** *CstnudMiner runs in polynomial time.*

Proof. We focus on **WeakenTC** since the operations carried out by **WeakenCL** are negligible (no partitioning of contingent links is done). Instead, **WeakenTC** hides internally inner cycles to compute $L_1(\mathcal{S}, B, A)$, $L_2(\mathcal{S}, B, A)$ and $L_3(\mathcal{S}, B, A)$ every time that it is called. The worst case happens when **CstnudMiner** applies **WeakenTC** as much as possible as

$L_3(\mathcal{S}, B, A)$ keeps growing. We show how to build a set of traces $\mathcal{I}$ that leads to this situation. Let $\mathcal{B}$ be a finite set of booleans and $\mathcal{T} = \{Z, X\} \cup \{Y_b \mid b \in \mathcal{B}\}$ a finite set of time points. In this way every boolean in b can be associated to a time point in $\mathcal{T} \setminus \{Z, X\}$. We shall see that it is enough to have $|\mathcal{I}| \leq 2^{|\mathcal{B}|}$. In our construction, if $\mathcal{B} = \{a_1, \ldots, a_{|\mathcal{B}|}\}$, then any trace has the form $Z = 0, Y_{a_1} = 0, a = \diamond, \ldots, Y_{a_{|\mathcal{B}|}} = 0, a_{|\mathcal{B}|} = \diamond, X = 0$ where $\diamond \in \{\top, \bot\}$ with trace specifying a different set of literals $\mathcal{S}$.

In this way, each trace $\tau \in \mathcal{I}$ has length $|\tau| = 2 + 2 \times |\mathcal{T} \setminus \{Z, X\}|$. Each trace specifies a different truth value assignment for the booleans in $\mathcal{B}$. When the last statement of each trace is processed, **CstnudMiner** can only add a new constraint between Z and X (for each direction). As a result, the maximum number of constraints that appear between Z and X (in any direction) is $|\mathcal{I}|$. However, when the "$X = 0$" of the $|\mathcal{I}|^{\text{th}}$ trace is processed, we know (from the proof of Theorem 8) that **WeakenTC**$(\mathcal{S} : X - Z \leq 0)$ (resp., **WeakenTC**$(\mathcal{S} : Z - X \leq 0)$) partitions the set of constraints between Z and X (resp., between Z and X) in $L_1(\mathcal{S}, X, Z), L_2(\mathcal{S}, X, Z), L_3(\mathcal{S}, X, Z)$ (resp., $L_1(\mathcal{S}, Z, X), L_2(\mathcal{S}, Z, X), L_3(\mathcal{S}, Z, X)$). The cost of this operation is $2 \times (|\mathcal{I}| - 1)$ (the number of constraints that are currently between Z and X in both directions). Eventually, when all traces have been processed, the overall cost of this operation is $2 \times \sum_{i=0}^{|\mathcal{I}|-1} n = 2 \times ((|\mathcal{I}| - 1) \times |\mathcal{I}|)/2 = \mathcal{O}(|\mathcal{I}|^2)$. Since this term is greater of any other number of analyzed constraints between Z and any Y_b in the trace, an upper bound for the algorithm is given by $\mathcal{O}(|\mathcal{I}|^2 \times |\tau|)$.   ◀

5   Conclusions and Future Work

Like any model-based engineering approach, creating a temporal network is a complex, time-consuming, and error-prone task. Along the lines of the bottom-up approach in the field of process mining, we proposed **CstnudMiner**, an algorithm for mining significant CSTNUDs from execution traces that also contain information on uncontrollable events. A CSTNUD is significant if it contains all time points, booleans and uncontrollable durations in the processed traces and each partial instantiation of a schedule and consistent set of literals arising from any processed trace satisfies all constraints involving those components. We proved that **CstnudMiner** runs in polynomial time with respect to the size of the set of input traces, and the length of each trace. Since in our approach once a trace is processed it can be forgotten, this paves the way for future "streaming" versions of the algorithm. As future work, we plan to carry out a thorough analysis of the properties of the mined CSTNUDs as well as adapting the algorithm for other classes of constraint networks involving resources either in isolation [20, 21, 23] or in conjunction with time [4].

References

1   Rakesh Agrawal, Dimitrios Gunopulos, and Frank Leymann. Mining process models from workflow logs. In *EDBT '98*, pages 469–483. Springer, 1998.

2   Massimo Cairo, Luke Hunsberger, Roberto Posenato, and Romeo Rizzi. A streamlined model of conditional simple temporal networks - semantics and equivalence results. In *TIME 2017*, volume 90 of *LIPIcs*, pages 10:1–10:19. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. `doi:10.4230/LIPIcs.TIME.2017.10`.

3   Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato, and Marco Roveri. Dynamic controllability via timed game automata. *Acta Informatica*, 53(6):681–722, 2016.

4   Carlo Combi, Roberto Posenato, Luca Viganò, and Matteo Zavatteri. Conditional simple temporal networks with uncertainty and resources. *Journal of Artificial Intelligence Research*, 64:931–985, 2019. `doi:10.1613/jair.1.11453`.

5   Patrick R. Conrad and Brian C. Williams. Drake: An efficient executive for temporal plans with choice. *JAIR*, 42(1):607–659, 2011.

6   Jonathan E. Cook and Alexander L. Wolf. Discovering models of software processes from event-based data. *TOSEM*, 7(3):215–249, 1998.

7   Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artif. Intell.*, 49(1-3):61–95, 1991.

8   Joachim Herbst. A machine learning approach to workflow management. In *ECML '00*, pages 183–194. Springer, 2000.

9   Luke Hunsberger, Roberto Posenato, and Carlo Combi. The Dynamic Controllability of Conditional STNs with Uncertainty. In *PlanEx at ICAPS-2012*, pages 1–8, 2012.

10   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*, pages 4–18. IEEE CPS, 2015. `doi:10.1109/TIME.2015.26`.

11   Erez Karpas, Steven James Levine, Peng Yu, and Brian Charles Williams. Robust execution of plans for human-robot teams. In *ICAPS '15*, pages 342–346. AAAI Press, 2015.

12   Steven J. Levine and Brian C. Williams. Concurrent plan recognition and execution for human-robot teams. In *ICAPS '14*, pages 490–498. AAAI, 2014.

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

14   Wil M. P. van der Aalst. Daelemans. automated discovery of workflow models from hospital data. In *BNAIC '01*, pages 25–26, 2001.

15   Wil M. P. van der Aalst, Boudewijn F. van Dongen, Joachim Herbst, Laura Maruster, Guido Schimm, and A. J. M. M. Weijters. Workflow mining: A survey of issues and approaches. *Data Knowl. Eng.*, 47(2):237–267, 2003.

16   Wil M. P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow mining: Which processes can be rediscovered? In *Eindhoven University of Technology*, pages 1–25, 2002.

17   Thierry Vidal and Hélène Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Jour. of Exp. & Theor. Artif. Intell.*, 11(1):23–45, 1999. `doi:10.1080/095281399146607`.

18   Peng Yu, Cheng Fang, and Brian Charles Williams. Resolving uncontrollable conditional temporal problems using continuous relaxations. In *ICAPS '14*, pages 341–349. AAAI, 2014.

19   Matteo Zavatteri. Conditional simple temporal networks with uncertainty and decisions. In *TIME 2017*, volume 90 of *LIPIcs*, pages 23:1–23:17. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. `doi:10.4230/LIPIcs.TIME.2017.23`.

20   Matteo Zavatteri, Romeo Rizzi, and Tiziano Villa. Dynamic Controllability and (J,K)-Resiliency in Generalized Constraint Networks with Uncertainty. In *ICAPS 2020*, pages 314–322. AAAI Press, 2020.

21   Matteo Zavatteri and Luca Viganò. Constraint networks under conditional uncertainty. In *ICAART 2018*, pages 41–52. SciTePress, 2018. `doi:10.5220/0006553400410052`.

22   Matteo Zavatteri and Luca Viganò. Conditional simple temporal networks with uncertainty and decisions. *Theoretical Computer Science*, 797:77–101, 2019. `doi:10.1016/j.tcs.2018.09.023`.

23   Matteo Zavatteri and Luca Viganò. Conditional uncertainty in constraint networks. In *Agents and Artificial Intelligence*, pages 130–160. Springer, 2019. `doi:10.1007/978-3-030-05453-3_7`.