An Algorithm for Checking the Dynamic Controllability of a Conditional Simple Temporal Network with Uncertainty

Carlo Combi[1], Luke Hunsberger[2*], Roberto Posenato[1]
[1]*Department of Computer Science, University of Verona, strada le grazie, Verona, Italy*
[2]*Computer Science Department, Vassar College, Poughkeepsie, NY, USA*
{*carlo.combi, roberto.posenato*}*@univr.it, hunsberg@cs.vassar.edu*

Keywords: Temporal Network : Temporal Controllability : Temporal Uncertainty : Temporal Workflow.

Abstract: A Simple Temporal Network with Uncertainty (STNU) is a framework for representing and reasoning about temporal problems involving actions whose durations are bounded but uncontrollable. A dynamically controllable STNU is one for which there exists a strategy for executing its time-points that guarantees that all of the temporal constraints in the network will be satisfied no matter how the uncontrollable durations turn out. A Conditional Simple Temporal Network with Uncertainty (CSTNU) augments an STNU to include observation nodes, where the execution of each observation node provides, in real time, the truth value of an associated proposition. Recent work has generalized the notion of dynamic controllability to cover CSTNUs. This paper presents an algorithm—called a DC-checking algorithm—for determining whether arbitrary CSTNUs are dynamically controllable. The algorithm, which is proven to be sound, is the first such algorithm to be presented in the literature. The algorithm extends edge-generation/constraint-propagation rules from an existing STNU algorithm to accommodate propositional labels, while adding new rules required to deal with the observation nodes. The paper also discusses implementation issues associated with the management of propositional labels.

1 Introduction

Workflow systems have been used to model business, manufacturing and medical-treatment processes. To meet the needs of such domains, Combi et al. (2010) presented a new workflow model that accommodates tasks with uncertain/uncontrollable durations; temporal constraints among tasks; and branching paths, where the branch taken is not known in advance. Subsequently, Hunsberger et al. (2012) introduced a *Conditional Simple Temporal Network with Uncertainty (CSTNU)* to represent the key features of that workflow model. The important property of *dynamic controllability* for CSTNUs was also defined. A CSTNU is dynamically controllable if there exists a strategy for executing the tasks in the associated workflow in a way that ensures that all temporal constraints will be satisfied no matter how the uncontrollable durations or branching events turn out.

This paper presents a *DC-checking algorithm* for CSTNUs (i.e., an algorithm for checking whether arbitrary CSTNUs are dynamically controllable). It is the first such algorithm in the literature. The algorithm, which is proven to be sound, extends the DC-checking algorithm for a simpler class of networks, called STNUs, developed by Morris and Muscettola (2005). It propagates *labeled values* on graph edges in a way that draws from prior work by Conrad and Williams (2011).

*Funded in part by the Phoebe H. Beadle Science Fund.

2 Motivating Example

In the following, we will consider, as a motivating example, a process taken from the healthcare domain. More precisely, consider the excerpt from a workflow schema depicted in Fig. 1, which follows the model proposed by Combi and Posenato (2009).

The *workflow schema* is a directed graph where nodes correspond to *activities* and edges represent *control flows* that define dependencies on the order of execution. There are two types of activity: *tasks* and *connectors. Tasks* represent elementary work units that will be executed by external agents. Each task is represented graphically by a rounded box and has a mandatory *duration* attribute that specifies the allowed temporal spans for its execution. Typically, the duration of a task is not controlled by the system responsi-

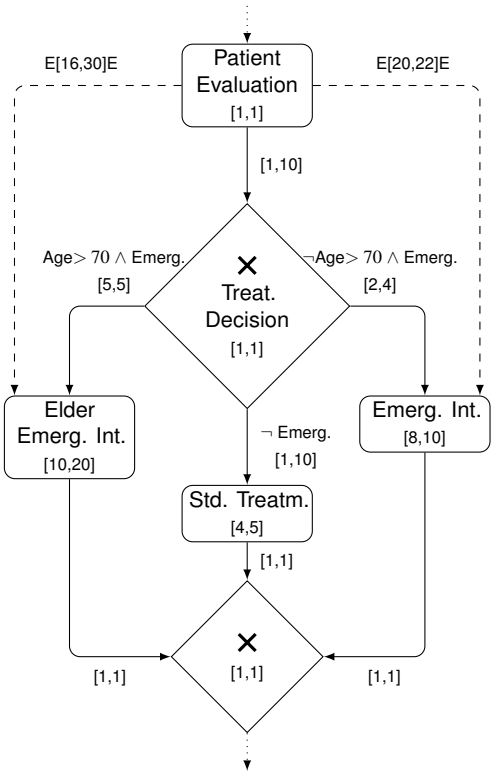


Figure 1: An excerpt of a healthcare workflow schema.

ble for managing the overall execution of the workflow (i.e., the Workflow Management System, WfMS). Unlike a task, a *connector* represents an internal activity whose execution *is* controlled by the WfMS. In particular, the WfMS uses connectors to coordinate the execution of the tasks. Connectors are represented graphically by diamonds. Like tasks, each connector has a mandatory *duration* attribute that specifies allowable temporal spans for its execution. However, unlike tasks, the WfMS can choose the value of each connector duration dynamically, in real time, to facilitate the coordination of the tasks in the workflow.

There are two kinds of connectors: *split* and *join*. *Split* connectors are nodes with one incoming edge and two or more outgoing edges. After the execution of the predecessor, possibly several successors have to be considered for execution. The set of nodes that can start their execution is determined by the kind of split connector. A *split* connector can be: *Total*, *Alternative* or *Conditional*. *Join* connectors are nodes with two or more incoming edges and only one outgoing edge. A *join* connector can be either *And* or *Or*.

Control flow is governed by oriented edges. Each oriented edge connects two activities, where the execution of the first activity (the *predecessor*) must be finished before starting the execution of the second one. Every edge has a *delay* attribute that specifies the allowed times that can be spent by the WfMS for possibly delaying the execution of the second activity.

Besides the temporal constraints associated with the duration and delay attributes of tasks, connectors and edges, a workflow schema can also include *relative constraints*. A relative constraint constrains the temporal interval between (the starting or ending time-points of) two non-consecutive workflow activities. Graphically, a relative constraint is represented by a directed edge from one activity to another, labeled by an expression of the form, $\mathsf{t}_1[\mathsf{MinD},\mathsf{MaxD}]\mathsf{t}_2$, where $t_1 \in \{S,E\}$ specifies whether the constraint applies to the starting or ending time-point of the first activity; $t_2 \in \{S,E\}$ specfies whether the constraint applies to the starting or ending time-point of the second activity; and $[\mathsf{MinD},\mathsf{MaxD}]$ specifies the allowed range for the temporal interval between the specified time-points.

The graph instance in Fig. 1 is a small excerpt from a process in a clinical domain. After the initial task, *Patient Evaluation,* whereby a physician determines whether the patient is in need of immediate medical attention (emergency state), there is an *alternative* connector, labeled *Treatment Decision*, from which three different treatment paths are possible, depending upon the age and emergency status of the patient. The three different treatments involve the following tasks: (1) *Elder Emergency Intervention*, (2) *Standard Treatment*, and (3) *Emergency Intervention.* The times at which the *Elder Emergency Intervention* and *Emergency Intervention* tasks must be completed, relative to the initial *Patient Evaluation* task, are restricted by the relative temporal constraints emanating from the *Patient Evaluation* node. These constraints are labeled $\mathsf{E}[16,30]\mathsf{E}$ and $\mathsf{E}[20,22]\mathsf{E}$, respectively, in the figure.

Given a particular workflow schema, it is important to determine in advance whether the WfMS is able to successfully execute the tasks in the schema, while observing all relevant temporal constraints, no matter how the durations of the tasks turn out. (Task durations are typically not controllable by the WfMS.) It is interesting to observe that the overall workflow schema in Fig. 1 may not be successfully executed by the WfMS for some possible task durations, even though each possible workflow subschema (or workflow path) is controllable when age and emergency status are known before execution begins.

A CSTNU is a more general formalism that allows the representation of all kinds of temporal constraints for workflow execution. In the following, after some background on related kinds of temporal networks, we will discuss CSTNUs and a new algorithm for determining the dynamic controllability of CSTNUs.

3 Background

Dechter et al. (1991) introduced Simple Temporal Networks (STNs). An STN is a set of time-point variables (or time-points) together with a set of simple temporal constraints, where each constraint has the form $Y - X \leq \delta$, where X and Y are time-points and δ is a real number. The *all-pairs, shortest-paths* matrix for the associated graph is called the *distance matrix* for the STN. For any STN, the following statements are equivalent:

- The STN has a solution (i.e., a set of values for the time-points that satisfy all of the constraints);
- The associated graph has no negative loops; and
- The distance matrix has zeros on its main diagonal.

Morris et al. (2001) presented Simple Temporal Networks with Uncertainty (STNUs) that augment STNs to include *contingent links* that represent uncontrollable-but-bounded temporal intervals. They gave a formal semantics for the important property of *dynamic controllability,* which holds if there exists a strategy for executing the time-points in the network that guarantees that all of the constraints will be satisfied no matter how the contingent durations turn out.[2] Crucially, the durations of contingent links are observed in real-time, as they complete; execution decisions can only depend on past observations.

Morris et al. (2001) also presented a *pseudo-polynomial-time* algorithm—called a *DC-checking algorithm*—for determining whether any given STNU is *dynamically controllable (DC).* Later, Morris and Muscettola (2005) presented the first polynomial DC-checking algorithm, which operates in $O(N^5)$ time. Because this algorithm plays an important role in this paper, it will henceforth be called the MM5 algorithm. Morris (2006) subsequently presented an $O(N^4)$-time DC-checking algorithm for STNUs, but it will not be discussed further in this paper.

Tsamardinos et al. (2003) introduced the Conditional Temporal Problem (CTP) which augments STNs to include *observation nodes*. When an observation node is executed, the truth value of its associated proposition becomes known. They presented a formal semantics for the important property of *dynamic consistency* which holds if there exists a strategy for executing the time-points in the network that guarantees that all of the constraints will be satisfied no matter how the observations turn out. Crucially, the truth values of propositions associated with observation nodes only become known in real time, as the observation nodes are executed. Tsamardinos et al. (2003) showed how to convert the semantic constraints inherent in the definition of dynamic consistency into a Disjunctive Temporal Problem (DTP). They then used an off-the-shelf DTP solver to determine the dynamic consistency of the original network in exponential time.

Hunsberger et al. (2012) combined the features of STNUs and CTPs to produce a Conditional Simple Temporal Network with Uncertainty (CSTNU). They proved that their definition of a CSTNU generalizes both STNUs and CTPs. In addition, they introduced a definition of dynamic controllability for CSTNUs that they proved generalizes the corresponding notions for STNUs and CTPs. They noted that because the existing DC-checking algorithms for STNUs and CTPs work so differently, they could not be easily combined to yield a DC-checking algorithm for CSTNUs. They also suggest that a new kind of algorithm has to be defined that incorporates new edge generation rules that take into account the propositional truth values generated by the observation nodes. In preparation for this kind of algorithm, they presented a *Label-Modification* rule for edges in a CSTNU that loosely resembles the *Label-Removal* rule for STNUs used by Morris and Muscettola.

This paper presents a DC-checking algorithm for CSTNUs that follows the proposal mentioned above. It extends the edge-generation/constraint-propagation MM5 algorithm for STNUs to accommodate observation nodes whose execution makes known the truth values of their associated propositions in real time. The algorithm, called the CSTNU DC-checking algorithm, generates edges that are labeled by propositions associated with observation nodes. Because there can be multiple such labeled edges between any pair of time-points, the algorithm carefully manages the potentially-exponential explosion of labels using techniques inspired by the work of Conrad and Williams (2011).

3.1 DC-Checking for STNUs

Following Morris et al. (2001), an STNU is a set of time-points and temporal constraints, like those in an STN, together with a set of contingent links. Each contingent link has the form, (A, x, y, C), where A and C are time-point variables (or time-points) and $0 < x < y < \infty$. A is called the *activation time-point*; C is the *contingent time-point*. Once A is executed, C is guaranteed to execute such that $C - A \in [x, y]$. However, the particular time at which C executes is uncontrollable. Instead, it is only observed as it happens.

Let $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ be an STNU, where $\mathcal{T}$ is a set of time-points, $\mathcal{C}$ is a set of constraints, and $\mathcal{L}$ is a set of contingent links. The graph associated with $\mathcal{S}$ has the form, $(\mathcal{T}, \mathcal{E}, \mathcal{E}_\ell, \mathcal{E}_u)$, where each time-point in $\mathcal{T}$

[2] Hunsberger (2009) subsequently corrected a minor flaw in the semantics of dynamic controllability.

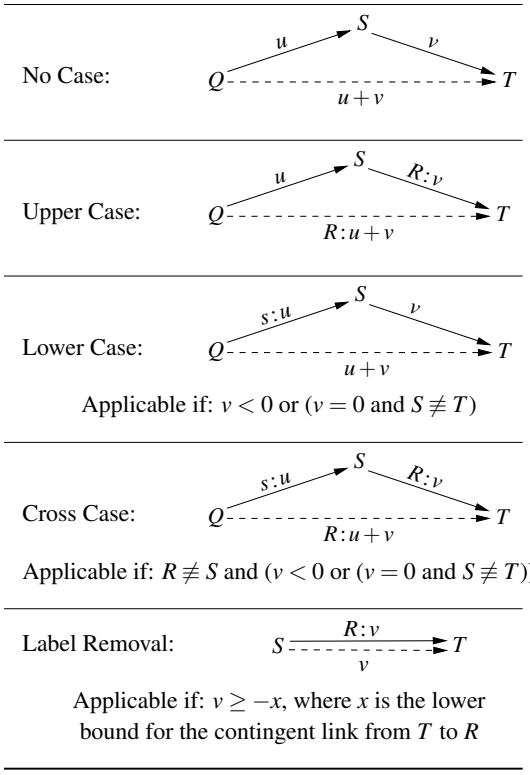


Table 1: Edge-generation rules for the MM5 algorithm. For each rule, the edge generated by the rule is dashed.

serves as a node in the graph; $\mathcal{E}$ is a set of *ordinary* edges; $\mathcal{E}_\ell$ is a set of *lower-case* edges; and $\mathcal{E}_u$ is a set of *upper-case* edges (Morris and Muscettola, 2005).

- Each ordinary edge has the form, $X \xrightarrow{v} Y$, representing the constraint, $Y - X \leq v$.
- Each lower-case edge has the form, $A \xrightarrow{c:x} C$, representing the *possibility* that the contingent duration, $C - A$, might take on its minimum value, x.
- Each upper-case edge, $C \xrightarrow{C:-y} A$, represents the *possibility* that the contingent duration, $C - A$, might take on its maximum value, y.

The MM5 algorithm works by recursively generating new edges in the STNU graph using the rules shown in Table 1. For each rule, pre-existing edges are denoted by solid arrows and newly generated edges are denoted by dashed arrows. Note that each of the first four rules takes two pre-existing edges as input and generates a single edge as its output. In contrast, the Label-Removal rule takes only one edge as input. Finally, applicability conditions of the form, $R \not\equiv S$, should be construed as stipulating that R and S must be distinct time-point variables, not as constraints on the *values* of those variables.

Procedure `MM5-DC-Check(G)`

Input: G: STNU graph instance to analyze.
Output: the controllability of G.
for 1 *to Cutoff_Bound* **do**
  if *(AllMax matrix inconsistent)* **then**
    return *false*;
  generate new edges using rules from Table 1;
  if *(no edges generated)* **then return** *true*;
end
return *false*

Note that the edge-generation rules only generate new ordinary or upper-case edges. Unlike the upper-case edges in the original graph, the upper-case edges generated by these rules represent conditional constraints, called *waits* (Morris et al., 2001). In particular, an upper-case edge, $Y \xrightarrow{C:-w} A$, represents a constraint that as long as the contingent time-point, C, remains unexecuted, then the time-point, Y, must wait at least w units after the execution of A, the activation time-point for C.

Procedure 1 gives pseudocode for the MM5 DC-checking algorithm. The algorithm performs at most $N^2 + NK + K = O(N^2)$ iterations, which is the number of distinct kinds of edges in a graph having N time-points and K contingent links. In each iteration, the algorithm first computes the *AllMax* matrix—which is the distance matrix for the STN formed by all of the original and generated, ordinary and upper-case edges (without their alphabetic labels)—and checks that there is no negative cycles in it and then applies the rules from Table 1 to all relevant combinations of edges of the STNU from the previous iteration. If no new edges are generated in any given iteration and there is no negative cycle at all, the algorithm reports that the network is dynamically controllable. If the algorithm continues generating new, stronger edges after the cutoff bound $N^2 + NK + K$, then the network cannot be DC. Since each iteration can be done in $O(N^3)$ time, the overall complexity of the MM5 algorithm is $O(N^5)$.

3.2 CSTNUs

A Conditional Simple Temporal Network with Uncertainty (CSTNU) is a network that combines the observation nodes and branching from a CTP with the contingent links of an STNU (Hunsberger et al., 2012). There is a one-to-one correspondence between observation nodes and propositional letters: the execution of an observation node generates a truth value for the corresponding proposition. However, nodes and edges in a CSTNU graph may be labeled by conjunctions of propositional literals. The time-point corresponding to

a node with label, ℓ, need only be executed in scenarios where ℓ is true. Similarly, the constraint corresponding to an edge with label, ℓ, is only applicable in scenarios where ℓ is true. The *label universe,* defined below, is the set of all possible labels.

Definition 1 (Label, Label Universe)**.** Given a set P of propositional letters, a *label* is any (possibly empty) conjunction of (positive or negative) literals from P. For convenience, the empty label is denoted by $\boxdot$. The *label universe* of P, denoted by P^*, is the set of all labels whose literals are drawn from P.

In the following, when not specified, lower-case Latin letters will denote propositions of P, while Greek lower-case letters will denote labels of P^*.

Definition 2 (Consistent labels, label subsumption)**.**

- Labels, ℓ_1 and ℓ_2, are called *consistent*, denoted by $Con(\ell_1,\ell_2)$, if and only if $\ell_1 \wedge \ell_2$ is satisfiable.
- A label ℓ_1 *subsumes* a label ℓ_2, denoted by $Sub(\ell_1,\ell_2)$, if and only if $\models (\ell_1 \Rightarrow \ell_2)$.

The following definition of a CSTNU is extracted from Hunsberger et al. (2012). The most important ingredients of a CSTNU are: $\mathcal{T}$, a set of time-points; $\mathcal{C}$, a set of *labeled* constraints; $\mathcal{OT}$, a set of observation time-points; and $\mathcal{L}$ a set of contingent links.

Definition 3 (CSTNU)**.** A Conditional STN with Uncertainty (CSTNU) is a tuple, $\langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, O, P, \mathcal{L} \rangle$, where:

- $\mathcal{T}$ is a finite set of real-valued time-points;
- P is a finite set of propositional letters;
- $L : \mathcal{T} \rightarrow P^*$ is a function that assigns a label to each time-point in $\mathcal{T}$;
- $\mathcal{OT} \subseteq \mathcal{T}$ is a set of observation time-points;
- $O : P \rightarrow \mathcal{OT}$ is a bijection that associates a unique observation time-point to each propositional letter;
- $\mathcal{L}$ is a set of contingent links;
- $\mathcal{C}$ is a set of *labeled* simple temporal constraints, each having the form, $(Y - X \leq \delta,\ \ell)$, where $X, Y \in \mathcal{T}$, δ is a real number, and $\ell \in P^*$;
- for any $(Y - X \leq \delta,\ \ell) \in \mathcal{C}$, the label ℓ is satisfiable and subsumes both $L(X)$ and $L(Y)$;
- for any $p \in P$ and $T \in \mathcal{T}$, if p or $\neg p$ appears in T's label, then
  - $Sub(L(T), L(O(p)))$, and
  - $(O(p) - T \leq -\varepsilon,\ L(T)) \in \mathcal{C}$, for some $\varepsilon > 0$;
- for each $(Y - X \leq \delta,\ \ell) \in \mathcal{C}$ and each $p \in P$, if p or $\neg p$ appears in ℓ, then $Sub(\ell, L(O(p)))$; and
- $(\mathcal{T}, \lfloor \mathcal{C} \rfloor, \mathcal{L})$ is an STNU, where $\lfloor \mathcal{C} \rfloor$ is the following set of unlabeled constraints: $\{(Y - X \leq \delta) \mid (Y - X \leq \delta, \ell) \in \mathcal{C} \text{ for some } \ell\}$.

The graph for a CSTNU is similar to that for an STNU except that some of the nodes may be observation nodes; and there may be propositional labels on nodes and edges. If p is a proposition, then the observation node whose execution generates a truth value for p shall be denoted by P? The propositional label of a node is usually represented near the node name, enclosed in square brackets. For example, a node labeled by $[cd]$ is only applicable to scenarios where propositions c and d are both true. Since edges in a CSTNU graph can have both propositional labels (associated with observation nodes) and alphabetic labels (associated with lower-case and upper-case edges in an STNU), these different kinds of labels are clearly distinguished in the *labeled values* for an edge, as follows.

Definition 4 (Labeled values)**.** A *labeled value* is a triple, $\langle PLabel,\ ALabel,\ Num \rangle$, where:

- $PLabel \in P^*$ is a propositional label,
- *ALabel*, an alphabetic label, is one of the following: (1) an upper-case letter, C, as on an upper-case edge in an STNU; (2) a lower-case letter, c, as on a lower-case edge in an STNU; or (3) $\diamond$, representing no alphabetic label, as for an ordinary STN edge.
- *Num* is a real number.

For example, $\langle p\neg q,\ c,\ 3 \rangle$ is a labeled lower-case edge; $\langle pq\neg r,\ C,\ -8 \rangle$ is a labeled upper-case edge; and $\langle \neg p,\ \diamond,\ 2 \rangle$ is a labeled ordinary edge.

Fig. 2 shows a sample CSTNU that represents a possible mapping of the main part of the workflow schema of Fig. 1. Initially, each ordinary edge in the network has only one labeled value, while each edge associated with a contingent link has two labeled values: one representing an ordinary STN constraint and the other representing an upper-case or lower-case STNU constraint. However, the new edge-generation rules given below will typically result in situations where a single edge may have numerous labeled values associated with it. The graph in the figure includes two observation nodes and three contingent links. Observation node A? generates a truth value for the proposition, a, which represents that the patient in question is over age 70. Observation node B? generates a truth value for the proposition, b, which represents that the patient is in need of immediate medical attention. The contingent link, $(C, 10, 20, D)$, represents an *Elder Emergency Intervention* task that takes between 10 and 20 minutes; the contingent link, $(H, 4, 5, I)$, represents a *Standard Treatment* task that takes between 4 and 5 minutes; and the contingent link, $(E, 8, 10, F)$, represents an *Emergency Intervention* task that takes between 8 and 10 minutes. To simplify the graph, only the lower-case and upper-case edges for each contin-

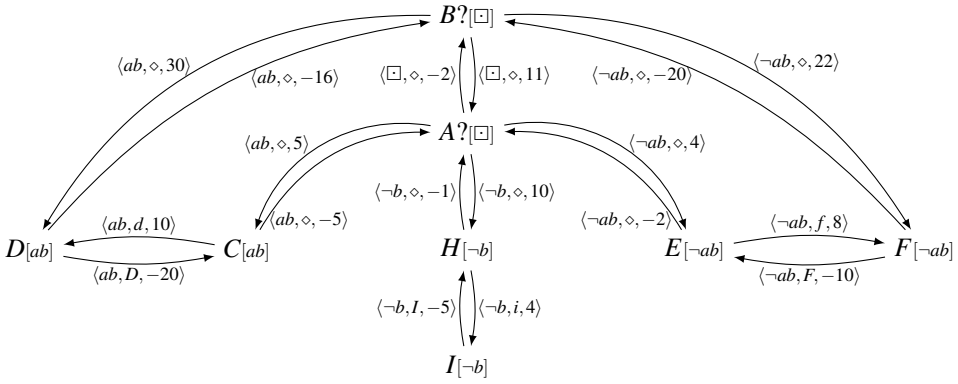


Figure 2: A possible CSTNU graph mapping the main part of the workflow schema of Fig. 1.

gent link are explicitly represented.[3] All other edges in the sample CSTNU represent ordinary temporal constraints. For example, the edges between $B?$ and $A?$ represent that the observation of proposition a must occur between 2 and 11 minutes after the observation of proposition b.

As defined in Hunsberger et al. (2012), a *scenario* s is a label that specifies a truth value for every propositional letter. The STNU formed by the nodes and edges (i.e., time-points and constraints) whose labels are true in a given scenario is called a *projection* of the CSTNU onto that scenario. A *situation* ω for an STNU specifies fixed durations for all of the contingent links. A *drama* (s, ω) is a scenario/situation pair that specifies fixed truth values for all of the propositional letters and fixed durations for all of the contingent links.

An *execution strategy* is a mapping from dramas to *schedules.* A schedule assigns an execution time to all of the time-points. Thus, if σ is an execution strategy and (s, ω) is drama, then $\sigma(s, \omega)$ is a schedule. For any time-point X, $[\sigma(s, \omega)]_X$ denotes the execution time assigned to X by the strategy σ in the drama (s, ω). A *dynamic* execution strategy is one in which the execution times assigned to non-contingent time-points only depends on *past* observations. A CSTNU is dynamically controllable if it has a dynamic execution strategy that guarantees the satisfaction of all temporal constraints no matter which drama unfolds in real time.

Note that a constraint whose propositional label is ℓ need only be satisfied in scenarios where ℓ is true. Similarly, a constraint whose alphabetic label is C need only be satisfied while C remains unexecuted.

Each of the STNUs obtained by projecting the sample CSTNU of Fig. 2 onto the scenarios, $ab, \neg ab$ and $\neg b$, is dynamically controllable—as an STNU. However, as will be shown below, the sample CSTNU is not dynamically controllable—as a CSTNU. This conforms to the observation by Combi and Posenato (2010) that the independent controllability of each *path* through a workflow is a necessary, but insufficient condition for the controllability of the entire workflow. For the workflow in Fig. 1, it turns out that there is no execution time for the observation node, $A?$, that will enable the rest of the network to be safely executed no matter how subsequent observations turn out.

[3] As proven elsewhere (Hunsberger, 2013), the ordinary edges associated with contingent links are not needed for the purposes of DC checking.

4 DC-Checking for CSTNUs

This section presents a DC-checking algorithm for CSTNUs. The basic approach is to extend the MM5 algorithm for STNUs to accommodate propositional labels. The presence of observation nodes also requires some new *label-modification* rules. In addition, since the propagation of labeled values involves conjoining labels, which can lead to an exponential number of labeled values, the paper also addresses the management of sets of labeled values.

In general, a label is said to be *enabled* in scenarios where the propositions composing the label are not false. For example, if we consider two scenarios S_1, where propositions $A, \neg B$ are true, C is false and D is unknown, and S_2, where only $\neg C$ is true and all others are unknown, it results that label ACD is enabled only in S_1 while label $\neg B$ is enabled in both S_1 and S_2.
During the execution of a CSTNU instance, to enact a time-point (node), it is necessary to consider all *enabled* labeled constraints involving the considered node and to verify that such labeled constraints are satisfied. In other words, it is possible that more than one labeled constraint between two nodes are enabled because associated to scenarios compatible with the current partial scenario and, therefore, all of them have to be satisfied. Hence, it is necessary to generate all possible constraints (edges) for all possible (partial)

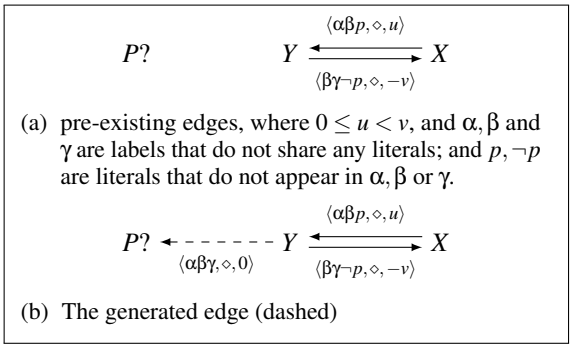


Figure 3: The Observation Case Rule (cf. Lemma 4.1)

scenarios in order to evaluate if a CSTNU is DC controllable.

Hereinafter, we indifferently denote such set of constraints as a set of different labeled constraints or as different labeled values of the same constraint.

4.1 Edge Generation for CSTNUs

The edge generation rules for CSTNUs may be divided in two main groups: the first one extends and completes the edge generation rules of MM5 algorithm to the case of labeled (possibly alternative) constraints; the second one modifies the labels of some constraints in order to allow the execution of different future scenarios.

4.1.1 Labeled Constraint Generation

We begin by modifying the edge-generation rules for STNUs to accommodate labels on edges, as shown in Table 2. Note that each of the first four rules generates an edge whose PLabel is the conjunction of the PLabels of its parent edges. If the resulting PLabel is unsatisfiable (e.g., $p\neg p$), then the new edge is not generated (or kept). The fifth rule allows us to remove the upper case label in order to make the constraint an ordinary one.

As for the last rule, when there is a constraint from X to Y with label ℓ and one from Y to X with a label not consistent with ℓ, it may be that we have to add a new constraint in order to guarantee the controllability, as shown in the following lemma.

Lemma 4.1 (Observation Case Rule)**.** Let σ be a dynamic execution strategy that satisfies the labeled constraint in Fig. 3-(a) in all scenarios where their labels are true, then σ must also satisfy the labeled constraint $(P?-Y\leq 0, \alpha\beta\gamma)$ in all scenarios where $\alpha\beta\gamma$ is true as shown in Fig. 3-(b).

Proof. Let σ be as in the statement of the lemma. However, suppose that there is some drama, (s,ω),

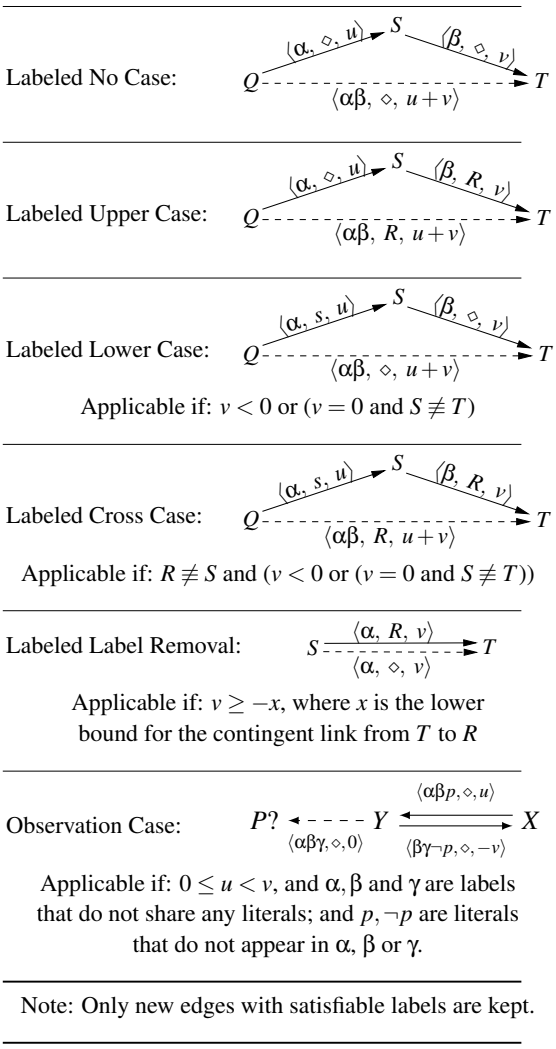


Table 2: New edge-generation rules for CSTNUs.

such that: (1) the label $\alpha\beta\gamma$ is true in scenario s; but (2) the schedule $\sigma(s,\omega)$, does *not* satisfy the constraint, $(P?-Y\leq 0, \alpha\beta\gamma)$. In that case, $P?-Y>0$, which implies that $P?>Y$. Since X precedes Y ($X-Y\leq -v$), both X and Y precede $P?$.

Next, let $\tilde{s}$ be the same scenario as s except that the truth value of p is flipped. Let t be the first time at which the schedules, $\sigma(s,\omega)$ and $\sigma(\tilde{s},\omega)$, differ. Thus, there must be some time-point T that is executed in one of the schedules at time t, and in the other at some time later than t. But in that case, the corresponding histories at time t must be different. But the only possible difference must involve the value of the proposition $P?$, since all other propositions and contingent durations are identical in the dramas, (s,ω) and $(\tilde{s},\omega)$. Thus, $P?$ must be executed before time t. Now, in the schedule $\sigma(s,\omega)$, we have

seen that both Y and X are executed before $P?$, and hence before t. Thus, $[\sigma(s,\omega)]_X = [\sigma(\tilde{s},\omega)]_X$ and $[\sigma(s,\omega)]_Y = [\sigma(\tilde{s},\omega)]_Y$. But this cannot be possible because in one scenario $Y - X \leq u$ and in the other one $Y - X \geq v$ and both constraints cannot be satisfied at the same time since $u < v$. □

4.1.2 Label Modification

This section introduces a variety of *label-modification* rules that share some resemblance to the Label-Removal rule in Table 2. Thus, we begin with a short description of the Label-Removal rule.

Suppose a CSTNU contains a contingent link, $(A, 5, 12, C)$. In other words, the contingent duration, $C - A$, is uncontrollable, but guaranteed to be within the interval, $[5, 12]$. Suppose further that the network also contains the upper-case edge, $Y \xrightarrow{C:-2} A$, which represents the following *wait* constraint: "As long as the contingent time-point C remains unexecuted, then Y must wait at least 2 units after the execution of its activation time-point, A. Given that the minimum duration of this contingent link is 5, it follows that the contingent time-point C must remain unexecuted until after the wait time of 2 has expired. As a result, the decision to execute Y must, in every scenario, wait at least 2 units after A. For this reason, the Label-Removal rule generates the ordinary edge, $Y \xrightarrow{-2} A$, which represents the unconditional constraint, $A - Y \leq -2$ (i.e., $Y \geq A + 2$). This example illustrates that in certain scenarios, a constraint conditioned on an uncontrollable event—in this case, the execution of the contingent time-point C—might have the force of an unconditional constraint because the uncertainty associated with the uncontrollable event will definitely not be resolved at the time a particular execution decision—in this case, the decision to execute Y—must be made.

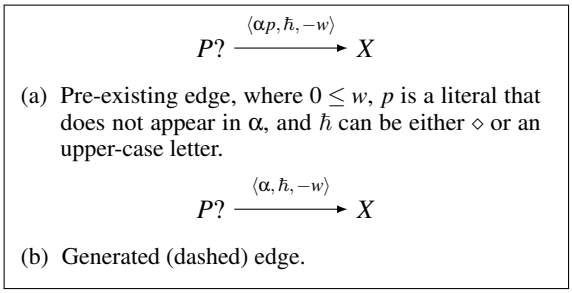
The label-modification rules presented below have the same general flavor, except that they deal with the uncertainty associated with observation nodes, rather than contingent links. For example, consider the edge, $P? \xrightarrow{\langle \alpha p,\ \diamond,\ -w \rangle} X$, where neither p nor $\neg p$ appears in α, and $w \geq 0$. This edge represents the conditional constraint that "in scenarios where αp is true, $X - P? \leq -w$ (i.e., $X + w \leq P?$) must hold". Given that $w \geq 0$, it follows that in scenarios where αp is true, X must be executed before the observation node $P?$. But it implies that the truth value of p cannot be known at the time X is executed. And, of course, the truth value of p cannot be known when the decision to execute $P?$ is made either. As a result, decisions about when to execute X and $P?$ cannot depend on the truth value of p. Thus, the PLabel on the edge from $P?$ to X should be modified to remove the occurrence of p, yielding the new edge, $P? \xrightarrow{\langle \alpha,\ \diamond,\ -w \rangle} X$, which represents the constraint that in scenarios where α holds, $X - P? \leq -w$ (i.e., $X + w \leq P?$) must hold. This is the idea behind the label-modification rule, R0, shown in Table 3. For each rule, pre-existing labels are represented as usual, labels to be substituted are represented in a dashed box, while newly generated labels are depicted in a shaded box. The following lemma shows that this rule is sound.

R0 Case:	$P? \xrightarrow[\langle \alpha, \hbar, -w \rangle]{\boxed{\langle \alpha p, \hbar, -w \rangle}} X$
	Applicable if: $0 \leq w$, p is a literal not in α. $\hbar$ can be either $\diamond$ or an upper-case letter.
R1 Case:	$P? \xrightarrow{\langle \alpha\beta, \diamond, -w \rangle} X \xrightarrow[\substack{\langle \alpha\beta\gamma, \hbar, v \rangle \\ \langle \neg\alpha\beta\gamma p, \hbar, v \rangle}]{\boxed{\langle \beta\gamma p, \hbar, v \rangle}} Y$
	Applicable if: $0 \leq w, v \leq w$, and α, β and γ are labels that do not share any literals; and p is a literal that does not appear in α, β or γ. $\hbar$ can be either $\diamond$ or an upper-case letter.
R2 Case:	$P? \xrightarrow[\boxed{\langle \beta\gamma p, \hbar, v \rangle}]{\langle \alpha\beta, \diamond, -w \rangle} X$ $\langle \alpha\beta\gamma, \hbar, v \rangle$ $\langle \neg\alpha\beta\gamma p, \hbar, v \rangle$
	Applicable if: $0 \leq w, v \geq w$, and α, β and γ are labels that do not share any literals; and p is a literal that does not appear in α, β or γ. $\hbar$ can be either $\diamond$ or an upper-case letter.
R3 Case:	$P? \xrightarrow{\langle \alpha\beta, \diamond, -w \rangle} X \xleftarrow[\substack{\langle \alpha\beta\gamma, \hbar, -v \rangle \\ \langle \neg\alpha\beta\gamma p, \hbar, -v \rangle}]{\boxed{\langle \beta\gamma p, \hbar, -v \rangle}} Y$
	Applicable if: $0 \leq w, v \leq w$, and α, β and γ are labels that do not share any literals; and p is a literal that does not appear in α, β or γ. $\hbar$ can be either $\diamond$ or an upper-case letter.

Table 3: Label-modification rules for CSTNUs. Each dashed-boxed PLabel is substituted by one(s) in the shadow box.

Lemma 4.2 (Label-Modification Rule, R0)**.** Suppose that $w \geq 0$ and α is a label that does not contain the literal p. If σ is a dynamic execution strategy that satisfies the labeled constraint, $(X - P? \leq -w, \alpha p)$ an shown in Fig. 4-(a), in all scenarios where αp is true, then σ must also satisfy the labeled constraint, $(X - P? \leq -w, \alpha)$, in all scenarios where α is true, as depicted in Fig. 4-(b).

Proof. Let (s, ω) be a drama such that: (1) the label



(a) Pre-existing edge, where $0 \leq w$, p is a literal that does not appear in α, and $\hbar$ can be either $\diamond$ or an upper-case letter.

(b) Generated (dashed) edge.

Figure 4: The Label-Modification rule, R0 (cf. Lemma 4.2).

$\alpha\neg p$ is true in scenario s; but (2) the schedule $\sigma(s,\omega)$ does *not* satisfy the constraint, $(X - P? \leq -w)$. In that case, $X + w > P?$. Next, let s' be the same scenario as s except that p is true in s'. Then αp is true in s', which implies that $(X - P? \leq -w)$ in (s',ω), and hence $X + w \leq P?$.

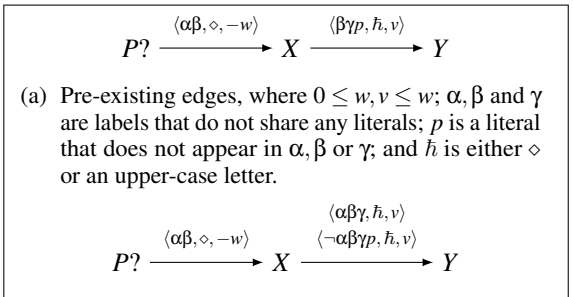
Next, let t be the first time at which the schedules, $\sigma(s,\omega)$ and $\sigma(s',\omega)$, differ. Thus, there must be some time-point T that is executed in one of the schedules at time t, and in the other at some time after t. But in that case, the corresponding histories at time t must be different. Since the dramas, (s,ω) and (s',ω), are identical except for the truth value of p, it follows that the observation node, $P?$, must be executed before time t. Now, in the drama (s',ω), the constraint, $X + w \leq P?$, is satisfied; thus, both X and $P?$ must be executed before time t in that drama. Since the schedules, $\sigma(s,\omega)$ and $\sigma(s',\omega)$, are identical prior to time t, it follows that the same constraint is satisfied by $\sigma(s',\omega)$, contradicting the choice of (s',ω). □

The rest of the label-modification rules have the same general flavor and they are summarized in Table 3. Rule R1 originally appeared in Hunsberger et al. (2012). The corresponding lemma, given below, shows that this rule is sound. Its proof is not repeated here.

Lemma 4.3 (Label-Modification Rule, R1 (Hunsberger et al., 2012))**.** Let σ be a dynamic execution strategy that satisfies the labeled constraint in Fig. 5-(a) in all scenarios where their labels are true, then σ must also satisfy the labeled constraint $(Y - X \leq v, \alpha\beta\gamma)$ in all scenarios where $\alpha\beta\gamma$ is true. The original constraint $(Y - X \leq v, \beta\gamma p)$ is replaced by the pair of labeled constraints, $(Y - X \leq v, \alpha\beta\gamma)$ and $(Y - X \leq v, \neg\alpha\beta\gamma p)$ as depicted in Fig. 5-(b);

Regarding the proof, here we only remark that when $v > w$, the rule cannot be applied because it could be possible to verify the constraint between X and Y after the execution of $P?$ and, therefore, to consider the original constraint at the due time.

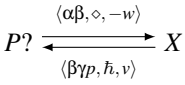
The lemma does not analyze the case when $P?$ and



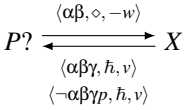
(a) Pre-existing edges, where $0 \leq w, v \leq w$; α, β and γ are labels that do not share any literals; p is a literal that does not appear in α, β or γ; and $\hbar$ is either $\diamond$ or an upper-case letter.

(b) New labels on the edge from X to Y.

Figure 5: The Label-Modification rule, R1 (cf. Lemma 4.3).



(a) Pre-existing edges, where $0 \leq w, v \geq w$; α, β and γ are labels that do not share any literals; p is a literal that does not appear in α, β or γ; and $\hbar$ is either $\diamond$ or an upper-case letter.



(b) New labels on the edge from Y to X.

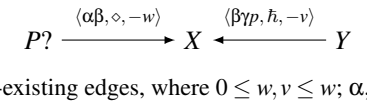
Figure 6: The Label-Modification rule, R2 (cf. Lemma 4.4).

Y are the same node: in such case the R1 cannot be applied: if $v < w$, we would have a negative cycle. However, we need to consider the case $P?$ and Y are the same node and $v \geq w$: label $\beta\gamma p$ has to be always considered before the execution of $P?$ and, therefore, it is necessary to propagate it without p. We call this rule R2.

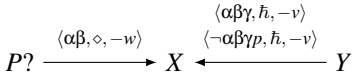
Lemma 4.4 (Label-Removal Rule (R2))**.** Let σ be a dynamic execution strategy that satisfies the labeled constraint in Fig. 6-(a) in all scenarios where their labels are true, then σ must also satisfy the labeled constraint $(Y - X \leq v, \alpha\beta\gamma)$ in all scenarios where $\alpha\beta\gamma$ is true. The original constraint $(Y - X \leq v, \beta\gamma p)$ is replaced by the pair of labeled constraints, $(Y - X \leq v, \alpha\beta\gamma)$ and $(Y - X \leq v, \neg\alpha\beta\gamma p)$ as depicted in Fig. 6-(b);

Proof. It is straightforward to prove the correctness of this label-modification rule, as it deals with the standard constraint between two ordered time-points. □

When there is a negative value on a constraint from Y to X, we have another case of label modification as shown in the following lemma.



(a) Pre-existing edges, where $0 \leq w, v \leq w$; α, β and γ are labels that do not share any literals; p is a literal that does not appear in α, β or γ; and $\hbar$ is either $\diamond$ or an upper-case letter.



(b) New labels on the edge from Y to X.

Figure 7: The Label-Modification rule, R3 (cf. Lemma 4.5).

Lemma 4.5 (Label-Modification Rule, R3)**.** Let σ be a dynamic execution strategy that satisfies the labeled constraint in Fig. 7-(a) in all scenarios where their labels are true, then σ must also satisfy the labeled constraint $(X - Y \leq -v, \alpha\beta\gamma)$ in all scenarios where $\alpha\beta\gamma$ is true. The original constraint $(X - Y \leq -v, \beta\gamma p)$ is replaced by the pair of labeled constraints $(X - Y \leq -v, \alpha\beta\gamma)$ and $(X - Y \leq -v, \neg\alpha\beta\gamma p)$ as depicted in Fig. 7-(b).

Proof. Let σ be as in the statement of the lemma. However, suppose that there is some drama, (s, ω), such that: (1) the label $\alpha\beta\gamma$ is true in scenario s; but (2) the schedule $\sigma(s, \omega)$, does *not* satisfy the constraint, $(X - Y \leq -v)$. In that case, $X - Y > -v$, which implies that $Y < X + v \leq X + w \leq P?$, since $v \leq w$, given that $\alpha\beta$ is true and the constraint from $P?$ to X is satisfied (i.e., $X - P? \leq -w$). Note also that $X \leq P?$.

Next, let $\tilde{s}$ be the same scenario as s except that the truth value of p is flipped. Let t be the first time at which the schedules, $\sigma(s, \omega)$ and $\sigma(\tilde{s}, \omega)$, differ. Thus, there must be some time-point T that is executed in one of the schedules at time t, and in the other at some time later than t. But in that case, the corresponding histories at time t must be different. But the only possible difference must involve the value of the proposition $P?$, since all other propositions and contingent durations are identical in the dramas, (s, ω) and $(\tilde{s}, \omega)$. Thus, $P?$ must be executed before time t. Now, in the schedule $\sigma(s, \omega)$, we have seen that both Y and X are executed before $P?$, and hence before t. Thus, $[\sigma(s, \omega)]_X = [\sigma(\tilde{s}, \omega)]_X$ and $[\sigma(s, \omega)]_Y = [\sigma(\tilde{s}, \omega)]_Y$. But then the value of $Y - X$ must be the same in both schedules. Thus, the constraint $X - Y \leq -v$ must be violated in both schedules. But this contradicts that the constraint $X - Y \leq -v$ is satisfied in scenarios where $\beta\gamma p$ is true.

Regarding the constraint $(X - Y \leq -v, \neg\alpha\beta\gamma p)$, similarly to (Hunsberger et al., 2012), it is simple to show that it is necessary to introduce it for maintaining the overall equivalence with respect to the original constraint: indeed, when α is not true, it is not known the relation between $P?$ and X. □

The application of rules R0, R1, R2 and R3 has to be considered for all pairs of time-points with respect to all suitable observation points.

All other combinations of constraints among $P?$, X and Y do not yield further constraints.

4.2 The CSTNU DC-Checking Algorithm

The CSTNU DC-checking algorithm checks whether a CSTNU instance is dynamically controllable (DC) trying to apply all possible labeled constraint generation rules of Table 2 and all possible label modification rules of Table 3 until either no more rules are possible or the associated AllMax matrix is inconsistent or the maximum number of cycles of rules application is reached. The pseudocode of the algorithm is shown in Procedure 2.

Procedure `CSTNU-DC-Check` (G)

Input: $G = \langle \mathcal{T}, \mathcal{C}, L, \mathcal{OT}, O, P, \mathcal{L} \rangle$: a CSTNU instance.
Output: the dynamic controllability of G.
$G' = G$;
for 1 *to* $|P|(|\mathcal{T}|^2 + |\mathcal{T}||\mathcal{L}| + |\mathcal{L}|)$ **do**
  if *(AllMax matrix of G is inconsistent)* **then**
  return *false*;
  // Label Modification Rules
  G =LabelModificationRuleR0(G);
  G =LabelModificationRuleR1(G);
  G =LabelModificationRuleR2(G);
  G =LabelModificationRuleR3(G);
  // Labeled Constraints Generation
  $G' = G' \cup$ needed LabeledNoCaseRule(G);
  $G' = G' \cup$ needed LabeledUpperCaseRule(G);
  $G' = G' \cup$ any LabeledCrossCaseRule(G);
  $G' = G' \cup$ any LabeledLowerCaseRule(G);
  $G' = G' \cup$ any LabeledLabelRemovalRule(G);
  $G' = G' \cup$ any ObservationCaseRule(G);
  if *(no rules were applied)* **then return** *true*;
  $G = G'$;
end
return *false*

The algorithm performs $p(n^2 + nk + k)$ rounds, where n is the number of time-points, k is the number of contingent ones and p is the number of propositional letters that appear in the network. In each round, the application of the labeled constraints generation rules is carried out considering as input edges those generated by the previous round. Input edges for the label modification rules are drawn from the graph obtained by the previous rule application of the same round. The reason of such difference in the considered

input is given by the different structure of the rules. The application of labeled constraint generation rules requires to add edges and/or labeled values to already present edges and, therefore, it is safe to consider the same starting graph, while the application of label modification rules requires to modify labels to some current labeled values: thus, it is not possible to make it starting from the graph of the previous round.

After those rounds have completed, if it is still possible to generate stronger constraints having the same labels, then the CSTNU is not DC. Proof of this is an easy extension of Morris and Muscettola's argument about the number of rounds in the MM5 algorithm.

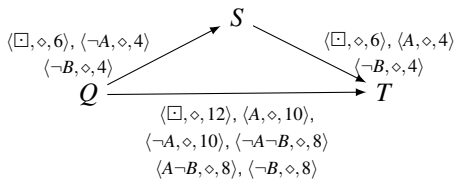
According to the previous lemmas, it is straightforward to show that the algorithm is sound: indeed, given a DC CSTNU instance, the algorithm always says that it is DC (since each rule by itself is sound, we know for sure that the entire algorithm is sound).

As for completeness, the algorithm should always say that a non DC CSTNU instance is not DC: even though we argue that our algorithm is complete, we are currently working on its formal proof, being it more complex and intricate than the soundness one.

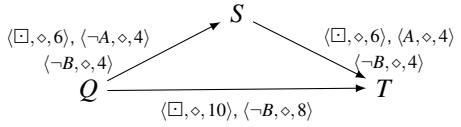
4.2.1 On the Management of PropLabels

The actual performance of the algorithm can also be affected by the management strategy of the labeled value sets of constraints. To better introduce the issue, let us consider the application of the No-Case rule to a pair of constraints containing different labeled values, as in the example of Fig. 8. Even though the new labeled values determined by the rule are stored guaranteeing only that, for each possible label, only the minimal value is stored, it is still possible to have an exponential explosion in the number of labeled values, as shown in Fig. 8-(a). Such exponential number is not always necessary because it is possible that some of them can be represented by only one: for example, $\langle \neg A, \diamond, 10 \rangle$ and $\langle A, \diamond, 10 \rangle$ represent the fact that any possible labeled value has to have value at most 10. Therefore, it is possible to substitute the labeled values $\langle \boxdot, \diamond, 12 \rangle$, $\langle \neg A, \diamond, 10 \rangle$, $\langle A, \diamond, 10 \rangle$ by $\langle \boxdot, \diamond, 10 \rangle$ as shown in Fig. 8-(b). In the following we propose some labeled value management rules in order to minimize the number of labeled values stored in each constraint set.

When there are two or more labeled values with labels that subsume the same "seed" label and they have all the same value, it is not necessary to represent all of them in a explicit way: it is sufficient to represent only the "seed" one. For example, the two labeled values of Fig. 8 $\langle A\neg B, \diamond, 8 \rangle$ and $\langle \neg B, \diamond, 8 \rangle$ can be represented by $\langle \neg B, \diamond, 8 \rangle$ only.



(a): no labeled value storing management.



(b): Optimal labeled value storing management.

Figure 8: Two different managements of labeled values in a no-case reduction rule application.

Rule 1 (Redundant Label Elimination 1 (RLE 1))**.** If a set of labeled values contains two labels (ℓ_1, i) and (ℓ_2, i) where ℓ_1 subsumes ℓ_2, then the labeled value (ℓ_1, i) is redundant and it can be removed.

The previous rule can be simply extended to the case when two labels differ for only one proposition:

Rule 2 (Redundant Label Elimination 2 (RLE 2))**.** If a set of labeled values contains two labeled values (ℓ_1, i) and (ℓ_2, j), where $\ell_1 = \alpha p$, $\ell_2 = \alpha \neg p$ and α does not contain p neither $\neg p$, there are two cases: (i) if $i = j$, then both labeled values can be represented by (α, i). (ii) if $i \neq j$, then remove, if any, α labeled value because it would be greater than both i and j (and thus not useful).

For example, in Fig. 8 $\langle A\neg B, \diamond, 8 \rangle$ and $\langle \neg A\neg B, \diamond, 8 \rangle$ can be substituted by $\langle \neg B, \diamond, 8 \rangle$.

Regarding empty label ($\boxdot$), as already said, it represents all the possible labels. If there is an empty-labeled value, such value has to be considered as the default value. If there are other labeled values, these last ones have to be smaller than the default value (otherwise, RLE 1 applies). It is possible to represent all possible combinations of labels not only with an empty label but even with a suitable set of labels. For example, in the set $\{\langle A, \diamond, 8 \rangle, \langle \neg A, \diamond, 8 \rangle, \langle \boxdot, \diamond, 10 \rangle\}$, the pair of labels $\langle A, \diamond, 8 \rangle$ and $\langle \neg A, \diamond, 8 \rangle$ represents all possible labels of the Universe and their values are smaller than the empty-labeled one: the empty labeled value can be removed. In general, it holds:

Rule 3 (Empty Label Elimination (RLE 3))**.** If a set of labeled values contains a subset of labels that are all possible combinations of a fixed set of propositions, then such subset represents the base of all possible labels. Therefore the possible empty-labeled value

has to be removed since, for construction, its value is greater than the values associated to the labels of the base.

The previous rules explain how to maintain a set of labeled values of a constraint in order to rightly represent all the possible values, while maintaining the minimal number of such values represented explicitly.

In general, if we have to add the labeled values of a set S_1 to the labeled values of a set S_2, it is necessary to add each value of the first set to each each value of the second one having a label that it is consistent with the label of the first value. The label of the sum of two values is the conjunction of the two considered labels. Such new labeled value has to be put in a new set that will represent the result of the operation. For example, given the two sets of Fig. 8, $S_1 = \{\langle\boxdot,\diamond,6\rangle,\langle\neg A,\diamond,4\rangle,\langle\neg B,\diamond,4\rangle\}$ and $S_2 = \{\langle\boxdot,\diamond,6\rangle,\langle A,\diamond,4\rangle,\langle\neg B,\diamond,4\rangle\}$, the sum of them is

$$S_1 + S_2 = \{\langle\boxdot,\diamond,6\rangle + \langle\boxdot,\diamond,6\rangle = \langle\boxdot,\diamond,12\rangle, \quad (1)$$
$$\langle\boxdot,\diamond,6\rangle + \langle A,\diamond,4\rangle = \langle A,\diamond,10\rangle, \quad (2)$$
$$\langle\boxdot,\diamond,6\rangle + \langle\neg B,\diamond,4\rangle = \langle\neg B,\diamond,10\rangle, \quad (3)$$
$$\langle\neg A,\diamond,4\rangle + \langle\boxdot,\diamond,6\rangle = \langle\neg A,\diamond,10\rangle, \quad (4)$$
$$\langle\neg A,\diamond,4\rangle + \langle A,\diamond,4\rangle = \text{not possible}, \quad (5)$$
$$\langle\neg A,\diamond,4\rangle + \langle\neg B,\diamond,4\rangle = \langle\neg A\neg B,\diamond,8\rangle, \quad (6)$$
$$\langle\neg B,\diamond,4\rangle + \langle\boxdot,\diamond,6\rangle = \langle\neg B,\diamond,10\rangle, \quad (7)$$
$$\langle\neg B,\diamond,4\rangle + \langle A,\diamond,4\rangle = \langle A\neg B,\diamond,8\rangle, \quad (8)$$
$$\langle\neg B,\diamond,4\rangle + \langle\neg B,\diamond,4\rangle = \langle\neg B,\diamond,8\rangle\} \quad (9)$$

The labeled value (5) is not possible because the conjunction of its labels is always false. Hence, it is necessary to apply possible label elimination rules. In particular, $\langle\neg B,\diamond,8\rangle$ makes $\langle\neg B,\diamond,10\rangle$ useless. Now, applying the RLE2 rule, it is possible to substitute $\langle A\neg B,\diamond,8\rangle$ and $\langle\neg A\neg B,\diamond,8\rangle$ by $\langle\neg B,\diamond,8\rangle$, already present. At last, applying the RLE1 rule, it is possible to substitute $\langle\neg A,\diamond,10\rangle$ and $\langle A,\diamond,10\rangle$ by $\langle\boxdot,\diamond,10\rangle$ that updates the already present $\langle\boxdot,\diamond,12\rangle$ Hence, the set becomes:

$$S_1 + S_2 = \{\langle\boxdot,\diamond,10\rangle,\langle\neg B,\diamond,8\rangle\}$$

5 Discussion and Conclusion

To verify and test the practical usability of the proposed algorithm, we have built a Java program, called CSTNUEDITOR, that allows one to graphically design a CSTNU instance and to check its dynamic controllability. Fig. **??** depicts a screen shot of the program running on a simple CSTNU instance.

The program implements different kinds of PropLabel management in order to better monitor the label propagation and its impact on the convergence of the algorithm.
First experiments show that the algorithm finds the solution in an average number of cycles one order of magnitude smaller than the theoretical estimated upper bound. Moreover, different policies in the PropLabel management have different consequences on the convergence of the algorithm: the number of cycles required to find a solution decreases when the PropLabel management minimizes (in any way) the number of stored labels but the running time of each cycle of the algorithm increases. It is under evaluation which is the best trade off between the (even partial) PropLabel management and the execution time.

In this article, we presented a *DC-checking algorithm* for CSTNUs: such algorithm is based on labeled constraint generation rules that extend the rules proposed in (Morris and Muscettola, 2005) and on new label modification rules, introduced to manage different possible alternative executions. It is the first such algorithm in the literature. The algorithm is proven to be sound and conjectured to be complete.

As for future work, we are going to formally analyze the completeness of the algorithm. Moreover, we will extensively test CSTNUEDITOR with synthetic and real world complex CSTNU networks, in order to evaluate its applicability in the area of temporal workflow systems.

REFERENCES

Combi, C. and Posenato, R. (2010). Towards temporal controllabilities for workflow schemata. In (Markey and Wijsen, 2010), pages 129–136.

Conrad, P. R. and Williams, B. C. (2011). Drake: An efficient executive for temporal plans with choice. *Journal of Artificial Intelligence Research (JAIR)*, 42:607–659.

Dechter, R., Meiri, I., and Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95.

Hunsberger, L. (2009). Fixing the semantics for dynamic controllability and providing a more practical characterization of dynamic execution strategies. In Lutz, C. and Raskin, J.-F., editors, *The 16th International Symposium on Temporal Representation and Reasoning (TIME-2009)*, pages 155–162. IEEE.

Hunsberger, L. (2010). A fast incremental algorithm for managing the execution of dynamically controllable temporal networks. In (Markey and Wijsen, 2010), pages 121–128.

Hunsberger, L. (2013). Magic loops in simple temporal networks with uncertainty. In *Fifth International Conference on Agents and Artificial Intelligence (ICAART-2013)*. SciTePress.

Hunsberger, L., Posenato, R., and Combi, C. (2012). The Dynamic Controllability of Conditional STNs with Un-

certainty. In *Workshop on Planning and Plan Execution for Real-World Systems: Principles and Practices (PlanEx) @ ICAPS-2012*, pages 1–8, Atibaia.

Markey, N. and Wijsen, J., editors (2010). *The Seventeenth International Symposium on Temporal Representation and Reasoning (TIME-2010)*. IEEE.

Morris, P. (2006). A structural characterization of temporal dynamic controllability. In Benhamou, F., editor, *Principles and Practice of Constraint Programming*, volume 4204 of *LNCS*, pages 375–389. Springer.

Morris, P. H. and Muscettola, N. (2005). Temporal dynamic controllability revisited. In Veloso, M. M. and Kambhampati, S., editors, *The Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 1193–1198. AAAI Press.

Morris, P. H., Muscettola, N., and Vidal, T. (2001). Dynamic control of plans with temporal uncertainty. In Nebel, B., editor, *The Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 494–502. Morgan Kaufmann.